Chapter 8
Next-Generation Sequence Assembly Overview

Abstract Next-generation sequence assembly can be viewed as a five-stage process of data processing and computational challenges. These stages are error correction, graph construction, graph simplification, scaffolding, and the assembly assessment stage. These stages communicate with each other to produce the final assembled sequences. Each stage receives a set of inputs from the preceding one and passes its output to the following stage. In this chapter, we will briefly introduce the basic functions of each stage and provide a coherent framework of the communications that occur between them.

8.1 Introduction to Next-Generation Sequence Assembly

The sequence assembly process was developed to resolve the limitations of current technologies that prevent the sequencing of the whole genome/chromosome during a single read. In first- and next-generation sequencing methods (see Chap. 3), the whole genome is sheared into short random fragments with short overlaps. Each fragment is sequenced independently and the resulting sequences are individually called a "read". Hence, the process of repositioning these random reads to reconstruct the whole genome is known as the "sequence assembly process" [1, 2].

According to the sample and type of raw data generated by sequencing instruments and the aim of the study, the assembly process may take many flavors including genome, transcriptome, or metagenome sequence assembly. If the raw data in the sequencing experiment is genomic DNA, the process is called genome assembly. Likewise, if the raw data is mRNA, the process is called transcriptome assembly, whereas assembling reads resulting from sequencing environmental samples that contain a mixture of organisms is called metagenome assembly. The ever-increasing number of applications in genomics, transcriptomics, metagenomics, and single-cell sequencing exhibits the need to acquire sequences from the viral, microbial, bacterial, or eukaryotic communities [3]. While the details of the assembly process and employed assembly tools are different in each case, the sequence assembly process always shares the same stages.

The process of sequence assembly starts with filtering the reads to remove or correct errors and then computing a set of overlaps among them to discover their arrangement. These overlaps are used to connect the reads together into long contiguous structures called "contigs". Similarly, contigs can also be connected together to form even longer sequence stretches called "scaffolds" [4].

According to the availability of the reference sequences, the sequence assembly process has two main approaches, comparative sequence assembly and de novo sequence assembly. In comparative sequence assembly (also known as reference-based sequence assembly), reference sequences from the same organism or closely related species help to guide the reconstruction process [5]. On the other hand, de novo assembly does not involve reference sequences and consequently is a more complicated process [1].

8.2 Sequence Assembly Framework

Sequence assembly is a multiphase process. These phases communicate together in order to produce the final assembled sequence. Not only does the organization of these phases differ from one assembly to another, but some phases are completely missing in certain assembly processes in accordance with various issues (Fig. 8.1) [6].

The first phase, commonly known as the error correction phase, aims at filtering erroneous reads by removing or correcting sequencing errors. The filtered reads are

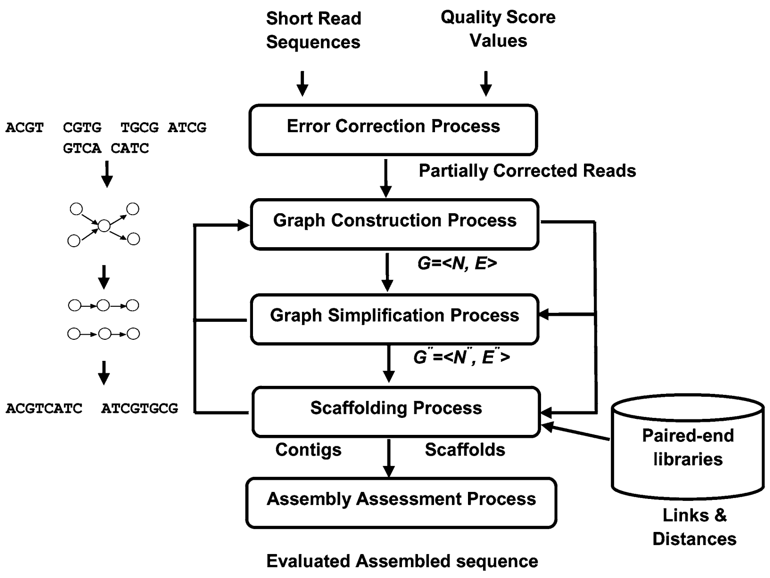

Fig. 8.1 Schematic representation of the five stages of next-generation sequence assembly process (*Note*: *G''* is a repairing version of graph *G* with *N* nodes and *E* edges)

then fed into the second phase that formulates them into a graph of nodes with their relationships represented as graph edges. This representation overcomes the limitation of available computational resources that are necessary to manage the high throughput nature of next-generation sequencers. However, the resulting graph may contain erroneous nodes or structures that were overlooked during the first phase. Hence, these erroneous structures must be removed or resolved, in the so called graph simplification phase, before the construction of the contigs. Following the graph simplification phase, the contigs are produced by finding the paths on the graph that connect the reads together. Subsequently, the scaffolding phase involves the filtering of the contigs, the detection of misassembled contigs and uncovering the relationships between them to build scaffolds [6]. Finally, the assembly assessment phase evaluates the assembled contigs/scaffolds in accordance with different metrics that reflect the quality, consistency, and accuracy of the algorithm used in the reconstruction process [7, 8].

There are many differing viewpoints when designing an assembler. Some designers rely on the early correction of errors in order to facilitate the remaining phases of the assembly process (i.e., graph building and simplification) [9–15]. Other designers propose to delay the error correction phase to the graph simplification process since both these phases aim at removing errors. Moreover, merging these two phases would reduce the overall computation time [16–22]. Hence, there are stand-alone error correction tools, scaffolding tools, and assessment tools that perform these phases independently from the other assembly phases. Certain designers rely on these independent tools to complete the missing parts in their assemblers.

8.2.1 Error Correction Phase

Correcting the errors that result from sequencing platforms represents one of the major challenges in the next-generation environment. These errors vary from the presence of simple ambiguous bases to the occurrence of substitution and indel errors (see Chap. 4). By detecting these errors early, the assembly process can be more efficient during the latter stages. The general approach followed by most error correction algorithms is examining the richness of the reads (i.e., read coverage) produced by the next-generation sequencers as a key to distinguish between correct and incorrect reads. This approach can be disrupted by repeats and non-uniform sampling of genomic sequences, which can lead to ambiguous choices during error correction [23].

8.2.2 Graph Construction Phase

There are diverse paradigms for graph construction in accordance with different graph models. These paradigms must overcome a host of computational challenges in relation to graph representation and path-finding algorithms for the contigs building (algorithms and challenges are discussed in detail in Chap. 9). Paradigms can generally be categorized into four main categories: overlap-based construction, *k*-mers-based construction, greedy-based construction, and hybrid-based construction [24, 25]. Each of these paradigms and their accompanying challenges are discussed in more detail in Chap. 9 as well.

8.2.3 Graph Simplification Phase

As mentioned previously, some errors are not recognized during the error correction phase and can subsequently complicate the efforts of path-finding algorithms that attempt to connect reads and assemble accurate contigs. These errors form diverse structures in the assembly graph which must be filtered through identification and correction before the building of contigs is initiated.

8.2.4 Scaffolding Phase

The process of creating scaffolds is not as simple as the process of creating contigs. The goal of the scaffolding process is to order and orient contigs that result from the assembly process. The scaffolding process is guided by paired-end reads that filter contigs, detect misassembled ones, and allow accurate contig extension into the repeated regions [6, 26].

8.2.5 Assembly Assessment Phase

Assessing the performance of an assembler is dependent on the metric(s) used during the evaluation process. One of these approaches targets the contiguity of the resulting contigs/scaffolds and utilizes different statistical metrics to assess the final assembled sequence [27–34]. Another approach scrutinizes the accuracy of the assembled contigs/scaffolds and uses one of the previously finished genomes as a reference to assess the draft sequence [29, 31]. Additional evaluative strategies include examining the constraints imposed by paired-end libraries, the nature of the sequences being assembled and the sequencing experiments themselves [31, 35, 36].

Since the assembler is a software program with a set of functionalities, it must be assessed not only in terms of its output but also in relation to other factors. These include responsiveness to user commands, the friendliness of the user interface components, and setup requirements. The evaluation of such functionalities allows the targeted assessment of the usability features of an assembler [37–39].

References

1. Pop M (2009) Genome assembly reborn: recent computational challenges. Briefings in bioinformatics 10 (4):354-366. doi:10.1093/bib/bbp026
2. Alkan C, Sajjadian S, Eichler EE (2011) Limitations of next-generation genome sequence assembly. Nat Methods 8 (1):61-65. doi:10.1038/nmeth.1527
3. Nagarajan N, Pop M (2013) Sequence assembly demystified. Nat Rev Genet 14 (3):157-167. doi:10.1038/nrg3367
4. Miller JR, Koren S, Sutton G (2010) Assembly algorithms for next-generation sequencing data. Genomics 95 (6):315-327. doi:10.1016/j.ygeno.2010.03.001
5. Pop M, Phillippy A, Delcher AL, Salzberg SL (2004) Comparative genome assembly. Briefings in bioinformatics 5 (3):237-248
6. El-Metwally S, Hamza T, Zakaria M, Helmy M (2013) Next-generation sequence assembly: four stages of data processing and computational challenges. PLoS Comput Biol 9 (12):e1003345. doi:10.1371/journal.pcbi.1003345
7. Earl D, Bradnam K, St John J, Darling A, Lin D et al. (2011) Assemblathon 1: a competitive assessment of de novo short read assembly methods. Genome research 21 (12):2224-2241. doi:10.1101/gr.126599.111
8. Bradnam KR, Fass JN, Alexandrov A, Baranay P, Bechner M et al. (2013) Assemblathon 2: evaluating de novo methods of genome assembly in three vertebrate species. Gigascience 2 (1):10. doi:2047-217X-2-10
9. Ilie L, Fazayeli F, Ilie S (2011) HiTEC: accurate error correction in high-throughput sequencing data. Bioinformatics 27 (3):295-302. doi:10.1093/bioinformatics/btq653
10. Kao WC, Chan AH, Song YS (2011) ECHO: a reference-free short-read error correction algorithm. Genome research 21 (7):1181-1192. doi:10.1101/gr.111351.110
11. Kelley DR, Schatz MC, Salzberg SL (2010) Quake: quality-aware detection and correction of sequencing errors. Genome Biol 11 (11):R116. doi:10.1186/gb-2010-11-11-r116
12. Medvedev P, Scott E, Kakaradov B, Pevzner P (2011) Error correction of high-throughput sequencing datasets with non-uniform coverage. Bioinformatics 27 (13):i137-i141. doi:10.1093/bioinformatics/btr208
13. Salmela L, Schroder J (2011) Correcting errors in short reads by multiple alignments. Bioinformatics 27 (11):1455-1461. doi:10.1093/bioinformatics/btr170
14. Schroder J, Schroder H, Puglisi SJ, Sinha R, Schmidt B (2009) SHREC: a short-read error correction method. Bioinformatics 25 (17):2157-2163. doi:10.1093/bioinformatics/btp379
15. Yang X, Dorman KS, Aluru S (2010) Reptile: representative tiling for short read error correction. Bioinformatics 26 (20):2526-2533. doi:10.1093/bioinformatics/btq468
16. Boetzer M, Henkel CV, Jansen HJ, Butler D, Pirovano W (2011) Scaffolding pre-assembled contigs using SSPACE. Bioinformatics 24 (4):578-579
17. Dayarian A, Michael TP, Sengupta AM (2010) SOPRA: Scaffolding algorithm for paired reads via statistical optimization. BMC bioinformatics 11:345. doi:10.1186/1471-2105-11-345
18. Donmez N, Brudno M (2013) SCARPA: scaffolding reads with practical algorithms. Bioinformatics 29 (4):428-434. doi:10.1093/bioinformatics/bts716
19. Gao S, Sung WK, Nagarajan N (2011) Opera: reconstructing optimal genomic scaffolds with high-throughput paired-end sequences. J Comput Biol 18 (11):1681-1691. doi:10.1089/cmb.2011.0170
20. Gritsenko AA, Nijkamp JF, Reinders MJ, de Ridder D (2012) GRASS: a generic algorithm for scaffolding next-generation sequencing assemblies. Bioinformatics 28 (11):1429-1437. doi:10.1093/bioinformatics/bts175
21. Koren S, Treangen TJ, Pop M (2011) Bambus 2: scaffolding metagenomes. Bioinformatics 27 (21):2964-2971. doi:10.1093/bioinformatics/btr520
22. Salmela L, Makinen V, Valimaki N, Ylinen J, Ukkonen E (2011) Fast scaffolding with small independent mixed integer programs. Bioinformatics 27 (23):3259-3265. doi:10.1093/bioinformatics/btr562

23. Yang X, Chockalingam SP, Aluru S (2013) A survey of error-correction methods for next-generation sequencing. Briefings in bioinformatics 14 (1):56-66. doi:10.1093/bib/bbs015
24. Medvedev P, Brudno M (2009) Maximum likelihood genome assembly. J Comput Biol 16 (8):1101-1116. doi:10.1089/cmb.2009.0047
25. Medvedev P, Georgiou K, Myers G, Brudno M (2007) Computability of Models for Sequence Assembly. In: Giancarlo R, Hannenhalli S (eds) Algorithms in Bioinformatics, vol 4645. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp 289-301. doi:10.1007/978-3-540-74126-8_27
26. Chaisson MJ, Brinza D, Pevzner PA (2009) De novo fragment assembly with short mate-paired reads: Does the read length matter? Genome research 19 (2):336-346. doi:10.1101/gr.079053.108
27. Church DM, Goodstadt L, Hillier LW, Zody MC, Goldstein S et al. (2009) Lineage-specific biology revealed by a finished genome assembly of the mouse. PLoS Biol 7 (5):e1000112. doi:10.1371/journal.pbio.1000112
28. Colbourne JK, Pfrender ME, Gilbert D, Thomas WK, Tucker A et al. (2011) The ecoresponsive genome of Daphnia pulex. Science 331 (6017):555-561. doi:10.1126/science.1197761
29. Li R, Fan W, Tian G, Zhu H, He L et al. (2010) The sequence and de novo assembly of the giant panda genome. Nature 463 (7279):311-317. doi:10.1038/nature08696
30. Lin Y, Li J, Shen H, Zhang L, Papasian CJ et al. (2011) Comparative studies of de novo assembly tools for next-generation sequencing technologies. Bioinformatics 27 (15):2031-2037. doi:10.1093/bioinformatics/btr319
31. Lindblad-Toh K, Wade CM, Mikkelsen TS, Karlsson EK, Jaffe DB et al. (2005) Genome sequence, comparative analysis and haplotype structure of the domestic dog. Nature 438 (7069):803-819. doi:nature04338
32. Liu Y, Qin X, Song XZ, Jiang H, Shen Y et al. (2009) Bos taurus genome assembly. BMC genomics 10:180. doi:10.1186/1471-2164-10-180
33. Locke DP, Hillier LW, Warren WC, Worley KC, Nazareth LV et al. (2011) Comparative and demographic analysis of orang-utan genomes. Nature 469 (7331):529-533. doi:10.1038/nature09687
34. Ming R, Hou S, Feng Y, Yu Q, Dionne-Laporte A et al. (2008) The draft genome of the transgenic tropical fruit tree papaya (Carica papaya Linnaeus). Nature 452 (7190):991-996. doi:10.1038/nature06856
35. Huson DH, Halpern AL, Lai Z, Myers EW, Reinert K et al. Comparing Assemblies Using Fragments and Mate-Pairs. In: WABI ‘01 Proceedings of the First International Workshop on Algorithms in Bioinformatics Århus, Denmark, 2001. Springer Berlin Heidelberg, pp 294-306
36. Phillippy AM, Schatz MC, Pop M (2008) Genome assembly forensics: finding the elusive mis-assembly. Genome Biol 9 (3):R55. doi:10.1186/gb-2008-9-3-r55
37. Golovko G, Khanipov K, Rojas M, Martinez-Alcantara A, Howard JJ et al. (2012) Slim-Filter: an interactive windows-based application for illumina genome analyzer data assessment and manipulation. BMC bioinformatics 13:166. doi:10.1186/1471-2105-13-166
38. Powell DR, Seemann T (2013) VAGUE: a graphical user interface for the Velvet assembler. Bioinformatics 29 (2):264-265. doi:10.1093/bioinformatics/bts664
39. Zhang W, Chen J, Yang Y, Tang Y, Shang J et al. (2011) A practical comparison of de novo genome assembly software tools for next-generation sequencing technologies. PLoS One 6 (3):e17915. doi:10.1371/journal.pone.0017915