Whole-Genome Assembly: An Experimental Study of Computational Costs and Architectural Opportunities

Elena Espinosa[1] , Ivan Fernandez[12] , Rafael Larrosa[13] and Oscar Plata[1]

Resumen— **Whole-genome sequencing (WGS) provides a huge amount of reads from which a complete genome could be assembled. The recent advent of long read sequencing technologies, such as PacBio and Oxford Nanopore, and the subsequent appearance of high quality long reads (single molecule high-fidelity, or HiFi) have improved the scaffolding of the genome. However, both biology and computing communities still face great challenges in terms of computational cost. Thus, it is essential a high precision characterization of the methods for a correct identification of the main computing bottlenecks. This study will allow us to design new methods to mitigate computational costs without losing accuracy and to adapt such methods to fully exploit new architectures that provide support to handle big amounts of data. In this paper, we experimentally study and characterize the most used whole-genome assemblers in order to design new approaches in this field.**

Palabras clave—**long reads, computational resources, near-memory processing, genome assembly**

I. Introduction

RE cently, we have witnessed great advances in the analysis of complete genomes and transcriptomes thanks to high-throughput sequencing. Next Generation Sequencing (NGS) platforms can deliver data output from multiple terabases in a single run, both at low cost and relatively low time consumption compared to the traditional Sanger Sequencing. For example, the first human genome draft, based on Sanger Sequencing technology, had a cost of $3 billion and took more than 10 years to complete. *Illumina*[1] NGS platforms, on the contrary, can sequence thousands to tens of thousands of genomes in one year with a precision usually better than 99 % [1]. However, as *Illumina* platforms provide reads up to 2x300 bp and the genome could present structural variations from 1 KB to 3 MB, this technology faces assembly obstacles.

Modern massive generation sequencing technologies such as PacBio[2] or Oxford Nanopore[3], which provide millions of reads of more than 20 KB, have enabled great advances in bioinformatics [2], specially in *de novo* genome assembly. However, Oxford Nanopore still faces an error rate of about 15 % regarding PacBio, which shows an error rate of 0.2 % thanks to the PacBio HiFi reads [3].

Such technologies have led to the development of assemblers for long reads and have created a niche for the design of new algorithms and new computer architectures that accelerate them. Nevertheless, both biology and computing communities still face great challenges in terms of computational cost for the assembly process. This fact becomes specially relevant when the genome reconstruction implies long repetitions or relies on a large heterozygosity. While state-of-the-art software solutions try to overcome this computational cost (e.g., *Canu* [4], [5], [6]), assembling large genomes still faces great challenges.

Traditional assemblers such as *Canu* and its predecessor, *Celera*, have been widely used by the scientific community, but they are computationally very expensive, which has made its use marginal. For example, *Hicanu*, added option to *Canu* for high-fidelity long reads (PacBio HiFi reads), can take over 1,000 CPU hours to assembly the human haploid cell line CHM13.

Many assemblers with new strategies have been emerging in the last decade to mitigate the computational cost problem. For example, *Hifiasm* was developed by PacBio for assembling HiFi reads. Also, Oxford Nanopore developed *Shasta* for Nanopore reads.

Despite these efforts, the balance between computational cost and quality of results remains a challenge. In this paper: (1) We have evaluated computational advances from traditional assemblers, such as *Canu*, to novel assemblers, such as *Hifiasm*, and (2) We have identified the main computing bottlenecks of *Hifiasm* whose resolution may represent an increase in the use of this technique.

II. Background

A. De Novo Assembly

De novo whole-genome assembly consists of the assembly of a large jigsaw puzzle where each piece is a nucleotide sequence of the genome. In this way, we should join each substring or contig to complete a chromosome by searching for overlapping regions between them.

De novo assembly avoids any bias that might have been introduced due to phylogenetic divergence, even genetic diversity. The 1000 Genome Project[4], the 10k

[1]Dept. of Computer Architecture, University of Malaga, e-mail: `{emesga,ivanfv,rlarrosa,oplata}@uma.es`

[2]Barcelona Supercomputing Center, Universitat Politecnica de Catalunya.

[3]Supercomputing and Bioinnovation Center, University of Malaga.

[1]https://www.illumina.com/

[2]https://www.pacb.com/

[3]https://nanoporetech.com/

[4]http://www.1000genomes.org/

UK Genome Project[5], the International Cancer Genome Consortium[6] and the 1001 Arabidopsis Genome Project[7] have successfully evidenced the genetic variety among individuals and cells by identifying single-nucleotide and structural variations.

However, *de novo* assembly has to overcome great challenges. First, it entails considerable time and computational resources. Second, high quality genome assembly still faces issues from biological perspective when: (1) the genome reconstruction relies on large heterozygosity, (2) the genome reconstruction implies nonrandom repeat elements as long interspersed nuclear elements (LINEs), short interspersed nuclear elements (SINEs), long terminal repeats (LTSs) and simple tandem repeats (STRs), and (3) the genome corresponds to polyploid organisms.

As a consequence, the assembly of long and complex genomes is unaffordable.

B. *Genome Assembly Pipeline*

The genome assembly process for long reads comprises three main tasks: (1) overlapping regions detection between reads, (2) correction of sequencing errors, and (3) contig construction and consensus. In general, we can define two assembly strategies: (1) overlap-layout-consensus (OLC), and (2) de-bruijn-graph (DBG). Figure 1 shows the general pipeline of both strategies.

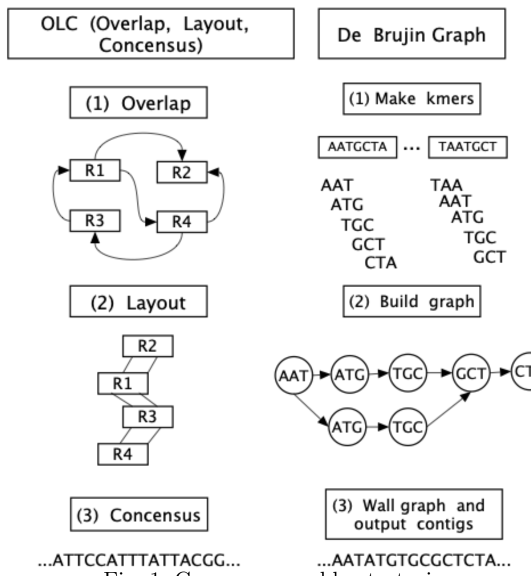

Fig. 1: Genome assembly strategies

The *OLC* approach compute alignment between reads to identify overlaps. It consists of three steps. First, in the overlap step, the algorithm compute overlap between all sequencing reads. Second, in layout step, the reads are layout into the most probable contiguous sequence stretches. Third, in the final step, the consensus sequence is determined for each contig by choosing the nucleotide, which is represented by the majority of the overlapping reads for every sequence position.

[5]http://www.uk10k.org/
[6]http://icgc.org/
[7]http://1001genomes.org/

Assemblers such as *Celera* use the OLC approach in their first versions. The overlap between the reads is computed using *BLASR* [7], calculating a suffix array index on the reference sequence to obtain the alignment between the target genome and the reference one. Later, it extracts a nonredundant graph (layout) and, finally, it merges the reads while correcting the sequencing errors (consensus).

The *DBG* approach extract k-mers to identify overlapping reads and later build connections between all k-mers, which differ in (k-1) of their bases.

String graph approach preserve the same properties and advantages as a de *Bruijn graph*, but, in this case, the nodes represents the overlap between the reads and the edge the read.

FALCON implements a typical *String graph* assembly pipeline as a modified version of *Gene Myers DALIGNER* for detecting overlaps and later building the graph. Another example is the novel assembler *Hifiasm*, which computes all performs all-versus-all read overlap alignment and later build a *String graph*. *Canu*, predecessor of *Celera* assembler implements *MinHash* algorithm which uses kmers to estimate the distance between two sequences, and later assembling reads into contigs using a modified version of the *String graph* algorithm.

C. *Data Intensive Computing*

State-of-the-art computing implementations of NGS algorithms are frequently focused on exploiting multicore platforms by means of parallel models such as MapReduce, MPI, and multi-threading.

Bioinformatics applications are often characterized by a high number of memory accesses and low operational cost, which usually results in memory bandwidth being the main bottleneck. However, current solutions do not usually face this problem. In addition, it is typical for memory access patterns to be randomized, resulting in an poor cache utilization. It gets worse with large data sets (e.g., large and complex genomes). As a consequence, bioinformatics applications greatly overwhelm the data storage and memory resources of a modern computer. Architectures based on near-memory processing seek to mitigate the data access latency, bandwidth and power consumption due to data movement. In this sense, the usage of architectures based on high-bandwidth memory and computation close to the data may improve the performance of bioinformatics pipelines.

III. Methodology

A. *Data Acquisition and Experimental Overview*

Genome sequencing data were downloaded from *Escherichia coli* and four eukaryotic organisms, *Arabidopsis thaliana*, *Saccharomyces cerevisiae*, *Drosophila melanogaster* and *Human*.

The genomes of those organisms were assembled using HiFi reads from the NCBI repository, in particular: *Escherichia coli*[8], *Arabidopsis thaliana*[9], *Saccharomyces cerevisiae*[10], *Drosophila melanogaster*[11] and two different human celular lines, haploid (*CHM13*[12]) and diploid (*HG002*[13]).

B. *Computational Resources*

We conducted the experiments in the *Picasso* cluster (*Supercomputing and Bioinnovation Center*, Malaga Techpark). We run the assemblers in Bull R282-Z90 nodes, where each one includes two 64-core AMD EPYC 7742 processors and 2 TB of RAM memory.

In addition, we also performed experiments in a server with two 18-core Intel Xeon Gold 6154 processors, with a total of 36 cores (72 hardware threads with hyperthreading), and 768 GB DDR4 memory.

C. *Pipeline Characterization*

We have profiled the assemblers using *Intel Advisor* and *VTune Profiler* from Intel oneAPI Base Toolkit. We have also used *GPROF* and *Perf* performance analysis tools.

C.1 Flops and Memops

We have measured the number of memory operations (loads and stores) and integer and floating point operations, calculating the arithmetic intensity. We have also obtained the cache miss/hit rates in order to characterize memory access patterns and determine the efficiency of the cache hierarchy.

C.2 Processor Performance

We have monitored the CPU workload during the execution of the program, identifying the CPU workload peaks. The thread count have been obtained to quantify the amount of parallelism and determine the performance in terms of processor usage. CPU and wall times have been measured in order to determine the inherent parallelism when using multithreading.

C.3 Memory Footprint

We have measured the RAM memory usage, identifying the peak usage as well as the memory growth as the genome complexity and length increases.

C.4 Assembly Quality

We have calculated quality parameters for each of the evaluated assemblers and tested genomes. In particular, we considered the following parameters: contiguity (e.g., N50 parameters, number of contigs) and completeness using *QUAST*.

IV. Experimental evaluation

A. *Quality Assembly*

Quality parameters for HiFi reads are shown in Table I for *Escherichia coli* and the eukaryotic organisms: *Saccharomyces cerevisiae*, *Arabidopsis thaliana* and the human haploid cell *CHM13*.

Hifiasm primary assembly presents the best performance in the assembly of the four tested genomes in terms of contiguity with respect to *Canu*. It reports the minimum number of contigs and the longest contig (containing bubbles). The quality assembly evaluation in base on a reference with *QUAST* shows that *Hifiasm* presents a higher number of unaligned contigs (fully unaligned contigs) with respect to *Canu* for the four species assembled evaluated. Also, *Hifiasm* presents a lower number of misassemblies and mismatches compared to *Canu*.

B. *CPU Workload Characterization*

The performance analysis of the different assemblers shows that the novel assembler *Hifiasm* presents a much lower CPU time with respect to the traditional assemblers as *Canu*, predecessor of *Celera*, or *HiCanu*, added to *Canu* pipeline for PacBio Hifi samples, especially in the assembly of complex genomes. For example, the assembly of the high complexity genome of Homo Sapiens using HiFi reads from the haploid celular line (*CHM13*) takes about 273 CPU hours compared to more than 1,000 CPU hours taken by *HiCanu*. In table I we show the CPU time of *HiCanu* and *Hifiasm* with Hifi samples and organisms with different types of complexity. Although the CPU time is lower in *HiCanu* compared to *Hifiasm* with small genomes such as *Saccharomyces cerevisiae* and *Escherichia coli*, *Hifiasm* shows improved speedup over *HiCanu* with larger genomes.

However, the exploited thread-level parallelism is still limited. Results from the execution of *Hifiasm* in the *Picasso* cluster are depicted in Table II, showing how the speedup decreases as thread count increases from 8 to 128 threads, using low and high complexity genomes. It can be noted that the scalability of *Hifiasm* is not optimal, slightly decreasing as the thread count increases.

In the hotspot analysis we have found that the detection of overlapping reads consumes most of the CPU time (over 50 % of the total time) for all the assemblers. Also, the CPU time increases dramatically as the genome lenght grows, due to the large rise in the number of read comparisons. This fact can lead to a computing bottleneck when processing large and complex genomes in systems with limited hardware support for thread-level parallelism.

Hifiasm has a lower CPU time consumption regarding *MHAP* and *Minimap2*, both implemented in the *Canu* pipeline. However, it still spends about 90 % of the CPU time in all-versus-all pairwise alignment. For the assembly of the diploid cell line *HG002*, *Hifiasm* take about 300 CPU hours, over 90 % of the time in the sequencing correction and overlapping detection between the raw and corrected read.

C. *Memory Footprint*

Regarding the RAM memory usage, the evaluated assemblers present, in general, an increase in the me-

[8] *Escherichia coli* NCBI SRA: SRR10971019
[9] *Arabidopsis thaliana* NCBI SRA: ERR6210723
[10] *Saccharomyces cerevisiae* NCBI SRA: SRR13577847
[11] *Drosophila melanogaster* NCBI SRA: SRR10238607
[12] *CHM13 cell line* NCBI SRA: SRX789768* +CHM13
[13] *HG002 cell line* NCBI SRA: SRR10382244, SRR10382245, SRR10382248, SRR10382249

Tabla I: Experimental results of the evaluated assemblers

Data set	Genome size	Assembler	Quality assembly				Computational resources	
			N50	Contigs	Misassemblies	Unaligned	CPU time	RAM usage
Escherichia coli	5 MB	Hifiasm	4.67	1	9	0	19 h 31 min 51 sec	22.005 GB
		Canu (MHAP)	4.66	10	8	0	26 min	2.8 MB
S. cerevisiae	12.07 MB	Hifiasm	0.96 MB	39	73	0	3h 36.2 sec	16.704 GB
		Canu (MHAP)	0.81 MB	53	68	0	1h 35 min	2.29 GB
Arabidopsis thaliana	135 MB	Hifiasm	12.441MB	1218	1080	1	67h 28.32 min	33.445 GB
		Canu (MHAP)	6.01MB	1705	1770	10	182h 13min 48 sec	13.12GB
Human (CHM13)	3.1 GB	Hifiasm	102.83 MB	102	18261	3	272h 45 min	118.156 GB
		Canu (MHAP)	66.22 MB	9035	22811	198	984h 55 min	90.38 GB

Tabla II: Speedup of *Hifiasm* assembler using organisms with different polyploidy and complexity level and thread count from 8 to 128 (speedup measured with respect to 4 threads)

Genome	**Speedup (thread count)**				
	8	**16**	**32**	**64**	**128**
S. cerevisiae	1.93	3.67	6.83	12.10	17.61
Arabidopsis thaliana	1.96	3.76	7.16	13.19	20.54
D. melanogaster	1.97	3.80	7.43	14.04	23.41
Human (CHM13)	1.95	3.76	7.12	12.79	19.44
Human (HG002)	1.97	3.77	7.13	12.66	19.72

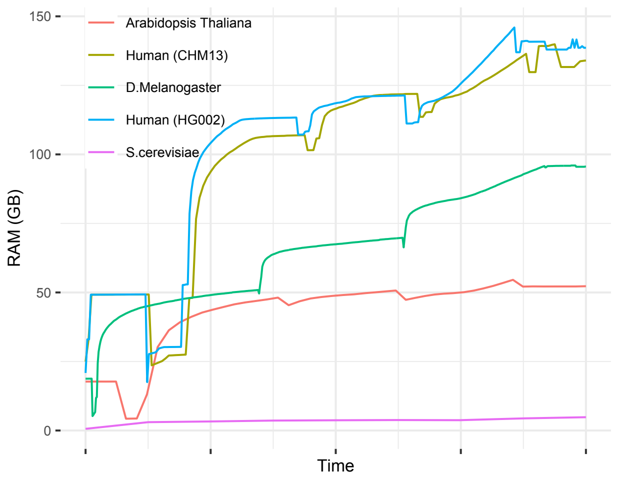

Fig. 2: *Hifiasm* memory usage with simple and complex genomes when executed in the *Picasso* cluster using 32 threads and 1.8 TB of RAM memory

mory footprint as the length of the genome or ploidy level grow.

The memory footprint of *Canu* is relatively small in relation to disk usage, as it works mainly with data on disk. Otherwise, the memory usage would be extremely huge. On the contrary, *Hifiasm* works mainly with data on memory, limiting the disk usage to the initial reading of sequencing files and the final writing of the resulting assembly. This fact boosts performance. It is also observed that some steps have an inefficient behavior when using files that only need to be accessed locally and are stored in a distributed file system. Figure 2 shows the memory usage during the execution of *Hifiasm* with the genomes considered in the evaluation. It can be noted that the processing of generated subsequences and the post-processing of overlapping sequences results in a huge memory footprint. This is explained as follows. If n is the read length and k is the k-mer size, then $n-k+1$ subsequences of k-mers are generated from a read and decomposing k-mers increases the length of the input sequence by a factor of $(n-k+1)*k/n$. As a result, processing the generated subsequences results in a peak memory explosion.

In particular, when processing the reads, *Hifiasm* has a RAM memory usage of more than 100 GB in the assembly of the human samples.

D. Performance Analysis

Figure 3 shows the performance metrics and Roofline for the *hifiasm* workflow obtained in the server with Intel Xeon Gold processors using 16 threads and the genome of *Saccharomyces cerevisiae*. The figure presents the relationship between performance (measured in integer operations, or INTOP) and arithmetic intensity of the different functions involved in the detection of overlapping regions between the reads and the sequencing correction. In red and yellow are marked those functions that show high and moderated costs in execution time, respectively.

A key observation based on the results is that most of the hotspots presents a low arithmetic intensity (below 0.1 INTOP/Byte) and, as consequence, their performance is limited by the available memory bandwidth. This is supported by the fact that their performance, in terms of INTOP/s, is (by far) lower than the computational peaks of the platform. From these preliminary results it follows that the *Hifiasm* assembler is memory bound most of the time, so as using an architecture with higher memory bandwidth could improve performance significantly.

V. Conclusions

In this work, we present the main biological and computational challenges in the genome assembly. We describe main methods in the genome assembly pipeline and present the main assemblers most widely used by the scientific community. We analyze experimentally the computational features of a recent assembler, *Hifiasm*, with respect to traditional assemblers, such as *Canu*, and present the main bottlenecks and computational costs.

We characterize the novel and fast assembler *Hifiasm* and find that the overlapping detection of the reads and the sequencing error correction present a low arithmetic intensity and high memory traffic. It could be a good niche for the research of new architectures based on near-memory processing which could mitigate the data movement and accelerate the most computational expensive functions in the genome assembly pipeline.

Acknowledgments

This work has been supported by the Government of Spain under project PID2019-105396RB-I00, and

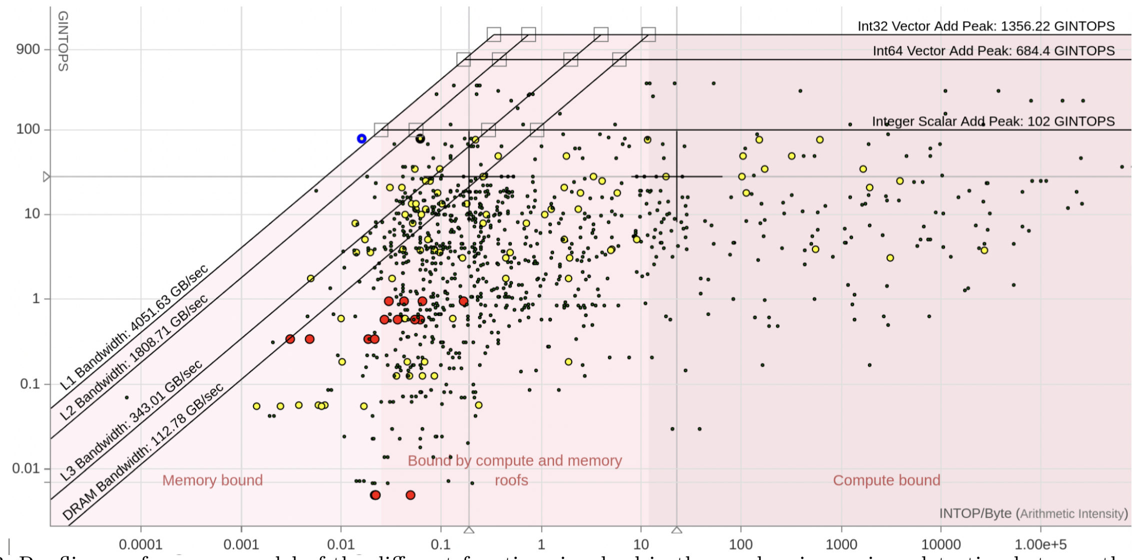

Fig. 3: Roofline performance model of the different functions involved in the overlapping regions detection between the reads and the correction of overlapping reads using *Hifiasm* as assembler and the genome of *Saccharomyces cerevisiae* as the input

Junta de Andalucia under projects P18-FR-3433 and UMA18-FEDERJA-197.

References

[1] Nicholas Stoler and Anton Nekrutenko, "Sequencing error profiles of Illumina sequencing instruments," *NAR Genomics and Bioinformatics*, vol. 3, no. 1, pp. lqab019, 2021.

[2] Mitchell R Vollger, Glennis A Logsdon, Peter A Audano, Arvis Sulovari, David Porubsky, Paul Peluso, Aaron M Wenger, Gregory T Concepcion, Zev N Kronenberg, Katherine M Munson, et al., "Improved assembly and variant detection of a haploid human genome using single-molecule, high-fidelity long reads," *Annals of Human Genetics*, vol. 84, no. 2, pp. 125–140, 2020.

[3] Ting Hon, Kristin Mars, Greg Young, Yu-Chih Tsai, Joseph W Karalius, Jane M Landolin, Nicholas Maurer, David Kudrna, Michael A Hardigan, Cynthia C Steiner, et al., "Highly accurate long-read HiFi sequencing data for five complex genomes," *Scientific Data*, vol. 7, no. 1, pp. 1–11, 2020.

[4] Sergey Koren, Brian P Walenz, Konstantin Berlin, Jason R Miller, Nicholas H Bergman, and Adam M Phillippy, "Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation," *Genome research*, vol. 27, no. 5, pp. 722–736, 2017.

[5] Glennis A Logsdon, Mitchell R Vollger, and Evan E Eichler, "Long-read human genome sequencing and its applications," *Nature Reviews Genetics*, vol. 21, no. 10, pp. 597–614, 2020.

[6] Haoyu Cheng, Gregory T Concepcion, Xiaowen Feng, Haowen Zhang, and Heng Li, "Haplotype-resolved de novo assembly using phased assembly graphs with hifiasm," *Nature Methods*, vol. 18, no. 2, pp. 170–175, 2021.

[7] Mark J Chaisson and Glenn Tesler, "Mapping single molecule sequencing reads using basic local alignment with successive refinement (blasr): application and theory," *BMC bioinformatics*, vol. 13, no. 1, pp. 1–18, 2012.

[8] Leslie Lamport, *LaTeX: A Document Preparation System*, Addison-Wesley, Reading, Massachusetts, 2nd edition, 1994.

[9] Firstname1 Lastname1 and Firstname2 Lastname2, "A very nice paper to cite," in *Proceedings of the 49th Annual IEEE/ACM International Symposium on Microarchitecture*, 2016.

[10] Firstname1 Lastname1, Firstname2 Lastname2, and Firstname3 Lastname3, "Another very nice paper to cite," in *Proceedings of the 48th Annual IEEE/ACM International Symposium on Microarchitecture*, 2015.

[11] Firstname1 Lastname1, Firstname2 Lastname2, Firstname3 Lastname3, Firstname4 Lastname4, Firstname5 Lastname5, Firstname6 Lastname6, Firstname7 Lastname7, Firstname8 Lastname8, Firstname9 Lastname9, Firstname10 Lastname10, Firstname11 Lastname11, and Firstname12 Lastname12, "Yet another very nice paper to cite, with many author names all spelled out," in *Proceedings of the 38th Annual International Symposium on Computer Architecture*, 2011.

[12] Adam M Phillippy, Michael C Schatz, and Mihai Pop, "Genome assembly forensics: finding the elusive misassembly," *Genome biology*, vol. 9, no. 3, pp. 1–13, 2008.

[13] Leslie Lamport, *LaTeX: User's guide & reference manual*, Addison-Wesley, 1986.

[14] Medhat Mahmoud, Marek Zywicki, Tomasz Twardowski, and Wojciech M Karlowski, "Efficiency of PacBio long read correction by 2nd generation Illumina sequencing," *Genomics*, vol. 111, no. 1, pp. 43–49, 2019.

[15] Alexander S Mikheyev and Mandy MY Tin, "A first look at the Oxford Nanopore MinION sequencer," *Molecular Ecology Resources*, vol. 14, no. 6, pp. 1097–1102, 2014.

[16] Konstantin Berlin, Sergey Koren, Chen-Shan Chin, James P Drake, Jane M Landolin, and Adam M Phillippy, "Assembling large genomes with single-molecule sequencing and locality-sensitive hashing," *Nature biotechnology*, vol. 33, no. 6, pp. 623–630, 2015.

[17] Gene Myers, "Efficient local alignment discovery amongst noisy long reads," in *International Workshop on Algorithms in Bioinformatics*. Springer, 2014, pp. 52–67.

[18] Heng Li, "Minimap2: pairwise alignment for nucleotide sequences," *Bioinformatics*, vol. 34, no. 18, pp. 3094–3100, 2018.

[19] Gennady Denisov, Brian Walenz, Aaron L Halpern, Jason Miller, Nelson Axelrod, Samuel Levy, and Granger Sutton, "Consensus generation and variant detection by celera assembler," *Bioinformatics*, vol. 24, no. 8, pp. 1035–1040, 2008.

[20] Sergey Nurk, Brian P Walenz, Arang Rhie, Mitchell R Vollger, Glennis A Logsdon, Robert Grothe, Karen H Miga, Evan E Eichler, Adam M Phillippy, and Sergey Koren, "Hicanu: accurate assembly of segmental duplications, satellites, and allelic variants from high-fidelity long reads," *Genome research*, vol. 30, no. 9, pp. 1291–1305, 2020.

[21] Mark JP Chaisson, Richard K Wilson, and Evan E Eichler, "Genetic variation and the de novo assembly of human genomes," *Nature Reviews Genetics*, vol. 16, no. 11, pp. 627–640, 2015.