De Novo Genome Assembly of Next-Generation Sequencing Data

4

Min Liu, Dongyuan Liu and Hongkun Zheng

Abstract

With rapid development of next-generation sequencing (NGS) technologies, de novo genome assembly appears increasingly common. However, inherent features of NGS data pose great challenges for de novo genome assembly. Many genomes, such as *Brassica rapa*, having undergone three paleo-polyploidy events, contain high content repeats, makes genome assembly of NGS data tougher. In past several years, numerous algorithms have been developed to address the challenges in de novo genome assembly from NGS reads. Here we summarize the main approaches for genome assembly. We also describe several algorithms for each approach. In addition, we compare the performance of existing assemblers in the accuracy and contiguity of assemblies. The comparative analysis shows that there is not any assembler that performs best in all the observed measures, which are also dependent on the dataset used.

4.1 Introduction

Rapid development of next-generation sequencing (NGS) technologies has greatly reduced DNA sequencing costs and made genome assembly increasingly common. However, inherent features of NGS data also pose new challenges for de novo genome assembly. In the past several years, numerous algorithms have been developed to cope with the challenges in de novo genome assembly from NGS reads. Most adopt the de Bruijn graph approach (Li et al. 2012), where a vertex represents a unique length-k substring called k-mer, and an edge connects two vertices if they appear consecutively in a read (Compeau et al. 2011). A few use the overlap–layout–consensus (OLC) approach, such as Edena (Hernandez et al. 2008) and string graph assembler (SGA) (Simpson and Durbin 2012). There are also some extension-based algorithms available for NGS reads, which do extension from 5′ or 3′ terminal of read by k-mer or read, such as SSAKE (Warren et al. 2007) and JR-Assembler (Chu et al. 2013).

M. Liu · D. Liu · H. Zheng (✉)
Biomarker Technologies Corporation, Beijing 101300, China
e-mail: zhenghk@biomarker.com.cn

M. Liu
e-mail: lium@biomarker.com.cn

D. Liu
e-mail: liudy@biomarker.com.cn

X. Wang and C. Kole (eds.), *The Brassica rapa Genome*, Compendium of Plant Genomes,
DOI 10.1007/978-3-662-47901-8_4

NGS reads are very short and error-prone, compared with traditional Sanger sequencing. To assemble this new kind of sequencing data, several assemblers, represented by MaSuRCA (Zimin et al. 2013), firstly construct longer reads (super-read), then assembly super-reads into contigs (Butler et al. 2008; Gnerre et al. 2011; Zimin et al. 2013). The basic construction process of super-reads is to extend each original read forwards and backwards, base by base, as long as the extension is unique. All reads that extend to the same super-read are replaced by that super-read. This allows subsequent computation quick and thus reduces memory requirements. Different from MaSuRCA, allpaths-lg uses an overlapping paired-end library with a suitable insert size to generate super-reads for contig assembly (Butler et al. 2008; Gnerre et al. 2011). These types of super-reads allow assembler to use OLC strategies with a few representative reads or de Bruijn graph approach with big *k*-mer.

Despite considerable progress made in the past years, genome assembly remains challenging. For example, recent completion of *Brassica rapa* genome sequencing has revealed that there are high content of repeats in *B. rapa* accession chiifu-401-42, which make many mate-pair library can't span the two side of unique region. Hence, the assembly had to use 199,452 BAC-end Sanger sequences, which have very long insert size to construct the super scaffold. Despite these efforts, the N50 of the assembled genome only reaches 1.9 Mb, with 283.8 Mb of the total size (Wang et al. 2011), much smaller than the real genome size ($2n = 2x = 529$ Mb). Therefore, there is a great need to provide novel algorithms and assemblers for de novo genome assembly of NGS data.

4.2 The Challenge of Genome Assembly

The primary difficulty in genome assembly is to merge overlapping reads along continuous sequences. First, contigs that the assembly algorithms produce are not complete and do not cover the entire chromosomes, due to sequencing errors and the existence of unsequenced parts. Even with high coverage, there is still a nonzero probability for the existence of unsequenced parts and sequencing errors. Second, repeats and heterozygous sequences will further complicate the assembly.

In order to assemble a genome, we first need to sequence random DNA fragments from the whole genome. The rapidly decreasing costs of NGS allow us to rapidly obtain vast amounts of DNA sequence data at a low cost. Unfortunately, the sequence length of NGS data is much shorter than that of the genomes or genomic features being studied, which commonly spans tens of thousands to billions of base pairs. Hence, many analyses starting with the computational process of sequence assembly that joins together the many sequence fragments the NGS generates. The workflow of a typical assembly algorithm is shown in Fig. 4.1.

Second, assembly algorithm will merge sequence fragments into contigs. A sequence contig is a contiguous, overlapping sequence read, which is assembled with the small DNA fragments generated by bottom-up sequencing strategies. Contig assembly is difficult in the

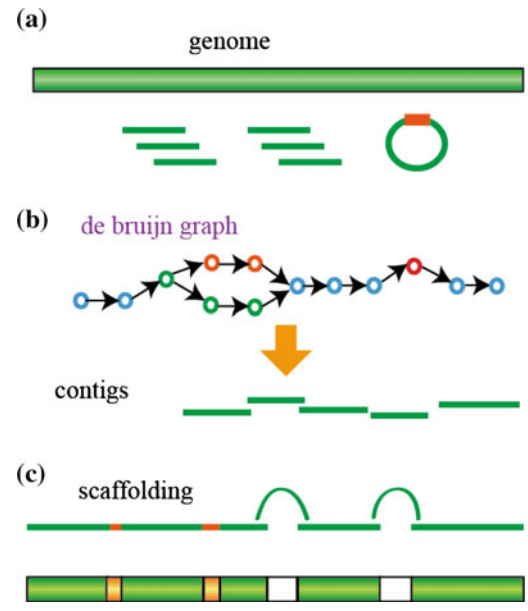

Fig. 4.1 The workflow of a typical assembly algorithm

process of genome assembly. Assemblers use the overlapping information of fragment to search contiguous paths. The sequence will be broken when faced with the branch, which may come from the sequencing error or repeats in the genome. Heterozygous sequences also can produce branches. For diploid species, there may be two paths for one single nucleotide polymprphism (SNP). For polyploidy, there may be many paths for different regions. Because of these enormous difficulties, contig assembly is important.

Third, contigs will be ordered to construct scaffold. Paired-end read libraries are useful in genome assembly. These data can help to extend contigs and resolve repeat areas. If one end of a paired-read is assembled in a contig and the other end in a second contig, it can be inferred that these contigs are adjacent in the final assembly. Because there may are erroneous links, assemblers need to filter out low weight links. For example, many assemblers only keep the links, which contain at least three or five paired-reads. There also exists the strategy, which firstly uses high weight links, and then makes use of low-weight links to form scaffolds. Recently, optical mapping is increasingly being used to order contigs or scaffolds.

Finally, gap closing will be used to fill the gaps in the scaffolds. After scaffolding, many assemblers will remap pair-end reads onto contigs and get linking information between them. The local un-assembly reads will be retrieved. Unaligned reads of the single aligned pair-end always can align multiple regions of genome. In small local regions, read overlapping information will be used to form sequences with much lenient standard.

4.3 De Novo Assembly Algorithm

4.3.1 Classification of De Novo Assembly Algorithm

Most of existing de novo assembly tools for NGS platforms utilize the de Bruijn graph approach (Li et al. 2012). In the de Bruijn graph, a vertex represents a unique length-*k* substring called *k*-mer, and an edge connects two vertices if they appear consecutively in a read (Compeau et al. 2011). There also exist some assemblers, which apply the overlap–layout–consensus (OLC) approach for handling NGS reads, such as Edena (Hernandez et al. 2008) and SGA (Simpson and Durbin 2012). Additionally, some extension-based assemblers also appeared to assemble NGS reads, which do extension from 5′ or 3′ terminal of read by *k*-mer or read, such as SSAKE (Warren et al. 2007) and JR-Assembler (Chu et al. 2013).

4.3.2 The Overlap Layout Consensus Approach

The sequence assembly problem can be taken as a graph problem by making an overlap graph of reads. In the overlap graph, reads are presented as nodes, and the existing overlap between two reads is presented as an edge between corresponding nodes. A modified version of the Smith-Waterman dynamic programming algorithm is usually used to find overlapping reads in almost all assemblers (Gnerre et al. 2011; Zimin et al. 2013).

In an overlap graph, assembling the reads into the genome is equivalent to finding a Hamiltonian path, a path that contains each node exactly once. Unfortunately, finding a Hamiltonian path is an NP-complete problem, which cannot be done in polynomial time.

Overlap layout consensus methods are based on graph theory. In these methods, an overlap graph is built from reads and the assembly problem is simplified to find a Hamiltonian path in the graph. ARACHNE (Metzker 2010), Celera (Li et al. 2012) and its revised version for short-reads (Peng et al. 2012), CAP3 (Jaillon et al. 2007), and Newbler (Zhang et al. 2012) use this method as their core idea.

An OLC algorithm starts by searching overlaps between reads (or graph nodes) (Fig. 4.2). In fact, it must check possible overlaps between any two reads in the input read set. The layout step will simplify the overlap graph by removing redundant information and will put these reads together using identified overlaps. The final step

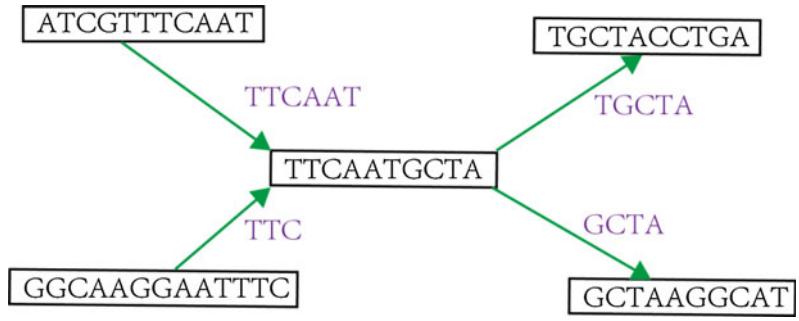

Fig. 4.2 An OLC assembly graph. Nodes are complete reads, and edges connect reads that overlap. Note that in an actual OLC assembly graph, reads and overlaps would be much larger. Here, theoretical reads and overlaps are shortened for clarity

is to find a consensus for the existing layout. The overlap step is computationally intensive. Therefore, this approach is more suitable for whole genome shotgun sequencing reads that Sanger sequencing technology produces. Similarly, the Hamiltonian path problem is an NP-complete problem in itself. It needs heuristic solutions.

4.3.3 De *Bruijn Graph Approach*

In 2001, Pevzner and Tang introduced a method based on the Eulerian path approach for assembling NGS reads (Pevzner and Tang 2001). In the new approach, reads are cut into smaller but regular pieces, called *k*-mers, which are then used to create a de Bruijn graph. By reducing the fragment assembly to a de Bruijn graph, the NP-complete Hamiltonian path is transformed to seek a Eulerian path in a de Bruijn graph. This approach avoids the complicated step of searching all overlaps between reads, which are required to form an overlap graph in the case of overlap layout consensus approach. There are polynomial time algorithms for finding Eulerian path problems. However, in practice, there may be several Eulerian paths in de Bruijn graphs. Finding the shortest Eulerian super path is still NP-hard (Zerbino and Birney 2008). Existing algorithms use heuristic methods to compute this super path by modifying the Eulerian graph. In addition, De Bruijn graph approach also simplifies the sequence repeat issue.

A *k*-dimensional de Bruijn graph is a directed graph whose nodes are all possible length-*k* sequences of *m* symbols. Obviously, each *k*-dimensional de Bruijn graph of *m* symbols has *mk* vertices. A de Bruijn graph is a representation based on all *k*-mers (length *k* words), which makes it suitable for high-coverage, very short-read data.

An edge in de Bruijn graphs connects two vertices (*k*-mers), if one vertices postfix of length $k - 1$ is equal to the prefix of the other one with the same length. The edge is directed, and the direction is from the *k*-mer, including the postfix to the *k*-mer including the prefix.

Given a sequence (GGATCGTTTCGTAAT), one can make a de Bruijn graph of it. To create a de Bruijn graph, it is enough to put the directed edges in the graph according to the sequence. The de Bruijn graph for this set is shown in Fig. 4.3.

In de Bruijn graph approach assembly algorithms, the graphs of input reads are created and then paths in graphs are used to detect contigs. Finding Eularian paths is the key to finding contigs in this step. Optionally, the algorithm may use other data, such as paired-end data, in order to make longer contigs and complete the assembly process. The need for predefined *k*-value, and also errors in reads that lead to a complex graph structure, are of issues in de Bruijn graph-based assembly algorithms.

The Euler assembler (Kent 2002) is the first algorithm that uses the de Bruijn approach for handling sequence assembly problems.

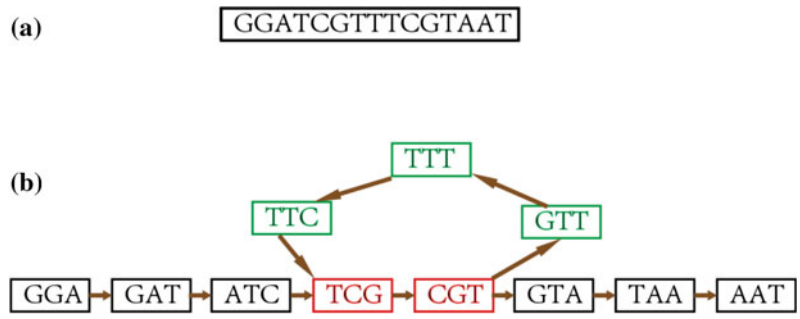

Fig. 4.3 The de Bruijn graph of an input set (GGATCGTTTCGTAAT)

Velvet (Zerbino and Birney 2008), Euler-USR (Chaisson et al. 2009), AllPaths (Butler et al. 2008; Maccallum et al. 2009), Abyss (Simpson et al. 2009), and IDBA (Peng et al. 2012) are some other assembly algorithms that use this approach.

4.3.4 *Extension*-Based Approach

The shotgun sequence assembly problem was first formalized by finding the shortest common superstring of the set of all reads (Delcher et al. 2002). Since this algorithm is computationally NP-complete, greedy approaches were introduced to solve the problem. The greedy approach uses a greedy idea, that is, to merge two reads with maximum overlap score at the time (Fig. 4.4). Reads and overlaps are considered to be nodes of graph and edges between nodes in a graph, respectively. Now the problem is simplified to find a Hamiltonian path in the graph.

Greedy algorithms for read assembly can be written in the following steps:

1. Calculate pairwise alignments of all fragments.
2. Choose two fragments with the largest overlap.
3. Merge chosen fragments.
4. Repeat steps 1, 2 and 3 until only one fragment is left.

The main problem of this approach is getting stuck in local maxima, as in the cases of all greedy algorithms. A local maxima can occur if the current contig takes on reads that would help further contigs grow even larger. Examples

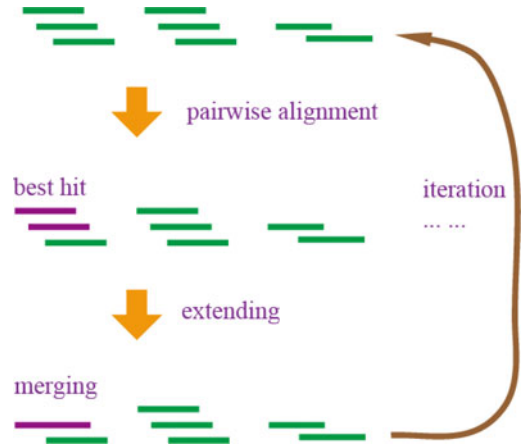

Fig. 4.4 The main steps in greedy algorithms for genome assembly

of algorithms using a greedy approach are PE-Assembler (Ariyaratne and Sung 2011), SSAKE (Warren et al. 2007), SHARCGS (Dohm et al. 2007), and VCAKE (Miller et al. 2010).

4.4 Comparison of Algorithms

4.4.1 Datasets and Assemblers

SPAdes 3.0 (Bankevich et al. 2012), MaSuRCA 2.2.1 (Zimin et al. 2013), SOAPdenovo2 (Li et al. 2010; Luo et al. 2012), and ALLPATHS-LG 44683 (Butler et al. 2008; Gnerre et al. 2011) were compared with nine bacterial data sets. ABySS 1.2.6 (Simpson et al. 2009), Edena 2.1.1 (Hernandez et al. 2008), SOAPdenovo 1.0.5,

Table 4.1 The basic information of the nine bacterial data sets of next generation sequence used for the assembly comparison

Species	Genome size (bp)	GC (%)	Library			Coverage (Gb)	SRA
			No.	Size	Type		
Acinetobacter baumannii NIPH24	3,893,975	39.16	1	180	PE	1.1	SRX236318
			2	5k	MP	1.1	SRX221053
Acinetobacter indicus CIP110367	3,211,639	45.34	1	180	PE	0.52	SRX342013
			2	180	PE	0.51	SRX342012
			3	5k	MP	0.71	SRX342014
			4	5k	MP	0.66	SRX342011
Enterobacter cloacae UCICRE12	5,210,535	55.59	1	180	PE	1.2	SRX342585
			2	5k	MP	1.5	SRX286723
Enterococcus faecium BM4538	3,133,897	38.07	1	180	PE	1.2	SRX341265
			2	5k	MP	1.8	SRX341264
Escherichia coli BIDMC 39	4,882,922	51.01	1	180	PE	0.33	SRX277757
			2	180	PE	0.57	SRX277758
			3	5k	MP	1.03	SRX277759
Klebsiella pneumoniae BIDMC41	5,702,446	26.95	1	180	PE	0.69	SRX277856
			2	180	PE	0.39	SRX277855
			3	5k	MP	1.18	SRX277857
Mucispirillum schaedleri ASF457	2,332,248	57.08	1	180	PE	1.2	SRX332194
			2	5k	MP	1.5	SRX332193
Pseudomonas aeruginosa CF614	6,797,445	31.01	1	180	PE	0.99	SRX366180
			2	5k	MP	0.81	SRX366181
			3	5k	MP	0.82	SRX366179
Streptococcus intermedius ATCC 27335	1,951,449	66.08	1	180	PE	1.1	SRX297066
			2	5k	MP	1.47	SRX297065

SOAPdenovo2 (Li et al. 2010; Luo et al. 2012), JR-Assembler 1.0 (Chu et al. 2013), and Velvet 1.0.19 (Zerbino and Birney 2008) were compared with median data sets.

The NGS datasets of *Streptomyces roseosporus*, *Neurospora crassa*, *Plasmodium falciparum*, and *Saprolegnia parasitica* genomes were downloaded from the National Center for Biotechnology Information (NCBI) Short Read Archive (SRA) (www.ncbi.nlm.nih.gov/sra) under accession numbers: SRX016044, SRX026747, SRX030834, SRX022535, SRX016057, and SRX016059. Another dataset covering nine bacterial genomes (*Staphylococcus aureus* and *Rhodobacter sphaeroides*) was also downloaded from NCBI SRA; the accession numbers are listed in Table 4.1. The *Escherichia coli* reference genome was retrieved from GenBank under accession no. NC_000913.

4.4.2 Performance Comparison Using Medium-Sized Genomes

The performance comparison of these assemblers was evaluated using four medium-sized dataset, including a bacterial genome (*S. roseosporus*, genome size 7.7 Mb) and three fungal genomes (*N. crassa*, *P. falciparum*, and *S. parasitica*, genome sizes 37.1, 22.9, and 53.1 Mb, respectively). The rank method was used to evaluate the assembler (Chu et al. 2013). No assembler outperformed other assemblers in total contig

Table 4.2 Assembly statistics of four median genomes

Species	Assembler	No. of contigs[a]	Total size (Mb)[b]	Max (bp)[c]	Mean (bp)	N50 (bp)[d]
Streptomyces roseosporus	JR-Assembler	1189	7.68	40,501	6461	11,374
	ABySS	1127	7.73	55,078	**6859**	**12,499**
	Velvet	1192	7.49	**61,423**	6286	11,075
	SOAPdenovo	2453	7.65	24,303	3120	4691
Neurospora crassa	JR-Assembler	12,244	38.61	**58,672**	3153	6074
	ABySS	13,420	38.05	45,381	2835	6350
	Velvet	10,187	36.11	45,599	**3544**	**6781**
	SOAPdenovo	16,261	40.25	31,423	2475	5029
	Edena	17,083	39.95	42,952	2338	4534
Saprolegnia parasitica	JR-Assembler	40,587	46.09	**119,543**	**1135**	**1510**
	ABySS	52,087	38.26	94,931	734	740
	Velvet	53,736	47.38	91,073	881	1021
	SOAPdenovo	66,456	45.59	30,400	686	712
	Edena	62,357	44.13	41,473	707	746
Plasmodium falciparum	JR-Assembler	13,352	11.02	7939	**825**	**975**
	ABySS	16,658	11.80	7934	708	826
	Velvet	16,423	11.91	**7940**	725	848
	SOAPdenovo	17,424	11.93	7939	684	786
	Edena	16,531	11.76	7936	711	831

The top two best values of each assembly metrics are marked in bold
[a]Contigs of length <300 bp were not counted
[b]"Total" refers to the total number of bases in the contigs
[c]"Max" and "Mean" refer to the length of the longest contig and the mean length of contigs, respectively
[d]N50 is the size of the smallest contig such that 50 % of the assembled bases are in the contigs of size equal to or larger than the N50 value
A contig is misassembled if it cannot be aligned in full-length to the reference genome

number, total contig size, the maximal contig length, the mean contig length, or N50 length (Table 4.2). With *S. parasitica and P. falciparum*, the N50 lengths and mean contig length were longer than other assemblers. With *S. roseosporus* and *N. crassa,* ABySS and velvet exhibit a relatively good performance, respectively.

4.4.3 Performance Comparison Using Nine Bacterial Genomes

The performance of four commonly-used assemblers was evaluated using nine genome datasets with high coverage. The raw sequence data were derived from a strain for which the assembly level is either scaffolds or contigs. Every raw datum contains at least two libraries: one paired ends library and one mate-pair library (Table 4.1).

The N50 contig sizes are summarized for all nine of these datasets in Table 4.3. For all assemblers, good or nearly-good assembly can be obtained. All data sets except the *Mucispirillum schaedleri* dataset were able to produce a high-contig N50 from 200 to 500 kb. These results are better than those produced by datasets, which produced only one-paired ends library (Salzberg et al. 2012; Magoc et al. 2013). Because of the mate-pair library, better scaffold N50s were also produced by most datasets. The best scaffold N50s ranged from 1.4 to 5.7 Mb in size, which span more than the half of the

Table 4.3 Assembly statistics of nine bacterial genomes

Species	Assembler	Total length	No. scaffolds	Scaffold N50	No. contigs	Contig N50
Acinetobacter baumannii NIPH 24	allpaths-lg	3,899,709	18	2,378,052	35	343,910
	Soapdenovo2	3,881,660	20	2,379,771	30	**586,913**
	SPAdes	4,420,053	68	538,328	68	538,328
	MaSuRCA	4,051,564	46	**2,414,221**	60	438,417
Acinetobacter indicus CIP 110367	allpaths-lg	3,188,830	8	**2,659,306**	46	130,481
	Soapdenovo2	3,192,750	68	**1,741,014**	129	133,591
	SPAdes	3,178,658	38	266,989	38	**266,989**
	MaSuRCA	3,061,054	28	914,711	65	133,907
Enterobacter cloacae UCICRE 12	allpaths-lg	5,167,463	24	**2,910,535**	83	152,889
	Soapdenovo2	5,167,151	55	2,892,397	132	154,254
	SPAdes	5,663,090	70	247,654	70	**247,654**
	MaSuRCA	5,141,681	48	**4,489,688**	102	227,675
Enterococcus faecium BM4538	allpaths-lg	3,131,274	7	954,529	49	126,411
	Soapdenovo2	3,068,531	66	767,650	156	87,971
	SPAdes	3,431,198	58	266,407	58	**266,407**
	MaSuRCA	3,165,896	102	**2,119,662**	168	85,115
Escherichia coli BIDMC 39	allpaths-lg	4,905,456	25	2,678,791	110	121,904
	Soapdenovo2	4,903,160	94	2,497,784	203	216,794
	SPAdes	4,902,401	65	284,858	65	**284,858**
	MaSuRCA	4,842,876	36	**3,718,131**	78	262,190
Klebsiella pneumoniae BIDMC 41	allpaths-lg	5,661,146	9	4,151,878	59	187,007
	Soapdenovo2	5,702,239	46	1,971,292	105	240,104
	SPAdes	5,751,074	34	813,379	35	**813,379**
	MaSuRCA	5,714,443	37	**4,562,366**	84	299,706
Mucispirillum schaedleri ASF457	allpaths-lg	2,311,286	20	741,189	82	60,693
	Soapdenovo2	2,337,314	62	594,782	136	68,178
	SPAdes	2,348,799	71	149,716	72	**149,716**
	MaSuRCA	2,335,222	62	**1,891,207**	106	84,260
Pseudomonas aeruginosa CF614	allpaths-lg	6,807,352	9	1,431,211	25	429,413
	Soapdenovo2	6,818,856	51	**5,774,020**	112	378,571
	SPAdes	6,751,614	31	965,679	31	**965,679**
	MaSuRCA	6,797,296	21	1,933,268	39	**689,346**
Streptococcus intermedius ATCC 27335	allpaths-lg	1,892,452	10	634,497	16	**284,930**
	Soapdenovo2	1,929,122	12	**910,762**	19	260,557
	SPAdes	1,918,222	11	277,339	11	**277,339**
	MaSuRCA	2,017,461	60	548,293	65	239,440

*The top two best values of each assembly metrics are marked in bold

Table 4.4 Assembly accuracy of assemblers evaluated by using REAPR

Species	Rank (FCD gap)		
	Allpaths-lg	MaSuRCA	SOAPdenovo2
Acinetobacter baumannii NIPH 24	3(4)	1(12)	3(4)
Acinetobacter indicus CIP 110367	1(9)	3(7)	2(8)
Enterobacter cloacae UCICRE 12	3(4)	1(27)	2(5)
Enterococcus faecium BM4538	3(4)	1(22)	2(20)
Escherichia coli BIDMC 39	2(16)	3(3)	1(25)
Klebsiella pneumoniae BIDMC 41	3(4)	1(17)	2(16)
Mucispirillum schaedleri ASF457	3(1)	1(22)	2(16)
Pseudomonas aeruginosa CF614	3(0)	2(1)	1(7)
Streptococcus intermedius ATCC 27335	3(1)	3(1)	1(2)
	24	16	16

genome. Results produced by all of the assemblers for *M. schaedleri* are far more fragmented than those of other datasets, with contig N50 sizes ranging from 60 to 149 kb. For this genome, the choice of assembler seems to have a large impact on the quality of the resulting assembly.

No assembler ranked highest among all metrics (Table 4.3). For this reason, a ranking approach was used to evaluate the overall performance of each assembler. For each assembly metric and dataset, the top two values are marked in bold (Table 4.3). The number of marked values was determined and used as the voting score for each assembler. For N50 length of scaffold, the scores for allpaths-lg (Gnerre et al. 2011), SOAPdenovo2 (Luo et al. 2012), SPAdes (Bankevich et al. 2012), and MaSuRCA (Zimin et al. 2013) were 26, 24, 9, and 31, respectively. In this way, MaSuRCA were found to have the best overall performance. For N50 length of contig, the scores were 15, 19, 34, and 22, respectively. In this way, SPAdes were found to have the best overall performance.

The accuracy of assembly was evaluated by REAPR (Table 4.4). REAPR uses the per-base error of the fragment coverage distribution (FCD) to detect assembly errors without the need for a reference sequence and provides corrected assembly statistics allowing the quantitative comparison of multiple assemblies (Hunt et al. 2013). For each data set, no assembler produced the lowest number of FCD gaps, and MaSuRCA was also found to produce highly accurate results in some datasets (*Acinetobacter indicus CIP 110367*, *E. coli BIDMC 39*, and *Streptococcus intermedius ATCC 27335*). Hence, a ranking approach was used to evaluate the overall performance in assembly accuracy: the higher the score, the more accurate the assembly (Chu et al. 2013). Because SPAdes only produced the contig assembly, it does not participate in the comparative analysis. Allpaths-lg produced the highest ranking score (24), which is much higher than the scores that MaSuRCA (16) and SOAPdenovo2 (16) produced.

4.5 Discussion

In this section, the main approaches for genome assembly are presented. For each approach, several algorithms are explained. There are three main categories for assembly algorithms: extension-based algorithms, overlap-layout consensus algorithms and de Bruijn graph algorithms. Overlap-layout consensus algorithms are based on overlap graphs and Hamiltonian path-finding. de Bruijn graph algorithms are based on de Bruijn graphs and Eulerian path-finding in assembly graphs. de Bruijn graph

methods show more strength for short-reads and in resolving repeats. Overlap graph methods are more suitable for Sanger shotgun data. While extension-based methods seemed applicable just on long sequences, some tricks used in new algorithms, which use paired-end reads, such as the PE-Assembler, could apply the greedy idea efficiently for short reads. The evaluation of performances of an assembly algorithm is bases on both the accuracy and contiguity of assemblies, and there is always a trade-off between different measures of assembly performances. It is not a trivial task to compare assembly algorithms.

In fact, the assembly results also depend on the dataset used, besides assembly algorithm. An algorithm may do well with a dataset but not with other datasets. For a new dataset, we cannot exactly predict which algorithm would produce a better assembly just based on previous assembly results, due to the difference in dataset used for assembly. In addition, some algorithms, such as parameter k in de Bruijn graph-based methods, require users to predefine assembly parameter. The use of parameters makes it more difficult to compare assembly algorithms, for the final assembly result is definitely dependent on the parameter chosen for the assembly task. There are some metrics available for comparing assembly algorithms, but the availability of a good metric that is not dependent on the reference genome is still missing from the literature.

References

Ariyaratne PN, Sung WK (2011) PE-assembler: de novo assembler using short paired-end reads. Bioinformatics 27:167–174

Bankevich A, Nurk S, Antipov D, Gurevich AA, Dvorkin M et al (2012) SPAdes: a new genome assembly algorithm and its applications to single-cell sequencing. J Comput Biol 19:455–477

Butler J, MacCallum I, Kleber M, Shlyakhter IA, Belmonte MK et al (2008) ALLPATHS: de novo assembly of whole-genome shotgun microreads. Genome Res 18:810–820

Chaisson MJ, Brinza D, Pevzner PA (2009) De novo fragment assembly with short mate-paired reads: does the read length matter? Genome Res 19:336–346

Chu TC, Lu CH, Liu T, Lee GC, Li WH, Shih AC (2013) Assembler for de novo assembly of large genomes. Proc Natl Acad Sci USA 110:E3417–E3424

Compeau PE, Pevzner PA, Tesler G (2011) How to apply de Bruijn graphs to genome assembly. Nat Biotechnol 29:987–991

Delcher AL, Phillippy A, Carlton J, Salzberg SL (2002) Fast algorithms for large-scale genome alignment and comparison. Nucleic Acids Res 30:2478–2483

Dohm JC, Lottaz C, Borodina T, Himmelbauer H (2007) SHARCGS, a fast and highly accurate short-read assembly algorithm for de novo genomic sequencing. Genome Res 17:1697–1706

Gnerre S, Maccallum I, Przybylski D, Ribeiro FJ, Burton JN et al (2011) High-quality draft assemblies of mammalian genomes from massively parallel sequence data. Proc Natl Acad Sci USA 108:1513–1518

Hernandez D, Francois P, Farinelli L, Osteras M, Schrenzel J (2008) De novo bacterial genome sequencing: millions of very short reads assembled on a desktop computer. Genome Res 18:802–809

Hunt M, Kikuchi T, Sanders M, Newbold C, Berriman M, Otto TD (2013) REAPR: a universal tool for genome assembly evaluation. Genome Biol 14:R47

Jaillon O, Aury JM, Noel B, Policriti A, Clepet C et al (2007) The grapevine genome sequence suggests ancestral hexaploidization in major angiosperm phyla. Nature 449:463–467

Kent WJ (2002) BLAT–the BLAST-like alignment tool. Genome Res 12:656–664

Li R, Zhu H, Ruan J, Qian W, Fang X et al (2010) De novo assembly of human genomes with massively parallel short read sequencing. Genome Res 20:265–272

Li Z, Chen Y, Mu D, Yuan J, Shi Y et al (2012) Comparison of the two major classes of assembly algorithms: overlap-layout-consensus and de-bruijn-graph. Brief Funct Genomics 11:25–37

Luo R, Liu B, Xie Y, Li Z, Huang W et al (2012) SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler. Gigascience 1:18

Maccallum I, Przybylski D, Gnerre S, Burton J, Shlyakhter I et al (2009) ALLPATHS 2: small genomes assembled accurately and with high continuity from short paired reads. Genome Biol 10:R103

Magoc T, Pabinger S, Canzar S, Liu X, Su Q et al (2013) GAGE-B: an evaluation of genome assemblers for bacterial organisms. Bioinformatics 29:1718–1725

Metzker ML (2010) Sequencing technologies—the next generation. Nat Rev Genet 11:31–46

Miller JR, Koren S, Sutton G (2010) Assembly algorithms for next-generation sequencing data. Genomics 95:315–327

Peng Y, Leung HC, Yiu SM, Chin FY (2012) IDBA-UD: a de novo assembler for single-cell and metagenomic sequencing data with highly uneven depth. Bioinformatics 28:1420–1428

Pevzner PA, Tang H (2001) Fragment assembly with double-barreled data. Bioinformatics 17:S225–S233

Salzberg SL, Phillippy AM, Zimin A, Puiu D, Magoc T et al (2012) GAGE: a critical evaluation of genome assemblies and assembly algorithms. Genome Res 22:557–567

Simpson JT, Durbin R (2012) Efficient de novo assembly of large genomes using compressed data structures. Genome Res 22:549–556

Simpson JT, Wong K, Jackman SD, Schein JE, Jones SJ, Birol I (2009) ABySS: a parallel assembler for short read sequence data. Genome Res 19:1117–1123

Wang X, Wang H, Wang J, Sun R, Wu J et al (2011) The genome of the mesopolyploid crop species *Brassica rapa*. Nat Genet 43:1035–1039

Warren RL, Sutton GG, Jones SJ, Holt RA (2007) Assembling millions of short DNA sequences using SSAKE. Bioinformatics 23:500–501

Zerbino DR, Birney E (2008) Velvet: algorithms for de novo short read assembly using de Bruijn graphs. Genome Res 18:821–829

Zhang T, Luo Y, Chen Y, Li X, Yu J (2012) BIGrat: a repeat resolver for pyrosequencing-based re-sequencing with Newbler. BMC Res Notes 5:567

Zimin AV, Marcais G, Puiu D, Roberts M, Salzberg SL, Yorke JA (2013) The MaSuRCA genome assembler. Bioinformatics 29:2669–2677