OXFORD

Briefings in Bioinformatics, 19(1), 2018, 23–40

doi: 10.1093/bib/bbw096
Advance Access Publication Date: 14 October 2016
Paper

The present and future of *de novo* whole-genome assembly

Jang-il Sohn and Jin-Wu Nam

Corresponding author. Jin-Wu Nam, FTC 1123, Hanyang University, 222 Wangsimni-ro, Seongdong-gu, Seoul 04763, Korea. Tel.: +82-2-2220-2428; Fax: +82-2-2298-0319; E-mail: jwnam@hanyang.ac.kr

Abstract

As the advent of next-generation sequencing (NGS) technology, various *de novo* assembly algorithms based on the de Bruijn graph have been developed to construct chromosome-level sequences. However, numerous technical or computational challenges in *de novo* assembly still remain, although many bright ideas and heuristics have been suggested to tackle the challenges in both experimental and computational settings. In this review, we categorize *de novo* assemblers on the basis of the type of de Bruijn graphs (Hamiltonian and Eulerian) and discuss the challenges of *de novo* assembly for short NGS reads regarding computational complexity and assembly ambiguity. Then, we discuss how the limitations of the short reads can be overcome by using a single-molecule sequencing platform that generates long reads of up to several kilobases. In fact, the long read assembly has caused a paradigm shift in whole-genome assembly in terms of algorithms and supporting steps. We also summarize (i) hybrid assemblies using both short and long reads and (ii) overlap-based assemblies for long reads and discuss their challenges and future prospects. This review provides guidelines to determine the optimal approach for a given input data type, computational budget or genome.

Key words: de novo assembly algorithms; de Bruijn graph; next-generation sequencing; single-molecule sequencing.

Introduction

De novo whole-genome assembly is often described as a giant jigsaw puzzle with millions of pieces. Because a whole-genome sequence cannot be read at one glance with any state-of-the-art technology, many fragmented sequences must be assembled by finding shared regions of reads up to the chromosome level [1–3]. For instance, the human genome, which comprises approximately 3 billion base pairs (~3 Gb), requires billions of jigsaw puzzle pieces (sequences) of approximately 100–250 nt in length to successfully assembly a whole genome. If the size of a whole human genome is represented by the perimeter of the earth ($\sim 4 \times 10^7 m$), constructing a whole-genome assembly would be like drawing a world map by using only a 1 m ruler. Moreover, the topological complexity and nonrandomness of genome sequences cause other challenges in *de novo* assembly. Approximately 50% of the human genome comprises nonrandom repeat elements, such as long interspersed nuclear elements (LINEs), short interspersed nuclear elements (SINEs), long terminal repeats (LTRs) and simple tandem repeats (STRs) [4, 5], which often cause misarrangements or gaps in the assembly. These repeat sequences also cause a nonuniform read depth, thus resulting in copy loss or gain in the assembly.

As innovative next-generation sequencing (NGS) technologies are rapidly developed and advanced sequencing technologies are newly introduced, the sequencing of billions of reads can be accomplished in an increasingly time- and cost-effective manner. Although the earliest NGS technology, the 454 platform, which sequences relatively long reads (400–700 nt), has previously been popularly used in the *de novo* assembly of bacterial genomes [6], it is currently rarely used and has been replaced by new NGS platforms and single-molecule sequencing (SMS) platforms. The average cost for sequencing of a whole genome has dramatically decreased, owing to new NGS platforms, such as Illumina and Ion Torrent [7–13]. Despite the short length of NGS reads (typically 50–300 nt for Illumina, 100 or

Jang-il Sohn, a PhD candidate in Statistical Physics at Korea University, has studied and developed *de novo* whole-genome assembly algorithms as a researcher at Hanyang University since January 2015.
Jin-Wu Nam has been an assistant professor at Hanyang University since 2012. He received a PhD in Bioinformatics from Seoul National University and studied computational biology at the Whitehead Institute for Biomedical Research affiliated with MIT as a postdoctoral associate.
Submitted: 3 June 2016; **Received (in revised form):** 24 August 2016

200 nt for Ion Torrent), *de novo* assembly of a large genome is enabled by simultaneously overlapping billions of these short reads. Many algorithmic and heuristic methods have been proposed to assemble such a large number of short reads.

Recent large-scale genome studies, such as the 1000 Genome Project (http://www.1000genomes.org/) [14, 15], 10k UK Genome Project (http://www.uk10k.org/), International Cancer Genome Consortium (http://icgc.org/) and 1001 Arabidopsis Genome Project (http://1001genomes.org/), have successfully identified genomic differences among individuals or cells. From the results of these studies, it is now understood that the structural and single-nucleotide variations among individuals or cells are more abundant than previously anticipated [16]. The genome resequencing approach often fails to detect these highly variable genomic regions [17, 18]. Hence, the *de novo* assembly of individual genomes would be a better choice for building a precise map of highly rearranged genomes and for understanding the associated phenotypes.

This article reviews the basic computational approaches used for *de novo* assembly, classifies graph-based assembly algorithms that use short and/or long reads and compares them in terms of the computational cost and quality of assembly. This review also discusses the current computational challenges in *de novo* assembly of high-throughput short reads and possible solutions.

De novo short read assembly

Basic strategy of *de novo* short read assembly

The basic strategy for *de novo* assembly for short NGS reads comprises three steps: (i) contig assembly, (ii) scaffolding and (iii) gap filling ([2, 3, 19–22]; Figure 1). In the contig assembly step, the reads are assembled as long consensus sequences (called contigs) without gaps. Then, in the scaffolding step, the contigs are connected by large-insert (pair-end/mate-pair) reads, which generally originate from large DNA fragments or fosmid inserts of several kilobases in length. The ordered set of connected contigs is defined as a 'scaffold'. Once the contigs are scaffolded, spaces called 'gaps' remain between the contigs if there is no overlap between the contigs, and undefined bases and approximate distances are estimated from the insert size of the large-insert reads. The gaps are carefully filled by using other independent reads (gap-filling step) to complete the assembly. The scaffolding and gap-filling steps can be performed iteratively to enhance the quality of the assembly until no contigs are scaffolded or no additional gaps are resolved [23–27].

The most popular approach for the short read assembly, the de Bruijn graph [1, 3, 22], has been largely applied to many different assemblers. The popular assemblers for short NGS reads are summarized in BOX A and Table 1. Generally, the de Bruijn graph approach has been used for the assembly of short reads by converting them to *k*-mers, whereas the overlap-layout consensus (OLC) approach has been used for the assembly of long reads. In this section, we will mainly focus on the de Bruijn graph approach.

The de Bruijn graph approach

The graph using the *k*-mers, known as the de Bruijn graph, can be simplified and significantly reduces the search time for the optimal path [1–3, 39]. The de Bruijn graph refers to a directed graph that represents overlaps between sequences with equal

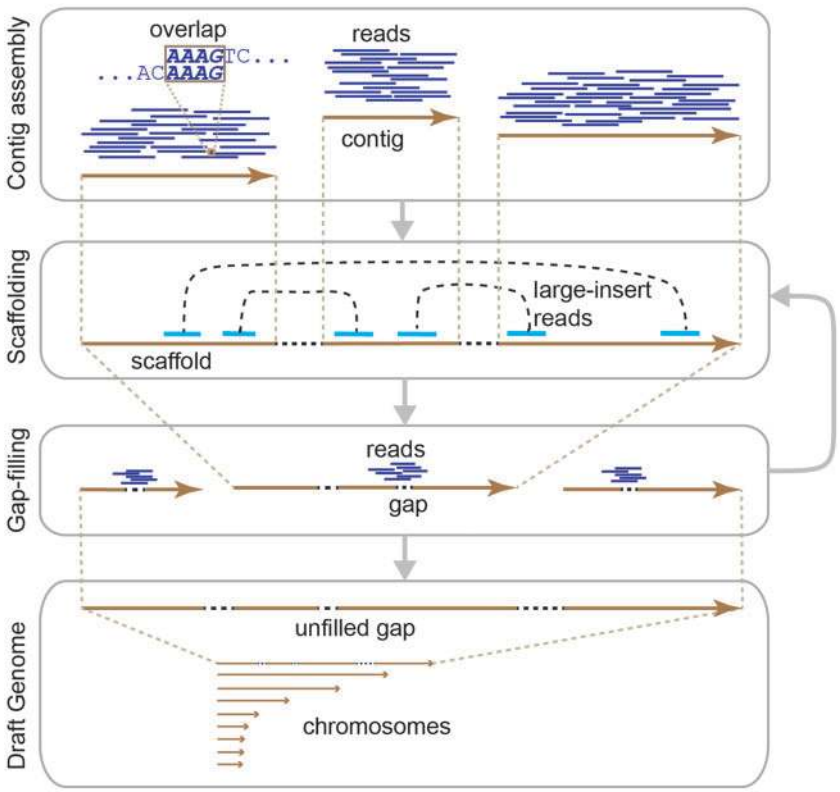

Figure 1. General workflow of the *de novo* assembly of a whole genome. By overlapping reads, contigs are assembled from short reads before scaffolding by large-insert reads, and the remaining gaps are filled. The scaffolding and gap-filling steps can be iteratively performed until no contigs are scaffolded or no additional gaps are resolved before completion. Through this procedure, a draft genome consisting of chromosomes is built. Some unfilled gaps may remain in the draft genome.

or similar in degrees and out degrees at each node, in which the overlapped sequences are represented with a *k*-mer [2, 39]. All reads are split into $(L - k + 1)$ *k*-mers, where *L* is the read length, and *k* is the size of the *k*-mer. The *k*-mers are connected by overlapping prefix and suffix (*k*-1)-mers (Figure 2). Once the de Bruijn graph is constructed, the optimal path is identified in the graph [2]. The contigs or scaffolds are obtained by inversely transforming the optimal path in the de Bruijn graph into sequences.

The de Bruijn graph is classified into two types, Hamiltonian and Eulerian de Bruijn graphs, according to the method of expressing the nodes and edges ([2, 20]; Figure 2). In the Hamiltonian de Bruijn graph, the *k*-mer itself becomes a node, and the (*k*-1)-mer suffix of the *k*-mer that overlapped with the

Box A. Short Read Assemblers

ALLPATHS-LG

ALLPATHS-LG [29] is based on the Eulerian de Bruijn graph, and is considered to be the most accurate assembler for short NGS reads. Different from other Hamiltonian graph-based assemblers, ALLPATHS-LG does not remove erroneous or redundant paths except certain ones until the last step because they may be the right paths. This requires relatively large memory and may take several weeks for large vertebrate genomes. Although ALLPATHS-LG was originally designed for only Illumina short reads, it is possible to use for PacBio long reads in small bacterial genomes [136].

SOAPdenovo

SOAPdenovo2 [25] implements an algorithm based on the sparse *k*-mer and reduces the required memory by up to 35 GB for assembly of a human whole genome. Its assembly quality is comparable with that of ALLPATHS-LG (Tables 1 and 2). The strong point of this assembler is that each step of *de novo* assembly (e.g. error correction, assembly of contigs or scaffolds, gap filling) can be performed separately and simply as circumstances demand.

SparseAssembler

SparseAssembler [34] uses the sparse *k*-mer and drastically reduces the required memory by an order of magnitude. For the assembly of human chromosome 14, the SparseAssembler only used 3 GB of memory for $k = 54$, while other programs that use the ordinary *k*-mer strategy required up to 30–50 GB of memory (37 GB for Velvet, 49 GB for ABySS and 30 GB for SOAPdenovo) [34].

SGA

SGA [32, 40] implements assembly algorithms using the FM-index, a compressed substring index based on the Burrows–Wheeler transform that reduces a large RAM memory. Basically, SGA is a one of the overlap-based assemblers [3], and can be classified as a *k*-mer based assembler. SGA searches overlaps between reads implementing *k*-mer methods, and assembles reads directly, instead of assemble *k*-mers. A recent version of SGA, implementing Bloom filter [32, 35, 62, 80, 81] to reduce running time, assembled the whole human genome with 56 GB of memory in 24 h on a single hexa-core XEON X5650 (2.66 GHz) machine. In terms of computational cost, the SGA shows the best performance (Table 3).

MaSuRCA

For assembly of the giant genome of the Loblolly pine (~22 Gb), MaSuRCA [48] (hybrid method of OLC and Eulerian de Bruijn graph) [48] compressed overlapping reads into super-reads of 3–13 kb, and the depth of the super-reads was reduced to 2–3X. The preassembled super-reads are assembly with OLC assembler, e.g. Celera assembler [70]. Using this approach, Zimin *et al.* [37] assembled the 'giant genome' of 22 Gb using 1 TB memory in 3 months.

Meraculous

Meraculous [28] builds a lightweight hash table, in which only high-quality extensions are stored. Owing to the high-quality extensions, Meraculous does not require any explicit error correction step, and produces more accurate results (Figure 8). Further, the required RAM memory size is reduced to <10 GB for whole human genome assembly, enabling parallel computing in distributed clustering systems with low memory per node.

JR-Assembler

The JR-Assembler [33] extends seed reads by overlapping other reads instead of using the de Bruijn graph, and is similar to greedy assemblers. However, they overcame the limitation of greedy assembler by introducing jumping process and misassembly/error-detecting methods (remapping/back-trimming). The N50 lengths of contigs or scaffolds are comparable with ALLPATHS-LG. Although the extension-based method could reduce the required memory size, >400 GB of memory is required for whole human genome assembly [33] (Table 3).

Velvet

Velvet [42] is one of the Eulerian de Bruijn graph assembler. Unlike other assemblers, Velvet uses bidirectional de Bruijn graph. The potential assembly errors are corrected by so-called 'tour-bus' algorithms in assembly graph level.

SPAdes

SPAdes [47] is one of Eulerian de Bruijn graph assemblers, and was designed for single-cell sequencing. This program uses paired de Bruijn graph [153], which is a kind of doubled-layered de Bruijn graph. The *k*-mers from DNA fragment reads build the inner de Bruijn graph, which is used for contig assembly. On the other hand, the 'paired *k*-mers' with large insert size build the outer de Bruijn graph, which is used for repeat resolving or scaffolding.

ABySS

ABySS [41] is computationally efficient. The main feature of ABySS is the distributed *k*-mer hash table. To reduce RAM memory requirement for a computer, ABySS distributes the *k*-mer hash table over clustering systems, in which the computers with small RAM memory are connected to each other by network.

Table 1. Summary of short read assemblers

Assembler	Speed[a]	Memory efficiency[a]	N50 length[b]	Input data type	Assembly steps
Celera	+	+	+++	S,P,Li,L	C,S,G
ALLPATHS-LG	+	+	+++	P,Li (L[c])	E,C,S,G
ABySS	++	+++	++	S,P,Li	E,C,S
Velvet	++	++	+	S,P,Li	C,S
SPAdes	++	+++	++	P,Li	E,C,S
SOAPdenovo	+++	++	++	S,P,Li	C,S,G
SparseAssembler	++	+++	++	S,P,Li	C,S
SGA	+++	++	+	S,P,Li	E,C,S
MaSuRCA	+	+	+++	S,P,Li,L	C,S,G
Meraculous	++	++	++	P,Li	C,S,G
JR-Assembler	+	+	+++	S,P,Li	E,C,S,G

Note: +++: high; ++: medium; +:low.
In the 'Data Type' column, the symbols S, P, M and L refer to Single-end reads, Paired-end reads, Large-insert reads and Long reads, respectively.
In the 'Assembly steps' column, the symbols E, C, S and G refer to Error-correction, Contig assembly, Scaffolding and Gap-filling steps, respectively.
[a]Reference [25, 28, 29, 31, 32, 33, 34, 36, 37, 38].
[b]Reference [17, 30, 31, 33, 34, 35, 37, 38].
[c]PacBio long reads can be used only for small genome for ALLPATHS-LG.

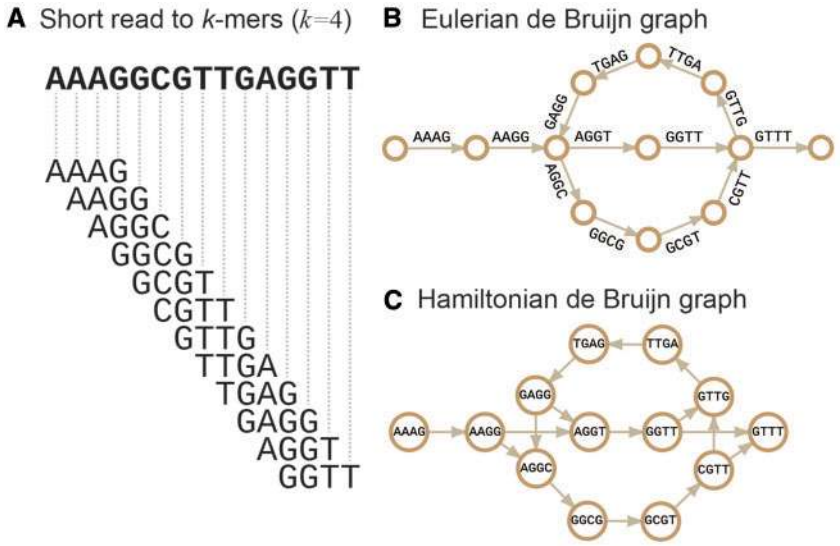

Figure 2. (**A**) In the de Bruijn graph approach, short reads are split into short *k*-mers before the de Bruijn graphs are built. (**B**) In the Hamiltonian approach, the *k*-mers (or sequences) are the nodes, whereas they are the edges in the Eulerian approach. The *k*-mers are connected to neighbors by overlapping prefix and suffix (*k*-1)-mers.

(*k*-1)-mer prefix of the next *k*-mer becomes an edge. In other words, if the prefix of a node is the same (or overlaps) as the suffix of another node, the two nodes are connected. In contrast, the Eulerian de Bruijn graph presents nodes and edges in the opposite manner: the sequence of the *k*-mer is an edge and the overlapped (*k*-1)-mer is a node.

The Hamiltonian de Bruijn graph approach is commonly applied in *de novo* assemblers such as SOAPdenovo [25], SGA [40], ABySS [41], Meraculous [28] and Velvet [42]. The Hamiltonian graph approach is similar to the OLC approach in that the node is the sequence and the edge is the overlap. In these approaches, the sequences are assembled by finding Hamiltonian paths that traverse all nodes, each of which is visited only once; this scenario is known as the nondeterministic polynomial time (NP)-complete problem when the number of nodes is not trivial [2, 39]. Thus, as the size (or topological complexity) of the genome increases, the computation time required to solve the graph problem diverges indefinitely (Figure 3A) because the scaffolding or repeat-resolving steps become too complicated. Normally, the computational complexity to find the Hamiltonian paths is $O(m \times 2^n)$, where m is the total number of nodes, and n is the number of branching nodes [43, 44]. The branching nodes refer to the nodes with ≥ 2 in or out degrees. As seen in Figure 3B, we have shown that the n is correlated to the contig length L. Therefore, the complex graph is often simplified when a large genome (e.g. a mammalian genome) is assembled. For instance, SOAPdenovo [45] removes all branching nodes to reduce the computational complexity, approaching $O(m)$. However, the debranching may produce many short contigs. Thus, its next version, SOAPdenovo2 partially resolves the branches caused by repeats, rather than removing all of them.

In contrast, the Eulerian de Bruijn graph approach is able to solve the complicated graph problem by finding the Eulerian paths that traverse all edges, each of which is visited only once without simplification in polynomial time, $O(n^2)$ [2, 39, 46].

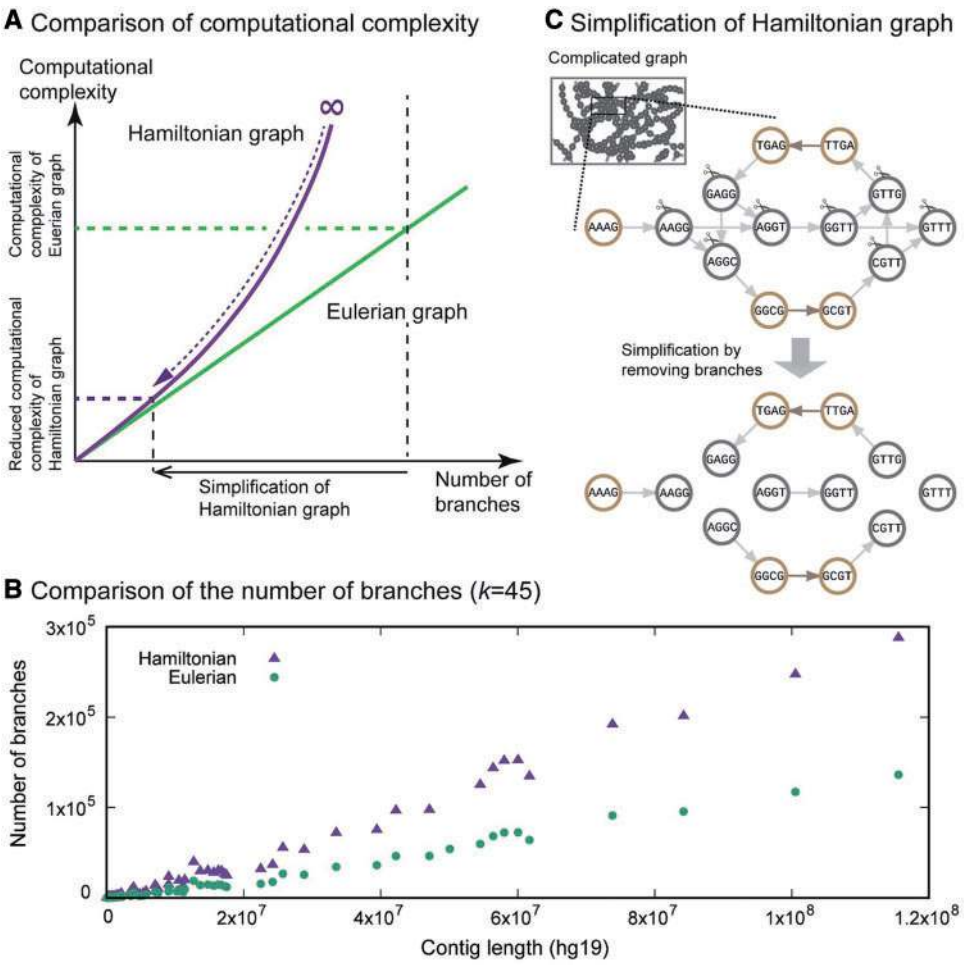

Figure 3. (**A**) The function $f(x)$ refers to the computational complexity, where x indicates the number of branches of the de Bruijn graph. If x is sufficiently large, the Hamiltonian approach leads to the NP problem, $f_{NP}(x) \sim e^{ax}$, and takes too long to complete the task. As the number of branches decreases (the arrow below the x-axis), the computational complexity of the Hamiltonian de Bruijn graph is drastically reduced. In contrast, there is no need to reduce the complexity in the Eulerian approach because it leads to the polynomial problem, $f_P(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n \rightarrow f(x) \sim x^n$, thus indicating that it can be completed in a finite time. (**B**) The number of branches is proportional to the length of contigs. The plots are obtained by analyzing the human reference genome (GRCh37/hg19). Each k-mer graph (45-mer) was constructed from each contig of human reference genome. Each dot indicates the number of the branches in a graph. The number of branches in the Hamiltonian de Bruijn graph is as much as 2-fold greater than the number of branches in the Eulerian de Bruijn graph. (**C**) Because the Hamiltonian de Bruijn graph is normally difficult to solve, simplification processes are necessary. For instance, the branches are removed in SOAPdenovo2; thus, the running time is drastically reduced in the contig assembly steps, and the branches are resolved in the scaffolding step.

Eulerian de Bruijn graph-based assemblers generally perform better in the assembly of a large genome than the Hamiltonian de Bruijn graph approach in terms of the assembly results. In fact, EULER [39], SPAdes [47], ALLPATHS-LG [29] and MaSuRCA [48], which implement a Eulerian de Bruijn graph method, have shown better performance in recent comparisons of assemblers [30, 31, 49].

Challenges in *de novo* short read assembly

Computational challenges in *de novo* assembly

De novo assembly is a nontrivial problem that requires expensive computations and a high-quality standard, and should be able to overcome many computational challenges within the workflow. In this section, we discuss four major computational challenges of *de novo* assembly and solutions to overcome the challenges. The strategies used to address the challenges for different assemblers are summarized in Table 2.

The first challenge is the correction of sequencing errors, which must be performed before or during assembly because the errors might impede the extension of contigs or scaffolds and introduce artifacts. The rate and types of sequencing errors vary according to the NGS platform and library preparation method, which should be taken into account when correcting the sequencing errors. For example, Illumina platforms generally produce short reads with $\leq$1% random sequencing errors, and the errors tend to be accumulated in the 3′ part of reads [12,50–52].

The second challenge is uneven read depth, which results from polymerase chain reaction (PCR), cloning, extreme GC bias, sequencing errors and copy number variations [53–59]. Uneven depth often causes breaks in the assembly, resulting in the introduction of gaps. The use of an optimal k might be needed to resolve this problem.

The topological complexity of repetitive elements in genomes presents a challenge. If the reads are long enough to cover the repetitive elements, this challenge can be resolved.

Table 2. Strategies for challenges

Challenges	Strategies	Assemblers
Assembly approach	Eulerian de Bruijn graph	ALLPATHS-LG, Velvet, MaSuRCA
	Hamiltonian de Bruijn graph	ABySS, SGA, SOAPdenovo, SparseAssembler, Meraculous
	String graph	SGA
	OLC	MaSuRCA
Sequencing error	Searching by *k*-mer counting	ALLPATHS-LG, SOAPdenovo, SGA, Meraculous
	Correction by other reads	ALLPATHS-LG, SGA
	Excluding erroneous *k*-mer (or reads)	ALLPATHS-LG, Meraculous, JR-assembler
	Correction by suffix tree	SOAPdenovo
Complexity reducing	Removing tips or bubbles	ABySS, Velvet, ALLPATHS-LG, SOAPdenovo, SparseAssembler, SGA
	Removing erroneous branches	AByss, SGA, JR-assembler, Velvet, SOAPdenvo, SparseAssembler
	Removing redundant connections	SparseAssembler, SGA
	Sparse *k*-mer	SOAPdenvo, SparseAssembler
	High quality *k*-mer table	ALLPATHS-LG, Meraculous
	Large *k*-mer size	ALLPATHS-LG
	Maximum likelihood	ABySS
Repeat resolving	Pulling repeats using paired reads	ALLPATHS-LG
	Identifying repeats by read depth	ALLPATHS-LG, SGA, MaSuRCA
	Breaking repeats	SOAPdenovo, Meraculous, JR-assembler
Uneven depth	Multiple *k*-mer	ALLPATHS-LG, SOAPdenovo
	Direct assembly of reads	SGA, JR-assembler
RAM memory	Distributed memory on linux cluster	ABySS, Meraculous
	2 bits nucleotides	SparseAssembler
	FM-index	SGA
	Bloom filter	SGA
	Reducing depth by super-reads	MaSuRCA
	Lightweight hash	ABySS, Meraculous

However, because Illumina sequences are generally short, and a sequence normally does not cover a repetitive region, long reads are preferred to address this problem.

The fourth challenge is algorithmic complexity that requires a high computation cost. *De novo* whole-genome assembly based on the de Bruijn graph requires substantial random-access memory (RAM), storage and long computation times. Although the *de novo* assembly of small genomes, such as bacterial genomes, takes only several minutes, the assembly of large genomes, such as mammalian genomes, typically takes from several days to weeks and requires over tens to hundreds of gigabytes (GB) of peak RAM memory, depending on the genome size and algorithms.

Sequencing error correction

Although the Illumina platform produces highly accurate reads [12, 50–52], the reads may also lead to misassembly. The sequencing errors occur more frequently in regions with an extremely high GC or AT content, such as constant heterochromatin regions, including centromeres, telomeres or highly repetitive sequences, all of which may generate a complex assembly graph. Therefore, the sequencing errors should be corrected for more accurate and contiguous *de novo* assembly before or during assembly.

The essential issue in correcting sequencing errors is how to identify sequencing errors and distinguish them from the heterozygous alleles. The error correction methods are categorized into (i) *k*-mer counting, (ii) suffix tree- or array-based methods, (iii) multiple sequence alignment-based methods and other methods, including hybrid error correction methods [52, 60, 61] (we recommend that these references be reviewed for more details about sequencing error correction).

Most of the sequencing error correction tools implement the *k*-mer counting methods, and even tools in other categories often use *k*-mer counting to detect sequencing errors. In *k*-mer counting methods, low-depth *k*-mers are assumed to be erroneous. In suffix tree-based methods, the erroneous *k*-mers are detected by low-frequency branches in the *k*-mer suffix tree [52, 60]. In contrast to these two methods, the multiple sequence alignment-based method is more intuitive: the sequencing errors are detected by directly aligning reads with each other and are corrected by consensus. Although the multiple sequence alignment-based method may be computationally expensive, this method is commonly used for error correction of long SMS reads. In the next paragraph, the *k*-mer counting methods for short NGS reads are discussed in detail.

If a genome is amplified through an ideal amplification processes, the fragments of the reads are evenly distributed over all regions, and the histogram of *k*-mer depth forms a normal distribution, e.g. a Poisson (if the coverage of the reads is rather low) or Gaussian (high coverage) distribution [62, 63]. However, if sequencing errors occur, the corresponding *k*-mer depth shows an exponentially decreasing curve (Figure 4). In that case, low-depth *k*-mers below the threshold can be simply excluded, or the reads including the sequencing errors can be corrected through multiple sequence alignment-based methods or other methods before or during contig assembly. For instance, SGA contains a precise sequencing error correction module in which erroneous reads are detected by *k*-mer counting, and the reads are corrected by a multiple sequence alignment-based method [32, 40].

However, in these error detection methods, the error-free *k*-mers below the threshold are treated as erroneous *k*-mers, and some erroneous *k*-mers remain, thus potentially leading to

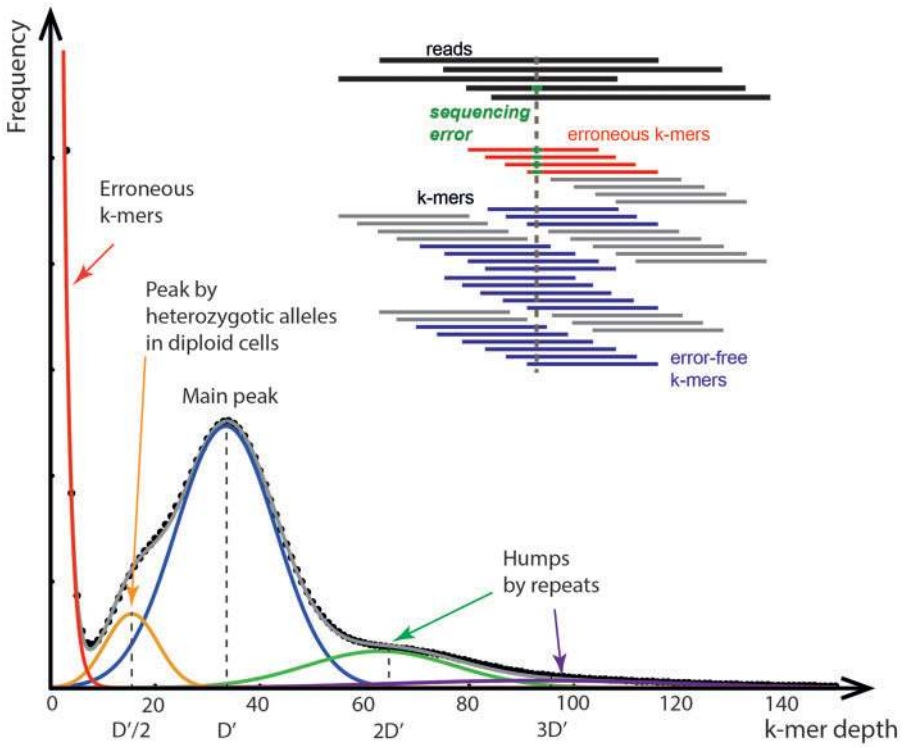

Figure 4. *K*-mer histogram. The x-axis refers to the *k*-mer depth $D(k)$, which indicates '*k*-multiplet'; the y-axis refers to the frequency of the *k*-multiplet, $f(D(k))$ [64]. For example, if a set of *k*-mers is given by K = {ATT, ATA, GTG, GCA, GCA, CAT, CAT, TAT, TAT, TAT, TAT}, the frequency is calculated as $f(1) = 3$ because there are three unique *k*-mers, {ATT}, {ATA} and {GTG}; $f(2) = 4$ because there are two twins, {GCA, GCA} and {CAT, CAT}; $f(3) = 0$ because there is no triplet; $f(4) = 4$ because there is one quadruplet, {TAT, TAT, TAT, TAT}. The curve of the *k*-mer histogram shows a normal distribution in ideal cases, provided that the depth of the read is sufficient. If there are sequencing errors in the reads, an exponentially decreasing curve is produced. The humps beyond the normal distribution peak are generated due to the repetitive structures and copy-gained regions. In the plot, a small peak resulting from heterozygous alleles appears below the main peak. The black dots are obtained by using ERR244145, and the exponential (erroneous; red) and Gaussian (error-free; orange, blue, green and purple) functions are ideal case curves. The gray line (sum of ideal cases) is similar to the real data (black dots).

misassembly. In particular, the *k*-mers resulting from heterozygosity should be more carefully managed because these *k*-mers are similar to the erroneous *k*-mers [65–67]; their peaks appear at D'/n, where D' is the main peak of the histogram, and n is ploidy. In the case of diploidy, the peak is located at $D'/2$ (Figure 4). Beyond heterozygosity, the reads in repetitive regions are also problematic. Because of the abnormally high frequency of *k*-mers in the repetitive regions, the sequencing errors may not be detected. These problems resulting from repeats or heterozygosity can be overcome with multiple sequence alignment-based error correction methods [52].

In some cases, the sequencing errors are discounted in the step of building the hash table of *k*-mers before the assembly step (ALLPATHS-LG [29] and Meraculous [28]) or are excluded by back trimming during extensions in the assembly steps (JR-Assembler [33]).

The *k*-mer counting method is also popularly used to estimate genome size. For contextual reasons, the genome size estimation is separately explained in BOX B.

Box B. Genome Size Estimation

The *k*-mer counting methods can be used also for estimating genome size using the depth and length of reads and the depth and size of the *k*-mer [154, 155], as follows:

$$D = \frac{D'l}{l-k+1},$$

and

$$G = \frac{N_{base}}{D} = \frac{N_{read}(l-k+1)}{D'},$$

where D and G are an estimated depth of reads and the genome size, respectively; l is the average read length; k is the *k*-mer size; D' is the *k*-mer depth at the peak of the *k*-mer histogram; $N_{k-mer} = (l-k+1)$ is the number of *k*-mers in a read; and N_{base} is the number of sequenced bases, where $N_{read} = N_{base}/l$ is the number of reads. This method is useful because genome size can be estimated before or without whole-genome *de novo* assembly.

Repetitive structures

Finding the shortest complete path, which is defined as the shortest path (π) going through all nodes, Vs, in a complex *k*-mer graph does not resolve the genome assembly problem, mainly because of the nonrandomness of genome sequences containing many repetitive structures, such as STRs, LINEs, SINEs or the highly repetitive sequences (near centromeres, telomeres or satellites on chromosomes). If a genome sequence is perfectly random, the *de novo* whole-genome assembly might be created by finding a super sequence containing all the subsequences. However, the real assembly problem is not the super

sequence problem because of the repeats, which sometimes form complicated repetitive structures. The highly repetitive sequences often generate numerous ambiguous paths and gaps in the resulting assembly [4, 5, 17, 68, 69]. Some of the repeat problems in the assembly can be resolved by the read depth or the information from large-insert or long reads [68]. In this section, we mainly discuss the methods that use the read depth and the information from paired reads with a large insert.

The statistics of the read depth (for instance, A-statistics [70]) are often used to resolve repetitive structures. Because the read depth in repeat regions is higher than that in other regions, the copy number of the region can be estimated (Figure 5A). However, this approach is not sufficient to resolve the repeat structures because the read depth can be altered by many other factors, thus resulting in a false estimation of the copy number.

A complementary method for resolving the repetitive structures is to use the insert size of reads, wherein each fragment matches a region outside of the repetitive regions (Figure 5B). If multiple paths exist in the graph because of the repetitive structures, these can be resolved by using the estimated insert size (Figure 5C). However, large-insert reads should be used carefully because constructing the assembly graph can be complicated by the large inner distance. As shown in Figure 5D, if the paired-end reads with a large insert size are applied first (long-first approach), long-range repeats (*g* in the graph) are resolved, but repetitive structures within a shorter range than the insert size remain (*f* in the graph). As a result, the complexity of the graph is not reduced, but can be more complex, depending on the case. The reads of shorter insert sizes must be applied first (short-first approach) and, in turn, longer insert sizes are applied in the last step to properly resolve the repetitive structures. By resolving the structures in this order, the repetitive structures produce a less complex graph (Figure 5D).

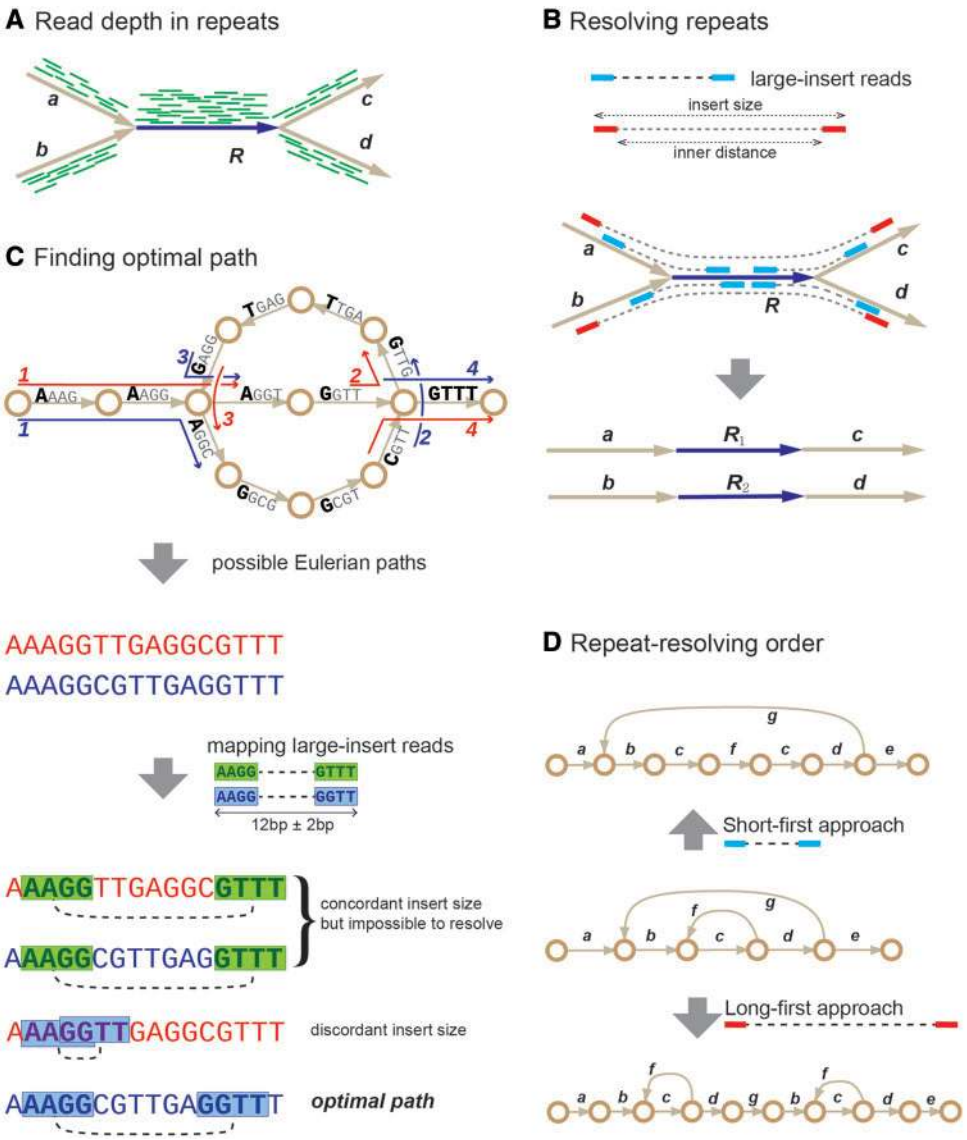

Figure 5. (**A**) The read depth in repetitive region is greater than those of other regions (**B**) The repeat *R* can be resolved, if the large-insert reads are supporting *R* and the flanks *a*, *b*, *c* and *d*. (**C**) In this figure, there are two Eulerian paths. Of the paths, the latter path, which matches the insert size of the large-insert read, is optimal. (**D**) The large-insert reads of the shortest insert size should be applied first to properly resolve the repeats. If the longest insert size is applied first, inadvertent additional repeats are generated, thus complicating the repeat-resolving processes.

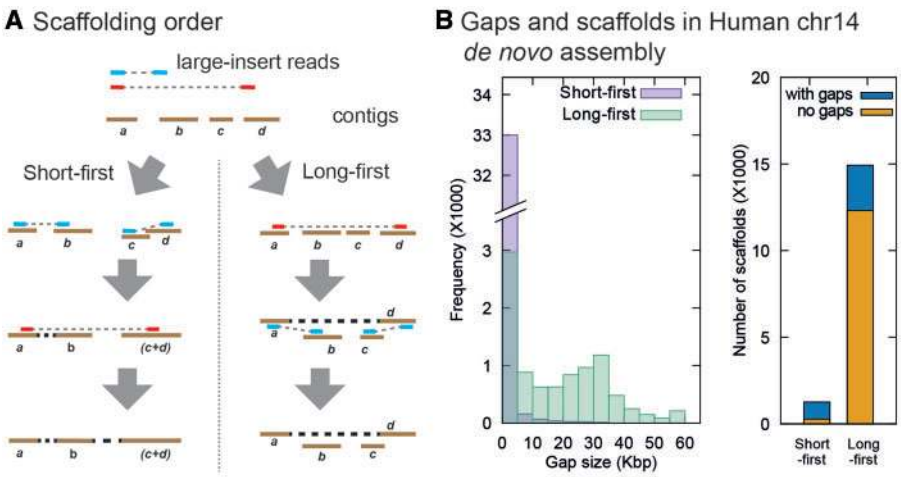

Figure 6. (**A**) In scaffolding, the reads of shortest insert size should be applied first. If the longest reads are applied first, advertent large gaps and unscaffolded contigs are generated. (**B**) We performed *de novo* assembly of human chromosome 14 with a Minia assembler [35] and OPERA scaffolder [73, 74] using the GAGE-A data set [49]. The left plots show the gap-size distribution in the scaffolds. If the large-insert reads with longer insert sizes are applied first (long-first scaffolding), numerous large gaps are generated, whereas a few large gaps are generated in short-first scaffolding. The right plots show the number of scaffolds with or without gaps. Most of the no-gap scaffolds are unscaffolded contigs in the long-first approach.

The complexity of scaffolding

Scaffolding based on paired-end or mate-pair reads can also help to overcome repeat problems [71, 72]. Recently, a study by Gao *et al.* [73] has shown that the number of misassemblies can be reduced by considering all large-insert read libraries at the same time during a scaffolding step. However, when using a long-first approach, large gaps and unscaffolded contigs can be inadvertently generated (Figure 6).

The complexity of scaffolding can be considered in two different levels, mapping and algorithms. If read pairs map to repetitive elements, they could generate a wrong link between two distant contigs. These false links should be removed to reduce the complexity of assembly graph. In the ideal case with neither sequencing error nor misassembly, contigs may be properly scaffolded by mate-pair reads as shown in Figure 7A. However, even in the ideal case, numerous misassemblies are possibly generated because of repeats or mapping errors (Figure 7B), if the information of read pairs are not sufficient, or if the algorithms consider only local information. The mis-scaffolding problem can be partly resolved by using long-spanning sequences (Figure 7C).

Highly repetitive regions are more problematic, which normally form star-like structures in assembly graphs (Figure 7D and E). In this case, it is extremely difficult to find an optimal path because of a high complexity. As a result, the highly repetitive regions are often remained as gaps during assembly.

Uneven read depth

The limitations of short NGS reads, such as GC biases and PCR biases, cause the problem of uneven sequencing depth, which generates gaps in a draft genome. This problem has been poorly resolved because of the absence of data. It could be addressed by sequencing a greater depth of reads, but doing so would greatly increase the computing time and cost. Multiple *k*-mer approaches have been used to at least partially address this problem [25, 29]. Short *k*-mers are used to connect reads in regions with a shallow sequencing depth, whereas long *k*-mers are used for the reads in other regions. This multiple *k*-mer approach is implemented in several short read assemblers, such as SOAPdenovo2 [25] and ALLPATHS-LG [29]. To our knowledge, there is no other algorithmic strategy for addressing the uneven sequencing depth, except the multiple *k*-mer approaches. However, the use of the small *k*-mers could potentially cause unintended misassembly. The low sequencing depth problem can be partially addressed in the gap closing step.

Computational cost

RAM memory

Because the *k*-mer approach generates $(l-k+1)$ *k*-mers from a read of length *l*, large RAM memory is required to assemble a large genome. Approximately $2(k+1)G$ bytes RAM memory is required to store *k*-mer table of a genome of size *G* [28]. For instance, approximately 0.5 terabytes (TB) of memory is required to build a hash table of 75-mers for the human genome (3 Gb). Indeed, ALLPATHS-LG (default *k*-mer size is 96) requires at least 512 GB of memory for a human whole-genome assembly [29]. Optimal data structures or complexity-reducing algorithms [78], such as sparse *k*-mer [25, 34], FM-index [40, 79], Bloom filter [32, 35, 62,80, 81], light-weight hash table [28] or the super-read approach [48], are benefited to assemble such large genomes with a tractable RAM memory. In particular, the super-read approach, which is a type of synthetic long read of short NGS reads, has been used to reduce the required RAM memory for the assembly of the large genome of the loblolly pine (~22 Gb) [48].

Computing time

It is known that the Hamiltonian approach (known as a NP-problem) is much slower than the Eulerian approach (known as a P-problem) in a large genome, which is represented as a high-complex *k*-mer graph with many nodes and edges (Figure 3A). Paradoxically, most of the assemblers that implement the Hamiltonian de Bruijn graph, including SOAPdenovo and SGA, are much faster than the assemblers that implement the Eulerian de Bruijn graph, such as ALLPATHS-LG (Table 3).

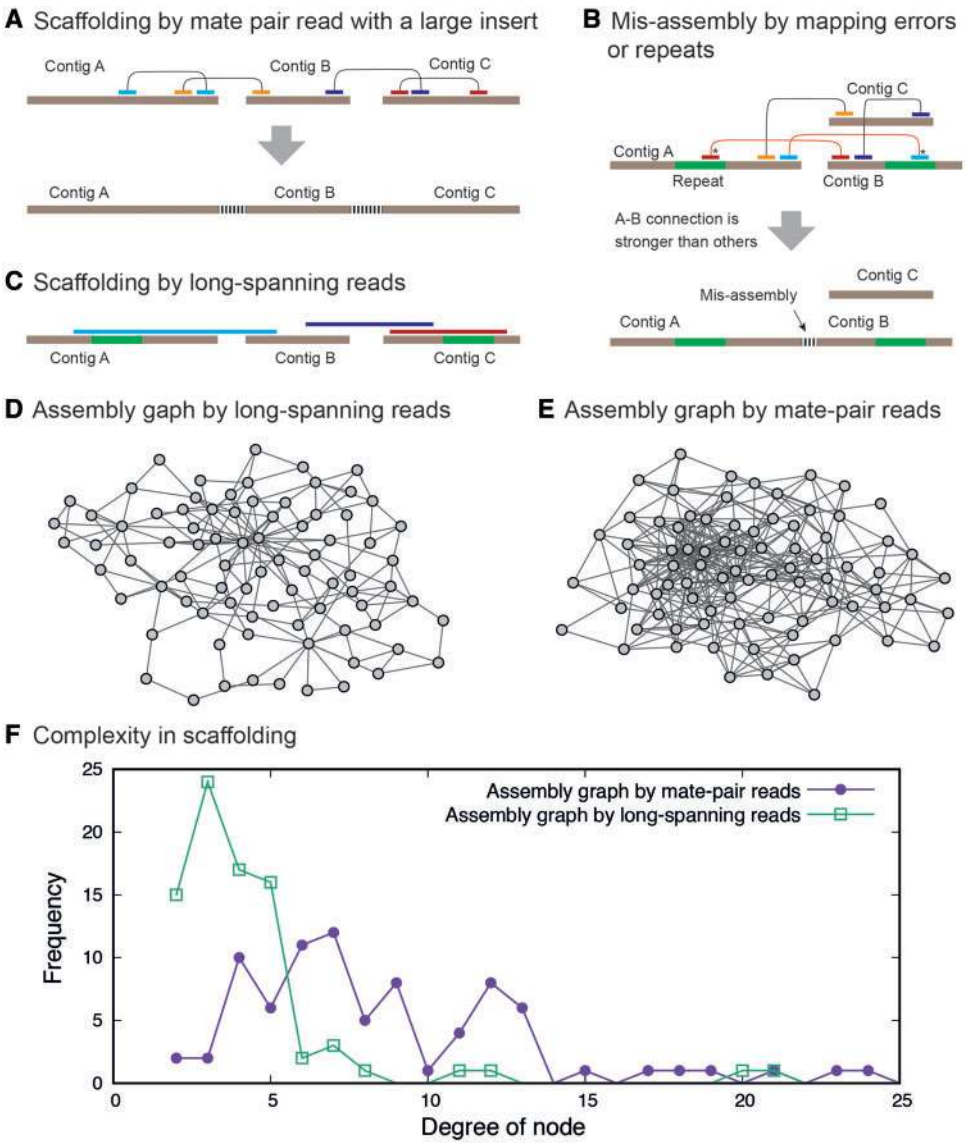

Figure 7. (**A**) Solving repetitive structures during scaffolding by large-insert reads. Each large-insert reads are mapped to the corresponding contigs. The dotted line indicates gaps. (**B**) Possible misassembly by erroneous mapping (indicating as *) of the mate-pair reads. The repetitive regions in the contig are indicated as a green line. (**C**) Solving repetitive structure during scaffolding by long-spanning reads indicating as red, blue and dark-blue lines. (**D–E**) Shown are the simulated assembly graphs, generated from *Escherichia coli* K-12 genome with simulated paired end with an insert size 200 bp (30X), mate-pair reads with an insert size 2500 bp (20X) and PacBio long reads with mean length 5000 bp (20X) using ART [75] and PBSIM [76]. We have assembled the paired-end reads to contigs using SOAPdenovo2, and aligned mate-pair reads and PacBio long reads to the contigs (>1000 bp) using BWA-MEM [77], respectively. From the mapping results, we have built two assembly (scaffolding) graphs (**D** and **E**), nodes and edges of which represent contigs and reads, respectively. We have set two contigs as connected if they are connected by at least three mate-pair or PacBio long reads. There are more hub (star-like) nodes in the mate-pair graph (**E**) than the PacBio graph (**D**). (**F**) The plots are obtained by counting nodes across their different degrees for the assembly graphs by mate pair (purple) and PacBio long reads (green), where the degree of node refers to the number of neighbor nodes.

In fact, ALLPATHS-LG required approximately 3 weeks with 48 processors and approximately 0.5 TB of memory for the *de novo* human whole-genome assembly, whereas the SOAPdenovo2 required only 3 days with eight Quad Core AMD processors (2.3 Gz) and 35 GB of memory [25], and SGA required only 1 day with a single hexa-core XEON processors and 56 GB of memory [32].

The complexity of the *k*-mer graph is the key to fast assemblers. Theoretically, the Hamiltonian de Bruijn graph approach could be faster than the Eulerian de Bruijn graph in a low-complexity graph (Figure 3A). As removing branches, the computational complexity of the Hamiltonian assemblers exponentially decreases for the NP curve (Hamiltonian de Bruijn graph), whereas the computational complexity of Eulerian assemblers linearly decreases for the P curve (Figure 3A). However, the accuracy (or continuity) represented by the N50 statistics is unavoidably reduced when such paths are removed, thus suggesting that there is a trade-off between accuracy and computational cost. In fact, both SOAPdenovo2 and SGA (Hamiltonian) use filtering steps to exclude ambiguous or redundant paths, thus leading to a short computational running time. In the case of Meraculous (Hamiltonian), the complexity of de Bruijn graph is reduced when building the hash table by excluding low-quality extensions [28]. In contrast, the ALLPATHS-LG (Eulerian) retains all possible paths up to the last step to enhance sensitivity [29, 82].

Table 3. Comparison of computational costs of short read assemblers

Assemblers	Algorithm type	**Genome**[a]	Sequencing depth	**Peak memory**[b] **(in GB)**	**Relative speed**[c]	**Relative computational cost**[d]	**References**[e]
ALLPATHS-LG	Eulerian	Human	∼100×	>512	1	597	[29]
SOAPdenovo2	Hamiltonian	Human	∼50×	35	10.8	4	[25]
ABySS	Hamiltonian	Human	∼35×	<16	11.9	<8	[41]
SGA[f]	Hamiltonian	Human	∼100×	56	65	1	[32]
JR-Assembler	Greedy like	Human	∼130×	418	1.8	279	[33]
MaSuRCA	Eulerian+OLC	Loblolly Pine	∼80×	800	1.4	644	[37, 38]

[a]Estimated genome size: Human, 3.1 Gb; Loblolly Pine, 22 Gb; bird (parrot), 1.2 Gb.
[b]RAM memory of ALLPATHS-LG is a minimum requirement, while the others are peak.
[c]Relative speed is estimated by (genome size)/(CPU time), and normalized by ALLPATHS-LG, where the CPU time is approximately (running time)/(number of threads). Note that the number of threads of Intel processors is twice the number of cores, whereas the number of threads of AMD processors is equal to the number of cores.
[d]Relative computational cost, (peak memory)/(relative speed), was normalized to SGA.
[e]ALLPATHS-LG took 3.5 weeks with Dell R815, 48 processors [29]; SOAPdenovo2 took 81 h with eight Quad-core AMD (2.3 GHz) [25]; SGA took 24 h with single Hexa-core XEON X5650 (2.66 GHz) [32]; JR-Assembler took 1.5 CPU weeks with four Octa-core Xeon E7-4820 (2 GHz) [33]; MaSuRCA took about 3 months with 64-core IA64 computer [37, 38].
[f]The relative speed of SGA was estimated with the time to contig assembly.

Quality evaluation and polishing of the *de novo* assembly

Normally, N50 is one of most popular metrics, considered for the assessment of *de novo* assembly. N50 is a median length of a set of contigs (or scaffolds). However, this metric is not sufficient to evaluate the overall quality of the *de novo* assembly. The metric does not validate information about misassembly and the actual genome size. For these reasons, the ordinary N50 metric is often used in modified forms, such as NG50, NA50 or NGA50. Notably, these metrics are usable only if a reference genome exists. Although the contig (or scaffold) N50 is calculated by setting the total size of contigs (or scaffolds), NG50 is an N50 normalized to the size of the reference genome. NA50, which was introduced in QUAST [83], indirectly considers the misassembly of the draft genome. The draft genome is first aligned to the reference genome, and large indels >1 kilobase pairs (kb) are assumed to be misassemblies, generating breaks at the positions. The NA50 is calculated by using the broken contigs.

Although nucleotide errors or single-nucleotide polymorphisms can be detected and polished by many tools, such as MAQ [84], SAMtools [85], SOAPsnp [86, 87], SNVmix [88], GATK [89], MaCH [90], FaSD [91], NGSEP [91] or VarDict [92], many misassembly problems, which have adverse effects on the downstream analysis, still remain. The misassemblies are generated by many factors, such as the heterozygosity and ploidy of genomes, repeats, low read depth or algorithmic limitations. For small genomes, the misassembly can be evaluated and/or fixed with Pilon [93], which uses paired-end and mate-pair reads. However, in the case of large genomes (i.e. vertebrates or plants), the misassemblies are poorly detected or resolved unless accurate reference genomes are provided. If a reference genome is provided, the quality assessment of the resulting assembly can be performed by using publicly available tools, such as GAGE [49], QUAST [83], REAPR [94], BUSCO [95], misFinder [96] or Mauve [97]. However, it should be noted that the detected misassemblies, such as insertions, deletions, inversions or translocations, may not be misassemblies, but may instead be existing structural variants.

Comparison of genome assemblers

The Assemblathon1 and 2 [30, 31] and GAGE-A and B [49, 98] competitions have evaluated assembly pipelines proposed by multiple teams using the same data, on the basis of computational cost and performance. In Assemblathon 2, which tested the assemblies of bird (*Melopsittacus undulatus*), fish (*Maylandia zebra*) and snake (*Boa constrictor* genomes), the pipeline proposed by the Baylor College of Medicine Human Genome Sequencing Center (BCM-HGSC) showed the best performance in the assemblies of the *M. undulatus* and *M. zebra* genomes in terms of contig and scaffold N50s (Figure 8). The pipeline integrated multiple software packages on the basis of ALLPATHS-LG. With the exception of the BMC-HGSC pipeline, the competition concluded that there was no absolute winner assembler for all examined species and that the performance depended on the species. For example, the SGA outperformed other assemblers in the assembly of the *B. constrictor* genome, but not the genomes of the other species.

Additionally, we surveyed several *de novo* assemblies of vertebrates from the NCBI Genome Resources (http://www.ncbi.nlm.nih.gov/genome/), which are elaborated in the list shown in Figure 8. The contig and scaffold N50s of ALLPATHS-LG are slightly greater than the other assemblers, although ALLPATHS-LG tends to have a greater gap percentage than do the other assemblers.

Long SMS reads

The advantages of the long SMS read

As discussed earlier, repetitive structures, uneven sequencing depth and missing gaps are substantial challenges in *de novo* assembly, particularly for the second-generation short reads. Although large-insert reads can mitigate the problems somewhat, they do not completely resolve the problems in genomic regions with long repeats, e.g. LINEs (6–8 kb) or LTRs (1–10 kb) [4, 5, 99], and in long, low-depth regions or gaps. In contrast, long SMS reads that cover the repeats could easily address the problem (Figure 7C), and because the long SMS reads are generated by a PCR-free method and are less biased to regions with high GC or AT contents [57, 100–102], they are greatly beneficial in overcoming the uneven sequencing depth and gap problems.

According to Koren and Phillippy [103], the complexity of the de Bruijn graph disappears if the *k*-mer length is greater than the 'golden threshold' of 7 kb; thus, the substantial challenges in *de novo* assembly can be resolved if the read length is >7 kb.

	Sequencing platform	Type	Assembler	Source	Total scaffold size (Gb)	Contig N50 (Kb)	Scaffold N50 (Mb)	Cont.N50/error	Gap (%)
Fish	Illumina, 454	Eulerian	BCM-HGSC pipeline	*Lake Malawi cichlid*(ASSEMBLATHON2)	1.00	31.67	4.97	3.95	12.51
	Illumina		ALLPATHS-LG		0.85	20.13	4.01	2.62	15.02
		Hamiltonian	SOAPdenovo2		1.10	8.15	1.33	1.34	25.20
			ABySS		0.88	5.96	1.07	1.02	19.48
			Meraculous		0.84	6.59	0.88	1.07	18.57
		String Graph	SGA		0.81	7.95	0.11	1.58	17.27
Reptilia	Illumina, 454	Eulerian	BCM-HGSC pipeline	*Boa constrictor constricta* (ASSEMBLATHON2)	1.46	17.55	1.53	11.30	7.58
	Illumina	Hamiltonian	ABySS		1.53	27.89	0.47	24.93	3.34
			SOAPdenovo2		1.62	17.87	1.77	173	6.57
			Meraculous		1.44	35.62	1.24	846	3.56
		String Graph	SGA		1.44	29.33	4.51	543	3.86
Aves	Illumina, 454	Eulerian	BCM-HGSC pipeline	*Melopsittacus undulatus* (ASSEMBLATHON2)	1.33	192.00	13.00	435	11.51
	Illumina		ALLPATHS-LG		1.33	57.06	14.95	501	3.68
		Hamiltonian	SOAPdenovo2		1.17	38.14	13.50	370	3.32
			Meraculous		1.09	39.05	9.83	930	4.30
		String Graph	SGA		1.15	18.18	2.80	337	7.45
Reptilia	Illumina	Eulerian	ALLPATHS-LG	*Crocodylus porosus*(Cpor_2.0)	2.12	32.74	0.20	-	1.53
				Thamnophis sirtalis(Thamnophis_sirtalis-6.0)	1.42	10.45	0.65	-	21.21
				Vipers berus berus(Vber.be_1.0)	1.53	11.73	0.13	-	14.10
		Hamiltonian	ABySS	*Crotalus horridus* (ASM162548v1)	1.52	5.84	0.02	-	12.37
			SOAPdenovo1	*Pelodiscus sinensis* (PelSin_1.0)	2.20	21.99	3.35	-	4.35
				Chelonia mydas (CheMyd_1.0)	2.21	29.24	3.86	-	4.44
				Apalone spinifera (ASM38561v1)	1.93	4.69	2.31	-	2.75
				Alligator sinensis (ASM45574v1)	2.27	23.41	2.19	-	3.17
Aves		Eulerian	ALLPATHS-LG	*Aquila chrysaetos canadensis*(Aquila_chrysaetos-1.0.2)	1.19	172.33	9.23	-	1.07
				Zosterops lateralis melanops(ASM128173v1)	1.04	34.51	3.58	-	3.31
				Lepidothrix coronata(Lepidothrix_coronata-1.0)	1.08	141.85	4.97	-	2.07
				Zonotrichia albicollis(Zonotrichia_albicollis-1.0.1)	1.05	112.75	4.87	-	4.40
				Corvus cornix cornix(Hooded_Crow_genome)	1.05	94.38	16.36	-	2.62
		Hamiltonian	SOAPdenovo1	*Haliaeetus leucocephalus* (Haliaeetus_leucocephalus-4.0)	1.18	105.49	9.15	-	1.63
				Falco peregrinus (F_peregrinus_v1.0)	1.17	28.65	3.94	-	1.58
				Falco cherrug (F_cherrug_v1.0)	1.17	31.33	4.15	-	2.03
				Corvus brachyrhynchos (ASM69197v1)	1.09	29.09	6.95	-	3.62
			SOAPdenovo2	*Phoenicopterus ruber ruber*(ASM68726v1)	2.43	31.90	2.42	-	1.33
				Geospiza fortis (GeoFor_1.0)	1.07	30.52	5.26	-	2.25
				Columba livia (Cliv_1.0)	1.11	26.58	3.15	-	1.90
Mammalia		Eulerian	ALLPATHS-LG	*Eptesicus fuscus* (EptFus1.0)	2.03	21.39	13.45	-	10.62
				Neotoma lepida (ASM167557v1)	2.35	9.58	0.15	-	12.65
				Cebus capucinus imitator (Cebus_imitator-1.0)	2.72	41.20	5.27	-	3.94
				Cricetulus griseus (Cgr1.0)	2.33	11.90	1.24	-	10.45
				Heterocephalus glaber (HetGla_female_1.0)	2.62	47.76	20.53	-	11.59
				Echinops telfairi (EchTel2.0)	2.95	20.43	45.76	-	11.60
				Ceratotherium simum simum(CerSimSim1.0)	2.46	92.96	26.28	-	3.96
				Octodon degus (OctDeg1.0)	3.00	19.85	12.09	-	15.68
				Sorex araneus (SorAra2.0)	2.42	22.62	22.79	-	9.54
				Mesocricetus auratus (MesAur1.0)	2.50	22.51	12.75	-	17.12
	Illumina, 454	Eulerian+OLC	MaSuRCA	*Bison bison bison* (Bison_UMD1.0)	2.83	19.97	7.19	-	6.92
	Illumina			*Bubalus bubalis* (UMD_CASPUR_WB_2.0)	2.84	21.94	1.41	-	2.62
				Choloepus hoffmanni(C_hoffmanni-2.0.1)	3.29	64.49	0.37	-	1.91
		String Graph	SGA	*Manis javanica* (ManJav1.0)	2.55	38.92	0.20	-	7.77
		Hamiltonian	ABySS	*Camelus dromedarius* (Cdrom64K)	2.06	40.20	1.48	-	2.60
				Ellobius lutescens (ASM168507v)	2.35	11.65	0.24	-	3.95
	Illumina, 454, SOLID		SOAPdenovo1	*Cricetulus griseus* (CriGri_1.0)	2.40	39.36	1.15	-	3.40
	Illumina			*Camelus ferus* (CB1)	2.01	90.26	2.01	-	1.18
				Myotis brandtii (ASM41265v1)	2.06	15.18	3.45	-	8.80
				Tupaia chinensis (TupChi_1.0)	2.85	25.94	3.87	-	4.92
				Ursus maritimus (UrsMar_1.0)	2.30	46.51	15.94	-	1.67
				Equus przewalskii (Burgud)	2.40	57.61	0.51	-	0.88
				Camelus dromedarius (PRJNA234474_Ca_dromedarius_V1.0)	2.00	69.13	4.19	-	1.13
				Sus scrofa (pig)	2.49	45.71	1.05	-	3.03
			SOAPdenovo2	*Myotis brandtii* (ASM41265v1)	2.11	23.29	3.23	-	5.95
				Miniopterus natalensis (Mnat.v1)	1.80	29.78	4.32	-	3.78
				Nannospalax galili (S.galili_v1.0)	3.06	30.35	3.62	-	4.83
				Sus scrofa (minipig_v1.0)	2.51	31.94	5.85	-	2.21

Figure 8. The *de novo* sort read assemblies of the bird (*M. undulatus*), fish (*M. zebra*) and snake (*B. constrictor*) genomes were reanalyzed using the Supplementary Table of Assemblathon2 [31]. In addition, many more assembly results below Assemblathon2 were collected from the NCBI assembly database.

Because the average lengths of the long PacBio or Oxford Nanopore reads are approximately the length of the golden threshold, it would be more beneficial to take advantage of the long PacBio or Oxford Nanopore reads to overcome the challenges.

De novo assembly for long reads

Generally, the assemblers for long SMS reads are classified into hybrid or long-read-only methods. The hybrid methods refer to those using both long SMS reads and short NGS reads to increase the quality of the assembly and to reduce sequencing cost, whereas the long-read-only methods refer to the methods that use only long SMS reads to correct the sequencing errors and generate the assembly.

The hybrid/long-read-only assembly pipelines can be classified into the five categories. (i) The first category includes the sequencing error correction methods for long SMS reads. SMS platform, such as PacBio or Oxford Nanopore, is highly error prone (PacBio RSII: ~13% error; Oxford Nanopore MinION: ~20% error), and the errors tend to occur in regions with low sequence complexity [12, 13, 104–106]. The hybrid assembly pipeline uses hybrid error correction methods, whereas the long-read-only assembly pipeline uses long-read-only methods. (ii) The second category is the approach for contig assembly. The error-corrected long reads are normally assembled by using overlap-based approaches (Celera [70], Allora [107], AMOS [108] and MIRA [109] for OLC or FALCON [110] for string graphs), rather than the de Bruijn graph approach. (iii) The third category is the aligners that generate the overlaps among the long reads. Current pipelines often use BLASR [111], MinHash [112] and DALIGNER [113]. (iv) The fourth category is the scaffolding using long SMS reads, which is used for only the hybrid assembly pipeline that connects contigs of short NGS reads. (v) The last category is the gap closers that use long SMS reads, which are also used for the hybrid assembly pipeline that closes the gaps between contigs of short NGS reads. The existing hybrid/long-read-only assembly pipelines are summarized by category in Table 4.

In the following subsections, we will further discuss the hybrid and long-read-only assemblies, overlap-based approaches for contig assembly, scaffolding by long reads and gap filling by long reads.

Hybrid assembly

There are two types of hybrid assemblies: (i) The assembly of long reads corrected with the support of short reads, and PBcR [36] and Nanocorr [101] are known as type-1 hybrid assembly pipelines (Table 4). (ii) The contig assembly by short read and the scaffolding by long reads. Cerulean [114], hybridSPades [115], AHA [107] and DBG2OLC [116] are known as type-2. PBcR

Table 4. Assembly pipelines for long SMS reads

Pipelines	Type	Input	Error corrector	Contig assembler	Aligner	Scaffolder
Cerulean	Hybrid	Illumina, PacBio	PBDAG-Con	ABySS (short read)	BLASR	Cerulean
hybridSPades	Hybrid	Illumina, PacBio	PBDAG-Con	SPAdes (short read)	(Own graph aligner)	exSPAnder
DBG2OLC	Hybrid	Illumina, PacBio	Sparc	SparseAssembler (short read)	(Own program)	DBG2OLC
AHA	Hybrid	Illumina, PacBio	AMOS	Allora	BLASR	AHA
PBcR	Hybird/long-read-only	Illumina, PacBio	PBDAG-Con	Celera	BLASR	—
HGAP	Long-read-only	PacBio	PBDAG-Con	Celera	BLASR	—
HMAP	Long-read-only	PacBio	PBDAG-Con	Celera	MHAP	—
FALCON	Long-read-only	PacBio	(own program)	(Own program)	DALIGNER	—
Nanocorr	Hybrid	Illumina, Nanopore	PBDAG-Con	Celera	BLAST	—

has been successfully applied to assemble large genomes [36, 112] by using both short NGS reads and long PacBio reads. In the Assemblathon2 competition, the CBCB team used PBcR with PacBio, 454 and Illumina reads to assemble the *M. undulatus* genome and reported the best result for contig N50 and coverage (Figure 8).

Hybrid error correction

Recently, new technologies have been introduced to reduce the sequencing errors in the long SMS reads. In the case of the PacBio, circular consensus reads tend to be shortened by approximately only ∼400 nt, thereby this method lacks the merits of the long-read approach [36], despite its high accuracy (∼98% per base [117]). Several groups have suggested hybrid error correction approaches that use both short and long reads in the hybrid assembly pipelines, to correct the sequencing errors in the long SMS reads with low sequencing costs by retaining the advantage of the long reads. For instance, LoRDEC [118], proovread [119], ECTools [120] and Jabba [121] can successfully correct the long reads with the support of short reads. LoRDEC and Jabba, which are computationally efficient, use the de Bruijn graph, whereas the others use multiple sequence alignment-based approaches.

In the case of Oxford Nanopore, the two-dimensional (sense and antisense) sequencing reads are used to reduce the error rates, which are still high [122]. The error correction tools for the PacBio long reads might be used for the Oxford Nanopore long reads; however, some correctors do not work well because their error rates are higher than the PacBio long reads [101]. For that reason, Nanocorr [101] and NaS [105] were specifically designed for the hybrid error correction of Oxford Nanopore reads by using short NGS reads.

The hybrid error correction method can reduce the sequencing errors in long SMS reads with a lower sequencing cost than the long-read-only error correction methods, which are further described in the next subsection. However, the hybrid methods have not performed better than long-read-only methods because (i) the sequences in the GC-rich region are rarely corrected because of the low depth of the short NGS reads and (ii) the sequences of the repeats are poorly corrected by mismapping of the short NGS reads.

Scaffolding and gap filling by long reads

SSPACE-LongRead [123], OPERA-LG [73], LINKS [124], DBG2OLC [116], AHA [125], Cerulean [114] and hybridSPades [115] are scaffolders that connect contigs, as guided by long reads. The detailed algorithms of the scaffolders vary, but all take advantage of the long reads as the backbones of the scaffolds.

This approach has many merits compared with scaffolding by mate-pair reads. It easily solves the problems that occur during scaffolding with NGS reads with a large insert because long SMS reads tend to have less systematic and nucleotide composition biases and to require less computational cost. Furthermore, this approach is flexible for the various existing state-of-the-art assemblers because the main function of this approach is scaffolding.

The gaps produced by the short read assemblers are inefficiently closed by other short NGS reads. Most of the gap closers are based on greedy-like extension processes and do not exhaustively search for the optimal solution; hence, the methods may fall into local minima. Some, but not all, gaps can be closed by existing gap closers, such as IMAGE [23], GapFiller [24], Sealer [26] or GapCloser [25]. However, according to a recent study [27], the misassembly rates by the gap closers that use the short NGS reads are 20–500-fold higher than those of the long read gap closers, such as PBJelly [126] and GMcloser [27].

Gap closing with the long SMS reads should be managed after or during the sequencing error correction. In the case of PBJelly, the remaining gaps in the scaffolds are closed by the consensus of the supporting long PacBio reads. Because PBJelly uses PBDAG-Con for consensus, the long Oxford Nanopore reads may not be suitable for PBJelly (PBDAG-Con does not work well for Oxford Nanopore [127]). For GMcloser [27], the gaps are closed by contigs or error-corrected long PacBio reads.

Long-read-only assembly

An ultimate approach for *de novo* assembly may be to assemble only long reads with sufficient depth because the long-read-only assembly of long reads is promising for capturing genomic structural variations and resolving long repeats in the genomes. HGAP [128] is one of long-read-only assemblers that map smaller long SMS reads to larger ones using BLASR [111] and corrects sequencing errors by using multiple sequence alignment with the consensus tool PBDAG-Con [128] (Table 4). However, PBDAG-Con does not work well for long Oxford Nanopore reads because of the differences in the error profile [127]. For the long Oxford Nanopore reads, one study has found that Sparc [127], a new consensus tool for both the long PacBio and Oxford Nanopore reads, works better than other tools.

The corrected reads are then assembled by overlap-based assemblers, such as Celera [70] or MIRA [109], and the resulting assembly is polished by the consensus algorithm Quiver [128]. However, this process for long-read-only assembly requires at

least 50X PacBio reads to construct a high-quality assembly, requiring substantial sequencing costs for large genomes [129]. Hence, recent assembly projects using the long-read-only approach have mainly been applied to species with small genomes, such as bacteria and viruses [36, 99, 103, 116, 118, 123, 128, 130–137]. Regardless of the high sequencing cost, several plant and animal genomes have been successfully assembled by using long-read-only assemblers (Table 5).

In long-read-only assembly, the alignment process is a major bottleneck because the long SMS reads should be aligned end to end to determine the overlapping pairs. To solve this problem, Koren *et al.* [112] have introduced MHAP, a long-read-only assembler pipeline that aligns the long PacBio reads by using the MinHash technique [143], corrects sequencing errors using PBcR [36] in a long-read-only manner, assembles the reads using the Celera assembler [70] and then polishes the reads with Quiver [128]. Using MinHash, this approach reduces both the execution time and required RAM memory. As a result, Koren *et al.* have assembled 20 Mb N50 contigs for *Drosophila melanogaster* and 4.3 Mb N50 contigs for human (Table 5) [112]. Among the contigs, the N50 size of *D. melanogaster* almost reaches the chromosome level.

Overlap-based approach

The overlap-based approach is a straightforward approach for long read assembly because it assembles the long reads themselves without converting to *k*-mers. In this subsection, we discuss two types of overlap-based assemblers, which are OLC and string graph assemblers.

Assemblers using the OLC approach, such as Celera [70], Allora [107], AMOS [108] and MIRA [109], have been introduced to relieve the local optimal problem. The OLC approach constructs an overlap graph based on pairwise sequence alignments by dynamic programming (overlap), extracts a nonredundant graph (layout) and merges the reads while correcting the sequencing errors (consensus). The OLC approaches usually produce robust results for relatively low-throughput long reads, such as 454 sequencing reads (~500 nt), Sanger sequencing reads (~1 kb) and long PacBio reads (up to a few tens of kilobases), but not for high-throughput short reads (generally, billions of short reads are used for large genomes) because a tremendous number of pairwise sequence alignments are required for the short reads to construct an overlap graph. This overlap graph of the short reads (similarly presented as a Hamiltonian de Bruijn graph) easily forms a complex graph that includes a large number of nodes and branches. In contrast, an overlap graph for the low-throughput long reads generally forms a much less complex graph.

The main hurdle of OLC approach is to properly and efficiently align long reads to identify overlaps. String graph assembly was first introduced by E. Myers [144] to overcome this hurdle; it uses *k*-mers to identify overlaps between reads. A typical string graph assembly pipeline for long SMS reads is FALCON, which implements DALIGNER [113], which uses *k*-mers to find overlaps. This approach can reduce the alignment time by as much as an order of magnitude as compared with BLASR [111]. Another string graph assembly pipeline, MHAP, uses the MinHash *k*-mer method [143] and reduces the alignment time to one similar to the time of DALIGNER. Note that the short read assembler SGA is also one of string graph assembler, but normally used for short NGS reads.

Table 5. Recent whole-genome assembly of large genomes for PacBio long reads

Species	Pipeline	N50 (contig) (in Mb)	Reference
Oropetium (grass)	HGAP	2.39	[138]
Arabidopsis thaliana	HGAP	6.36	[139]
	MHAP	11.16	[112]
D. melanogaster	PBcR	15.3	[140]
	MHAP	20.99	[112]
Capra hircus (goat)	PBcR	2.56	[141]
Homo sapiens	HGAP	2.56	[142]
	FALCON	4.38	[130]
	MHAP	4.32	[112]
	FALCON[a]	26.9	—

Note: Error-corrected PacBio long reads were assembled with Celera assembler [70] in all pipelines except FALCON [110].
[a]GenBank assembly accession: GCA_001297185.1.

Discussion

In this review, we have discussed the challenges in *de novo* whole-genome assembly for short NGS reads and how these challenges can be overcome. Notably, the methods using short NGS reads perform poorly for repetitive structures or GC-biased regions. Assemblers using long reads are highly helpful in resolving the problems, and they generate more contiguous results [36, 99, 106, 116, 118, 123, 128, 130–137]. However, the sequencing and computational costs are currently substantially higher for long SMS reads. At a minimum, the sequencing cost problem can be mitigated by the hybrid approaches of short NGS reads and long SMS reads, although there is an argument that the hybrid approach may produce numerous misassemblies [145].

The scaffold size should be close to the chromosome scale with a low number of gaps to ultimately generate the *de novo* assembly. Recently, chromosome-scale scaffolding has been achieved by using advanced physical mapping methods, such as optimal mapping [146] and chromatin-interaction mapping [147]. Optimal mapping takes advantage of genome-wide restriction site maps from a single DNA, generating ordered, high-resolution restriction maps. In contrast, genome-wide chromatin interaction mapping provides ordered information about the long-range interactions within a chromosome, including centromeres, and was originally designed to detect the regulation of gene expression by the three-dimensional interactions of chromosomes. Recently, this method has been applied to ultra-long-range scaffolding of *de novo* assemblies, building highly qualitative chromosome-scale scaffolds. Given the successful scaffolding methods, the advent of new technologies for generating genome-wide, ultra-long-range physical maps should significantly improve the quality of the *de novo* assemblies in the near future.

Future sequencing technologies may also offer improvements. Currently, the original PacBio long read sequencing is expensive. However, with the advent of Oxford Nanopore MinION and PacBio sequel platforms [148], inexpensive long read sequencing technologies are within reach and may lead to long-read-only assembly being more frequently used. It has been also expected that quantum sequencing technologies may further reduce the cost problem by increasing the throughput and read length [149–151]. According to Di Ventra and Taniguchi [150], the throughput of quantum sequencing should reach ~100 Tb per day by 2018. Although this throughput is only an expectation, it appears promising that the throughput

problem of long SMS reads can be solved. Once the high-throughput sequencing of long reads becomes a reality, the current long read assemblers would be not suitable, owing to overflowing memory, and thus a high-priority challenge in *de novo* assembly will be the development of new assembly algorithms with efficient memory and computational costs.

The Oxford Nanopore Company provides an online-based platform for real-time data analysis. The real-time assemblies for large genomes are not currently available, although assemblies for the small bacterial and eukaryotic genomes are available [152]. For the real-time assembly of large genomes, the error correction and assembly algorithms for erroneous long reads must be much more efficient than ever before. In particular, the Oxford Nanopore reads often include indels, and the indel-mediated mismatches may be abundantly detected during read overlapping and error correction, thus potentially resulting in error-prone assembly. Methods for addressing these problems must be developed in the future.

In addition, as interest in precision medicine and personalized genomics increases, scientists and nonscientists are expecting to analyze their genomes to understand their current and future health conditions on a personal computer or smartphone. Genome analyzers that include assembly may need to be implemented as a light application that requires a small amount of memory and computing to make precision, personalized medicine a reality. Together, the data suggest that given the advanced technologies and clinical interests in personal genomes, more memory- and computing-efficient technologies to generate *de novo* assembly of personal genomes and metagenomes will be beneficial for clinical uses in the future.

Key Points

- The complexity of the de Bruijn graph is key to solving the trade-off between computational cost and the accuracy.
- The execution time of the assemblers based on the Hamiltonian de Bruijn graph, compared with the Eulerian de Bruijn graph, is more efficiently reduced when the de Bruijn graph is simplified because the complexity is reduced.
- Long SMS reads, such as PacBio or Oxford Nanopore, are needed to overcome the challenges resulting from the complexity of the genome structure.

Acknowledgments

We thank the members of BIG lab. for helpful comments and discussions.

Funding

The Basic Science Research Program through the NRF, which was funded by the Ministry of Science, ICT and Future Planning (grant number NRF-2014M3C9A3063541), the Korea Health Technology R&D Project through the Korea Health Industry Development Institute (KHIDI), which was funded by the Ministry of Health and Welfare (grant number HI15C3224) and the Cooperative Research Program for Agriculture Science and Technology Development (Project title: National Agricultural Genome Program, Project No. PJ01045303).

References

1. Miller JR, Koren S, Sutton G. Assembly algorithms for next-generation sequencing data. *Genomics* 2010;**95**:315–27.
2. Compeau PE, Pevzner PA, Tesler G. How to apply de Bruijn graphs to genome assembly. *Nat Biotechnol* 2011;**29**:987–91.
3. Nagarajan N, Pop M. Sequence assembly demystified. *Nat Rev Genet* 2013;**14**:157–67.
4. Cordaux R, Batzer MA. The impact of retrotransposons on human genome evolution. *Nat Rev Genet* 2009;**10**:691–703.
5. Koito A, Ikeda T. Intrinsic immunity against retrotransposons by APOBEC cytidine deaminases. *Front Microbiol* 2013;**4**:28.
6. Luo C, Tsementzi D, Kyrpides N, *et al*. Direct comparisons of Illumina vs. Roche 454 sequencing technologies on the same microbial community DNA sample. *PLoS One* 2012;**7**:e30087.
7. Mardis ER. The impact of next-generation sequencing technology on genetics. *Trends Genet* 2008;**24**:133–41.
8. Pop M, Salzberg SL. Bioinformatics challenges of new sequencing technology. *Trends Genet* 2008;**24**:142–9.
9. Metzker ML. Sequencing technologies - the next generation. *Nat Rev Genet* 2010;**11**:31–46.
10. Schadt EE, Turner S, Kasarskis A. A window into third-generation sequencing. *Hum Mol Genet* 2010;**19**:R227–40.
11. Huang YF, Chen SC, Chiang YS, *et al*. Palindromic sequence impedes sequencing-by-ligation mechanism. *BMC Syst Biol* 2012;**6 (Suppl 2)**:S10.
12. Liu L, Li Y, Li S, *et al*. Comparison of next-generation sequencing systems. *J Biomed Biotechnol* 2012;**2012**:251364.
13. Quail MA, Smith M, Coupland P, *et al*. A tale of three next generation sequencing platforms: comparison of Ion Torrent, Pacific Biosciences and Illumina MiSeq sequencers. *BMC Genomics* 2012;**13**:341.
14. 1000 Genomes Project Consortium; Abecasis GR, Auton A, *et al*. An integrated map of genetic variation from 1,092 human genomes. *Nature* 2012;**491**:56–65.
15. 1000 Genomes Project Consortium, Abecasis GR, Altshuler D, *et al*. A map of human genome variation from population-scale sequencing. *Nature* 2010;**467**:1061–73.
16. Pendleton M, Sebra R, Pang AW, *et al*. Assembly and diploid architecture of an individual human genome via single-molecule technologies. *Nat Methods* 2015;**12**:780–6.
17. Chaisson MJ, Wilson RK, Eichler EE. Genetic variation and the *de novo* assembly of human genomes. *Nat Rev Genet* 2015;**16**:627–40.
18. Motahari AS, Bresler G, Tse DNC. Information theory of DNA shotgun sequencing. *IEEE Trans Inf Theory* 2013;**59**:6273–89.
19. Paszkiewicz K, Studholme DJ. *De novo* assembly of short sequence reads. *Brief Bioinform* 2010;**11**:457–72.
20. El-Metwally S, Hamza T, Zakaria M, *et al*. Next-generation sequence assembly: four stages of data processing and computational challenges. *PLoS Comput Biol* 2013;**9**:e1003345.
21. He Y, Zhang Z, Peng X, *et al*. *De novo* assembly methods for next generation sequencing data. *Tsinghua Sci Technol* 2013;**18**:500–14.
22. Simpson JT, Pop M. The theory and practice of genome sequence assembly. *Annu Rev Genomics Hum Genet* 2015;**16**:153–72.
23. Tsai IJ, Otto TD, Berriman M. Improving draft assemblies by iterative mapping and assembly of short reads to eliminate gaps. *Genome Biol* 2010;**11**:R41.
24. Boetzer M, Pirovano W. Toward almost closed genomes with GapFiller. *Genome Biol* 2012;**13**:R56.

25. Luo R, Liu B, Xie Y, *et al*. SOAPdenovo2: an empirically improved memory-efficient short-read *de novo* assembler. *Gigascience* 2012;**1**:18.
26. Paulino D, Warren RL, Vandervalk BP, *et al*. Sealer: a scalable gap-closing application for finishing draft genomes. *BMC Bioinformatics* 2015;**16**:230.
27. Kosugi S, Hirakawa H, Tabata S. GMcloser: closing gaps in assemblies accurately with a likelihood-based selection of contig or long-read alignments. *Bioinformatics* 2015;**31**:3733–41.
28. Berkeley L, Chapman JA, Ho I, *et al*. Meraculous: *de novo* genome assembly with short paired-end reads. *PLoS One* 2011;**6**:e23501.
29. Gnerre S, Maccallum I, Przybylski D, *et al*. High-quality draft assemblies of mammalian genomes from massively parallel sequence data. *Proc Natl Acad Sci USA* 2011;**108**:1513–8.
30. Earl D, Bradnam K, St John J, *et al*. Assemblathon 1: a competitive assessment of *de novo* short read assembly methods. *Genome Res* 2011;**21**:2224–41.
31. Bradnam KR, Fass JN, Alexandrov A, *et al*. Assemblathon 2: evaluating *de novo* methods of genome assembly in three vertebrate species. *Gigascience* 2013;**2**:10.
32. Simpson JT. Exploring genome characteristics and sequence quality without a reference. *Bioinformatics* 2014;**30**:1228–35.
33. Chu TC, Lu CH, Liu T, *et al*. Assembler for *de novo* assembly of large genomes. *Proc Natl Acad Sci USA* 2013;**110**:E3417–24.
34. Ye C, Ma ZS, Cannon CH, *et al*. Exploiting sparseness in *de novo* genome assembly. *BMC Bioinformatics* 2012;**13**:S1.
35. Chikhi R, Rizk G. Space-efficient and exact de Bruijn graph representation based on a Bloom filter. *Algorithms Mol Biol* 2013;**8**:22.
36. Koren S, Schatz MC, Walenz BP, *et al*. Hybrid error correction and *de novo* assembly of single-molecule sequencing reads. *Nat Biotechnol* 2012;**30**:693–700.
37. Zimin A, Stevens KA, Crepeau MW, *et al*. Sequencing and assembly of the 22-Gb loblolly pine genome. *Genetics* 2014;**196**:875–90.
38. Zimin A, Marcais G, Puiu D, *et al*. Assembly Improvements to Move Beyond Loblolly Pine Assembly v1. 0. In: *Plant and Animal Genome XXII Conference*. Sandiago, CA, USA, 2014.
39. Pevzner PA, Tang H, Waterman MS. An Eulerian path approach to DNA fragment assembly. *Proc Natl Acad Sci USA* 2001;**98**:9748–53.
40. Simpson JT, Durbin R. Efficient *de novo* assembly of large genomes using compressed data structures. *Genome Res* 2012;**22**:549–56.
41. Simpson JT, Wong K, Jackman SD, *et al*. ABySS: a parallel assembler for short read sequence data. *Genome Res* 2009;**19**:1117–23.
42. Zerbino DR, Birney E. Velvet: algorithms for *de novo* short read assembly using de Bruijn graphs. *Genome Res* 2008;**18**:821–9.
43. Held M, Karp RM. A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied Mathematics*. 1962;**10**: 196–210.
44. Thomason A. A simple linear expected time algorithm for finding a hamilton path. *Discrete Math* 1989;**75**:373–9.
45. Li R, Zhu H, Ruan J, *et al*. *De novo* assembly of human genomes with massively parallel short read sequencing. *Genome Res* 2010;**20**:265–72.
46. Cole R, Ost K, Schirra S. Edge-coloring bipartite multigraphs in O(ElogD) time. *Combinatorica* 2001;**21**:5–12.
47. Bankevich A, Nurk S, Antipov D, *et al*. SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. *J Comput Biol* 2012;**19**:455–77.
48. Zimin AV, Marçais G, Puiu D, *et al*. The MaSuRCA genome assembler. *Bioinformatics* 2013;**29**:2669–77.
49. Salzberg SL, Phillippy AM, Zimin A, *et al*. GAGE: a critical evaluation of genome assemblies and assembly algorithms. *Genome Res* 2012;**22**:557–67.
50. Mardis ER. Next-generation DNA sequencing methods. *Annu Rev Genomics Hum Genet* 2008;**9**:387–402.
51. Schirmer M, Ijaz UZ, D'Amore R, *et al*. Insight into biases and sequencing errors for amplicon sequencing with the Illumina MiSeq platform. *Nucleic Acids Res* 2015;**43**:e37.
52. Alic AS, Ruzafa D, Dopazo J, *et al*. Objective review of *de novo* stand-alone error correction methods for NGS data. *Wiley Interdiscip Rev Comput Mol Sci* 2016;**6**:111–46.
53. Xie C, Tammi MT. CNV-seq, a new method to detect copy number variation using high-throughput sequencing. *BMC Bioinformatics* 2009;**10**:80.
54. Medvedev P, Fiume M, Dzamba M, *et al*. Detecting copy number variation with mated short reads. *Genome Res* 2010;**20**:1613–22.
55. Aird D, Ross MG, Chen WS, *et al*. Analyzing and minimizing PCR amplification bias in Illumina sequencing libraries. *Genome Biol* 2011;**12**:R18.
56. Oyola SO, Otto TD, Gu Y, *et al*. Optimizing Illumina next-generation sequencing library preparation for extremely AT-biased genomes. *BMC Genomics* 2012;**13**:1.
57. Ross MG, Russ C, Costello M, *et al*. Characterizing and measuring bias in sequence data. *Genome Biol* 2013;**14**:R51.
58. Chen YC, Liu T, Yu CH, *et al*. Effects of GC bias in next-generation-sequencing data on *de novo* genome assembly. *PLoS One* 2013;**8**:e62856.
59. Sims D, Sudbery I, Ilott NE, *et al*. Sequencing depth and coverage: key considerations in genomic analyses. *Nat Rev Genet* 2014;**15**:121–32.
60. Allam A, Kalnis P, Solovyev V. Karect: accurate correction of substitution, insertion and deletion errors for next-generation sequencing data. *Bioinformatics* 2015;**31**:3421–8.
61. Laehnemann D, Borkhardt A, McHardy AC. Denoising DNA deep sequencing data-high-throughput sequencing errors and their correction. *Brief Bioinform* 2016;**17**:154–79.
62. Li X, Waterman MS. Estimating the repeat structure and length of DNA sequences using L-tuples. *Genome Res* 2003;**13**:1916–22.
63. Kelley DR, Schatz MC, Salzberg SL. Quake: quality-aware detection and correction of sequencing errors. *Genome Biol* 2010;**11**:R116.
64. Lander ES, Linton LM, Birren B, *et al*. Initial sequencing and analysis of the human genome. *Nature* 2001;**409**:860–921.
65. Meacham F, Boffelli D, Dhahbi J, *et al*. Identification and correction of systematic error in high-throughput sequence data. *BMC Bioinformatics* 2011;**12**:451.
66. Liu B, Shi Y, Yuan J, *et al*. Estimation of genomic characteristics by analyzing k-mer frequency in *de novo* genome projects. arxiv preprint arXiv:1308.2012, 2013.
67. Song L, Florea L, Langmead B. Lighter: fast and memory-efficient sequencing error correction without counting. *Genome Biology* 2014;**15**:509.
68. Treangen TJ, Salzberg SL. Repetitive DNA and next-generation sequencing: computational challenges and solutions. *Nat Rev Genet* 2012;**13**:36–46.
69. Peng Y, Leung HC, Yiu SM, *et al*. IDBA-UD: a *de novo* assembler for single-cell and metagenomic sequencing data with highly uneven depth. *Bioinformatics* 2012;**28**:1420–8.
70. Myers EW, Sutton GG, Delcher AL, *et al*. A whole-genome assembly of Drosophila. *Science* 2000;**287**:2196–204.

71. Dayarian A, Michael TP, Sengupta AM. SOPRA: scaffolding algorithm for paired reads via statistical optimization. *BMC Bioinformatics* 2010;**11**:345.
72. Boetzer M, Henkel CV, Jansen HJ, *et al*. Scaffolding preassembled contigs using SSPACE. *Bioinformatics* 2011;**27**:578–9.
73. Gao S, Bertrand D, Chia BK, *et al*. OPERA-LG: efficient and exact scaffolding of large, repeat-rich eukaryotic genomes with performance guarantees. *Genome Biol* 2016;**17**:102.
74. Gao S, Sung WK, Nagarajan N. Opera: reconstructing optimal genomic scaffolds with high-throughput paired-end sequences. *J Comput Biol* 2011;**18**:1681–91.
75. Huang W, Li L, Myers JR, *et al*. ART: a next-generation sequencing read simulator. *Bioinformatics* 2012;**28**:593–4.
76. Ono Y, Asai K, Hamada M. PBSIM: PacBio reads simulator–toward accurate genome assembly. *Bioinformatics* 2013;**29**:119–21.
77. Li H. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. arXiv preprint arXiv:1303.3997, 2013.
78. Conway TC, Bromage AJ. Succinct data structures for assembling large genomes. *Bioinformatics* 2011;**27**:479–86.
79. Ferragina P, Manzini G. Opportunistic data structures with applications. In: *The 41st Annual Symposium on Foundations of Computer Science*, Redondo Beach, CA, USA, 2000.
80. Melsted P, Pritchard JK. Efficient counting of k-mers in DNA sequences using a bloom filter. *BMC Bioinformatics* 2011; **12**:333.
81. Pell J, Hintze A, Canino-Koning R, *et al*. Scaling metagenome sequence assembly with probabilistic de Bruijn graphs. *Proc Natl Acad Sci USA* 2012;**109**:13272–7.
82. Butler J, MacCallum I, Kleber M, *et al*. ALLPATHS: *de novo* assembly of whole-genome shotgun microreads. *Genome Res* 2008;**18**:810–20.
83. Gurevich A, Saveliev V, Vyahhi N, *et al*. QUAST: quality assessment tool for genome assemblies. *Bioinformatics* 2013;**29**:1072–5.
84. Li H, Ruan J, Durbin R. Mapping short DNA sequencing reads and calling variants using mapping quality scores. *Genome Res* 2008;**18**:1851–8.
85. Li H, Handsaker B, Wysoker A, *et al*. The sequence alignment/map format and SAMtools. *Bioinformatics* 2009;**25**:2078–9.
86. Li R, Li Y, Kristiansen K, *et al*. SOAP: short oligonucleotide alignment program. *Bioinformatics* 2008;**24**:713–4.
87. Li R, Li Y, Fang X, *et al*. SNP detection for massively parallel whole-genome resequencing. *Genome Res* 2009;**19**:1124–32.
88. Goya R, Sun MG, Morin RD, *et al*. SNVMix: predicting single nucleotide variants from next-generation sequencing of tumors. *Bioinformatics* 2010;**26**:730–6.
89. McKenna A, Hanna M, Banks E, *et al*. The genome analysis toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. *Genome Res* 2010;**20**:1297–303.
90. Li Y, Willer CJ, Ding J, *et al*. MaCH: using sequence and genotype data to estimate haplotypes and unobserved genotypes. *Genet Epidemiol* 2010;**34**:816–34.
91. Xu F, Wang W, Wang P, *et al*. A fast and accurate SNP detection algorithm for next-generation sequencing data. *Nat Commun* 2012;**3**:1258.
92. Lai Z, Markovets A, Ahdesmaki M, *et al*. VarDict: a novel and versatile variant caller for next-generation sequencing in cancer research. *Nucleic Acids Res* 2016;**44**:e108.
93. Walker BJ, Abeel T, Shea T, *et al*. Pilon: an integrated tool for comprehensive microbial variant detection and genome assembly improvement. *PLoS One* 2014;**9**:e112963.
94. Hunt M, Kikuchi T, Sanders M, *et al*. REAPR: a universal tool for genome assembly evaluation. *Genome Biol* 2013;**14**:R47.
95. Simao FA, Waterhouse RM, Ioannidis P, *et al*. BUSCO: assessing genome assembly and annotation completeness with single-copy orthologs. *Bioinformatics* 2015;**31**:3210–2.
96. Zhu X, Leung HC, Wang R, *et al*. misFinder: identify misassemblies in an unbiased manner using reference and paired-end reads. *BMC Bioinformatics* 2015;**16**:386.
97. Darling AE, Tritt A, Eisen JA, *et al*. Mauve assembly metrics. *Bioinformatics* 2011;**27**:2756–7.
98. Magoc T, Pabinger S, Canzar S, *et al*. GAGE-B: an evaluation of genome assemblers for bacterial organisms. *Bioinformatics* 2013;**29**:1718–25.
99. Ashton PM, Nair S, Dallman T, *et al*. MinION nanopore sequencing identifies the position and structure of a bacterial antibiotic resistance island. *Nat Biotechnol* 2015;**33**: 296–300.
100. Gross DC, Lichens-Park A, Kole C (eds). *Genomics of Plant-Associated Bacteria*. Springer, 2014.
101. Goodwin S, Gurtowski J, Ethe-Sayers S, *et al*. Oxford Nanopore sequencing, hybrid error correction, and *de novo* assembly of a eukaryotic genome. *Genome Res* 2015;**25**:1750–6.
102. Oikonomopoulos S, Wang YC, Djambazian H, *et al*. Benchmarking of the Oxford Nanopore MinION sequencing for quantitative and qualitative assessment of cDNA populations. *bioRxiv* 2016;048074.
103. Koren S, Phillippy AM. One chromosome, one contig: complete microbial genomes from long-read sequencing and assembly. *Curr Opin Microbiol* 2015;**23**:110–20.
104. Roberts RJ, Carneiro MO, Schatz MC. The advantages of SMRT sequencing. *Genome Biol* 2013;**14**:405.
105. Madoui MA, Engelen S, Cruaud C, *et al*. Genome assembly using Nanopore-guided long and error-free DNA reads. *BMC Genomics* 2015;**16**:327.
106. Laver T, Harrison J, O'Neill PA, *et al*. Assessing the performance of the Oxford Nanopore Technologies MinION. *Biomol Detect Quantifi* 2015;**3**:1–8.
107. Rasko DA, Webster DR, Sahl JW, *et al*. Origins of the E. coli strain causing an outbreak of hemolytic–uremic syndrome in Germany. *N Engl J Med* 2011;**365**:709–17.
108. Treangen TJ, Sommer DD, Angly FE, *et al*. Next generation sequence assembly with AMOS. *Curr Protoc Bioinformatics* 2011;**Chapter 11**:Unit 8.
109. Chevreux B, Wetter T, Suhai S, Genome sequence assembly using trace signals and additional sequence information. In: *The German Conference on Bioinformatics*, Hannover, Germany, 1999.
110. FALCON. https://github.com/PacificBiosciences/FALCON.
111. Chaisson MJ, Tesler G. Mapping single molecule sequencing reads using basic local alignment with successive refinement (BLASR): application and theory. *BMC Bioinformatics* 2012;**13**:238.
112. Berlin K, Koren S, Chin CS, *et al*. Assembling large genomes with single-molecule sequencing and locality-sensitive hashing. *Nat Biotechnol* 2015;**33**:623–30.
113. Myers G. Efficient local alignment discovery amongst noisy long reads. 2014;**8701**:52–67.
114. Deshpande V, Fung EDK, Pham S, *et al*. Cerulean: a hybrid assembly using high throughput short and long reads. In: *The 13th Workshop on Algorithms in Bioinformatics (WABI 2013)*, Sophia Antipolis, France, 2013.
115. Antipov D, Korobeynikov A, McLean JS, *et al*. hybridSPAdes: an algorithm for hybrid assembly of short and long reads. *Bioinformatics* 2016;**32**:1009–15.

116. CMH CY, Wu Ruan SJ, Ma Z. DBG2OLC: efficient assembly of large genomes using the compressed overlap graph. arXiv:1410.2801v3.
117. Jiao X, Zheng X, Ma L, *et al*. A benchmark study on error assessment and quality control of CCS reads derived from the PacBio RS. *J Data Mining Genomics Proteomics* 2013;**4**:136.
118. Salmela L, Rivals E. LoRDEC: accurate and efficient long read error correction. *Bioinformatics* 2014;**30**:3506–14.
119. Hackl T, Hedrich R, Schultz J, *et al*. proovread: large-scale high-accuracy PacBio correction through iterative short read consensus. *Bioinformatics* 2014;**30**:3004–11.
120. Lee H, Gurtowski J, Yoo S, *et al*. Error correction and assembly complexity of single molecule sequencing reads. *bioRxiv* 2014;006395.
121. Miclotte G, Heydari M, Demeester P, *et al*. Jabba: hybrid error correction for long sequencing reads. *Algorithms Mol Biol* 2016;**11**:10.
122. Jain M, Fiddes IT, Miga KH, *et al*. Improved data analysis for the MinION nanopore sequencer. *Nat Methods* 2015;**12**:351–6.
123. Boetzer M, Pirovano W. SSPACE-LongRead: scaffolding bacterial draft genomes using long read sequence information. *BMC Bioinformatics* 2014;**15**:211.
124. Warren RL, Yang C, Vandervalk BP, *et al*. LINKS: scalable, alignment-free scaffolding of draft genomes with long reads. *Gigascience* 2015;**4**:35.
125. Bashir A, Klammer AA, Robins WP, *et al*. A hybrid approach for the automated finishing of bacterial genomes. *Nat Biotechnol* 2012;**30**:701–7.
126. English AC, Richards S, Han Y, *et al*. Mind the gap: upgrading genomes with Pacific Biosciences RS long-read sequencing technology. *PLoS One* 2012;**7**:e47768.
127. Ye C, Ma ZS. Sparc: a sparsity-based consensus algorithm for long erroneous sequencing reads. *PeerJ* 2016;**4**:e2016.
128. Chin CS, Alexander DH, Marks P, *et al*. Nonhybrid, finished microbial genome assemblies from long-read SMRT sequencing data. *Nat Methods* 2013;**10**:563–9.
129. Van Dijk EL, Auger H, Jaszczyszyn Y, *et al*. Ten years of next-generation sequencing technology. *Trends Genet* 2014;**30**:418–26.
130. Liao Y-C, Lin S-H, Lin H-H. Completing bacterial genome assemblies: strategy and performance comparisons. *Sci Rep* 2015;**5**:8747.
131. Quick J, Quinlan AR, Loman NJ. A reference bacterial genome dataset generated on the MinION(TM) portable single-molecule nanopore sequencer. *Gigascience* 2014;3–22.
132. Miyamoto M, Motooka D, Gotoh K, *et al*. Performance comparison of second-and third-generation sequencers using a bacterial genome with two chromosomes. *BMC Genomics* 2014;**15**:699.
133. Faino L, Thomma BP. Get your high-quality low-cost genome sequence. *Trends Plant Sci* 2014;**19**:288–91.
134. Koren S, Harhay GP, Smith TP, *et al*. Reducing assembly complexity of microbial genomes with single-molecule sequencing. *Genome Biol* 2013;**14**:R101.
135. Fichot EB, Norman RS. Microbial phylogenetic profiling with the Pacific Biosciences sequencing platform. *Microbiome* 2013;**1**:10.
136. Ribeiro FJ, Przybylski D, Yin S, *et al*. Finished bacterial genomes from shotgun sequence data. *Genome Res* 2012;**22**:2270–7.
137. Au KF, Underwood JG, Lee L, *et al*. Improving PacBio long read accuracy by short read alignment. *PLoS One* 2012;**7**:e46679.
138. VanBuren R. *De novo* assembly of a complex panicoid grass genome using ultra-long PacBio reads with P6/C4 chemistry. In: *Plant and Animal Genome XXIII*. San Diego, CA, USA, 2015.
139. PacBio Blog. http://www.pacb.com/blog/data-release-54x-long-read-coverage-for/.
140. Landolin JM, Chin J, Kim K, *et al*. Initial *de novo* Assemblies of the D. melanogaster genome using long-read PacBio sequencing. *Pac Biosci* 2014.
141. Smith TP. A genome assembly of the domestic goat from 70x coverage of single molecule, real-time sequence. In: *Plant and Animal Genome XXIII Conference*. San Diego, CA, USA, 2015.
142. McCombie WR. PacBio Long Read Sequencing and Structural Analysis of a Breast Cancer Cell Line. In: *The 16th annual Advances in Genome Biology and Technology (AGBT) meeting*, Marco Island, FL, 2015.
143. Broder AZ, On the resemblance and containment of documents. In: *Compression and Complexity of Sequences*. Amalfitan Coast, Salerno, Italy, 1997.
144. Myers EW. The fragment assembly string graph. *Bioinformatics* 2005;**21 (Suppl 2)**:ii79–85.
145. Lin HH, Liao YC. Evaluation and validation of assembling corrected pacbio long reads for microbial genome completion via hybrid approaches. *PLoS One* 2015;**10**:e0144305.
146. Dong Y, Xie M, Jiang Y, *et al*. Sequencing and automated whole-genome optical mapping of the genome of a domestic goat (Capra hircus). *Nat Biotechnol* 2013;**31**:135–41.
147. Burton JN, Adey A, Patwardhan RP, *et al*. Chromosome-scale scaffolding of *de novo* genome assemblies based on chromatin interactions. *Nat Biotechnol* 2013;**31**:1119–25.
148. Wj A. Next Generation DNA Sequencing (II): techniques, applications. *J Next Gener Seq Appl* 2015;**01**.
149. Arjmandi-Tash H, Belyaeva LA, Schneider GF. Single molecule detection with graphene and other two-dimensional materials: nanopores and beyond. *Chem Soc Rev* 2016;**45**:476–93.
150. Di Ventra M, Taniguchi M. Decoding DNA, RNA and peptides with quantum tunnelling. *Nat Nanotechnol* 2016;**11**:117–26.
151. Heerema SJ, Dekker C. Graphene nanodevices for DNA sequencing. *Nat Nanotechnol* 2016;**11**:127–36.
152. Cao MD, Nguyen SH, Ganesamoorthy D *et al*. Scaffolding and completing genome assemblies in real-time with nanopore sequencing. *bioRxiv* 2016:054783.
153. Medvedev P, Pham S, Chaisson M, *et al*. Paired de bruijn graphs: a novel approach for incorporating mate pair information into genome assemblers. *J Comput Biol* 2011;**18**:1625–34.
154. Marcais G, Kingsford C. A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. *Bioinformatics* 2011;**27**:764–70.
155. Kurtz S, Narechania A, Stein JC, *et al*. A new method to compute K-mer frequencies and its application to annotate large repetitive plant genomes. *BMC Genomics* 2008;**9**:517.