Capítulo 5

From Sequence Assembly to Ancestry Testing: HPC Challenges for Bioinformatics

Mariana Carmin (UFPR), Cláudio Torres Júnior (UFPR), André Ricardo Abed Grégio (UFPR) e Marco Antonio Zanata Alves (UFPR)

Abstract

The bioinformatics field makes several types of analyses possible, such as verifying susceptibility to certain diseases and identifying samples in forensic science. Despite the growing use, the problems solved with bioinformatics techniques are still limited by the processing time and memory storage capacity, since the algorithms used are complex, often of quadratic or cubic orders, and may rely on large volumes of data. Therefore, it is necessary to have a correct understanding of the basic functioning of those techniques and the limitations and main challenges related to biological problems. As a contribution, in this chapter, we aim to provide an introductory view of embracing the collection and analysis of biological data. To accomplish that goal, we will discuss the main algorithms and databases and how to improve the algorithms applied to bioinformatics problems.

5.1. Introduction

Computers are powerful tools to help in solving today's biological problems. Most of these problems involve a huge amount of data. Commonly, this data comes in the form of images [2] or, as we will discuss in this Chapter, biological problems also make use of data from biological sequences [1].

When the problem involves working with images, those may be pictures or another data format used to construct images (e.g., magnetic resonance imaging (MRI) scan or recreation of the 3D protein structure[5]). For example, Klaczko and Blanche [3] calculate the size and shape of D. mediopunctata's wing using a photography of the microscope slide. To do so, they adjust an ellipse to fit the wing contour, and by using the ellipse parameters, they can calculate the size and shape of the wing. This paper can be applied to studies aiming to analyze the impacts of the selection process, evolution, and environmental genetic factors, and how these aspects influence Drosophila's wings [3].

Another widely studied field is predicting 3D protein structure. The multiple struc-

ture alignment (MSA) can be used to understand the conserved and divergent areas during evolution and to predict the function of the protein analyzed [5].

When we analyze the problems that use a biological sequence as data, the main focus of this chapter, we face many problems—from plague control to genetic disease prevention. We provide some specific examples below.

The Moniliophthora perniciosa causes the witches' broom disease (WBD) in cocoa and is an example of the use of bioinformatics applied to a real problem. In Brazil, there have been severe cases of WBD, in which phyto-sanitation practices and resistance breeding programs were applied to control the disease. However, these methods alone were not enough to solve the problem. In addition, the largest producers of cacao in the world are Africa and Asia, and if WBD reaches these areas, the worldwide production of chocolate could be affected [40]. Thus, sequencing the Moniliophthora perniciosa genome helped develop better control strategies for WBD.

Genome studies are also applied to humans. The Human Genome Project is the most significant collaborative work, which started in 1989 and ended in 2003. It encompasses scientists worldwide with a shared mission: to sequence all the human genome. Therefore, it is essential to notice the complexity associated with this task, as the human genome contains more than 3 billion base pairs (bp). Nevertheless, these complex steps are essential to advance genetic researches since knowing our genome code could help us understand how we function at the chemical level. Also, it could explain the role of genetic factors in many diseases, such as cancer, Alzheimer's disease, and schizophrenia, all of which decrease the lifespan of millions of people [41].

The human genome made it possible to perform another analysis. Another example is considering sequence variation, in which we can use this information for understating genetic diseases or investigate the predisposition for some disease or condition, not intending to act as treatment but prevention before any symptoms appear [2].

Furthermore, the information from sequence variation can be used to recognize the different effects/side effects of drugs in individuals, since they can vary from person to person. Therefore, this information can be used to develop more effective pharmaceuticals for everyone or a personalized drug for each patient according to the observed genomic sequence [2].

Nowadays, we cannot talk about bioinformatics research without considering the computation behind it. However, more than that, we also observe a quick adoption of computational analyses in the medical area [1].

Extracting scientific information from biological sequences is known as bioinformatics, by biological sequences we consider deoxyribonucleic acid (DNA), ribonucleic acid (RNA), or protein sequences. These sequences represent a vast amount of data, making it almost impossible to analyze them without a computer [2]. Furthermore, the more complex data lead to a necessity to improve the performance of the tools used.

For that reason, this chapter aims to explore the main areas in bioinformatics and describe the most famous algorithms and how we can improve their performance.

This chapter is organized as follows. First, section 5.2 describes the biological

information flow, from the DNA molecule to the sequence that can be processed and stored in a computer. Next, section 5.3 describes the main steps in bioinformatics, error correction, sequence assembly, and analyses. Next, Section 5.4 describes in detail the sequence alignment process, and finally, Section 5.5 describe the remains challenges in the area.

5.2. Biological information flow

DNA is the basis of all the genetic information used to produce everything necessary for an organism [21]. It's formed by a composition of nucleotides that is composed by a pentose (5-carbon) sugar, a phosphate group and a nitrogen base, which can be of four different types: **Adenine (A)**, **Cytosine (C)**, **Guanine (G)** and **Thymine (T)**. However, the DNA molecule is not an alone strand. Each molecule is formed by two strands forming a 3D spiral structure like a double helix, as seen in Figure 5.1. The strands connect to each other in a specific order: **A's** bases only pairs up with **T's**, and **C's** with **G's**.

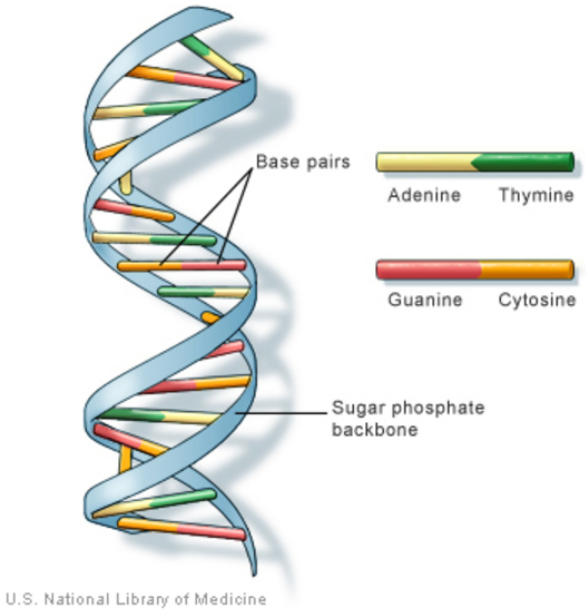

Figure 5.1. DNA molecule. Source: U.S. National Library of Medicine

Before starting the work with the DNA, we need to extract this molecule from the organism that will be studied. For this, many methods can be used and knowing the suitable one can be time and money-saving in the future steps of the experiment [11]. Different studies can have different purposes, so different tissues may need specific preparation methods. However, they share standard steps, like Efficient DNA extraction from the organism, creating necessary samples for the experiment to be performed, cleaning contaminants, and separating the DNA strands with higher quality [12].

When the DNA is ready and processed, the goal is to determine the ordered nitrogen bases from that sequence. For that reason, many technologies have emerged to carry out these processes and allow the realization of the most diverse experiments.

DNA was not the first molecule to be sequenced. In the early 50's, Frederick Sanger found out the constituent elements of insulin protein [22]. After that, many improvements and new methods were created and improved. Finally, in the mid of 70's, Sanger was able to sequence a whole DNA genome, the first one in history. The method used is considered the initial step to the first generation of sequencers [21].

In short, the double strand needs to be separated when a DNA molecule is inside the cell and needs to be duplicated (replication step). The Figure 5.2 show the steps:

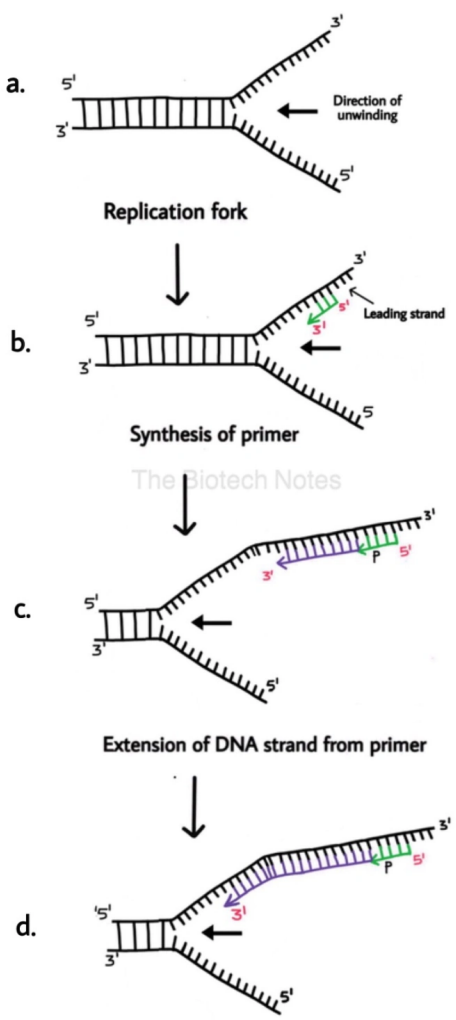

Figure 5.2. DNA replication. Credit: The Biotech Notes

a) An enzyme called *DNA helicase* breaks the bonds between bases.

b) When a portion is separated, a small sequence called *primer* binds to the start of the main strand.

c) The enzyme *DNA polymerase* creates the new strand by binding to the *primer* and extending the sequence with the corresponding base (if the base in the main strand is **A**, the incorporated base will be **T**).

d) In the end, the generated strand will be identical to the separated initial strand (the other strand goes through the same process but with a few more steps because it is read in reverse).

This is the basic principle used in most sequencers.

In 1977 Sanger improved his technique and developed the known Sanger Sequencing, as shown in Figure 5.3:

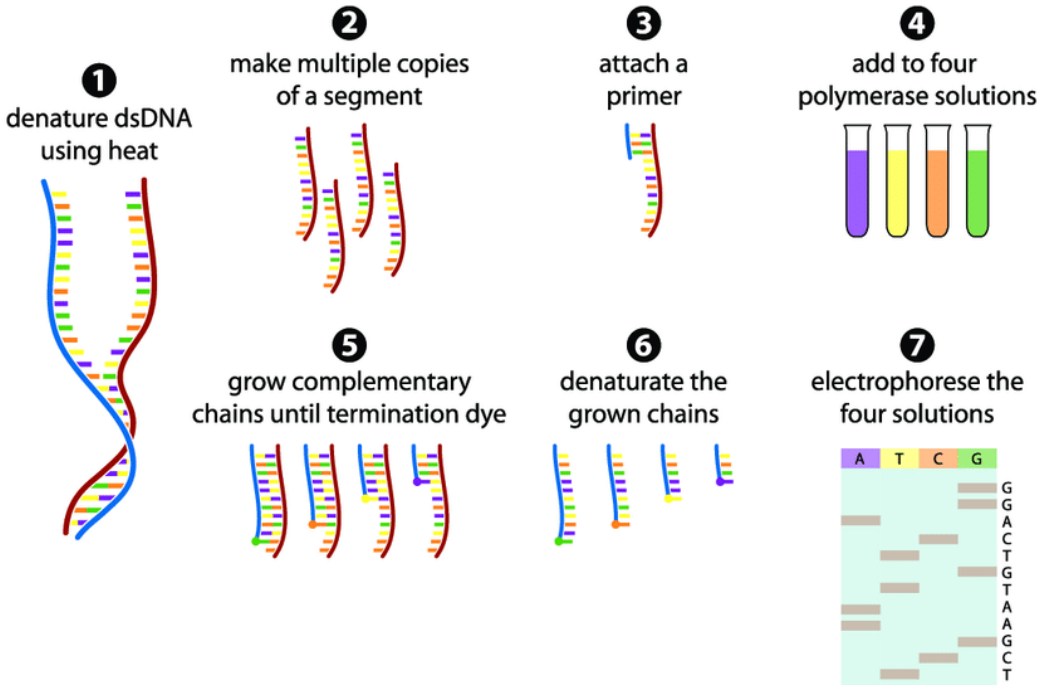

Figure 5.3. Sanger Sequencing [25]

1. Separate the double strand using heat.
2. For each strand (or segment of it) that will be sequenced make multiple copies.
3. Bind a primer to each segment.
4. Prepare four solutions containing enzymes *DNA polymerases* and all four nucleotides. The only difference is that one type of modified nucleotide is added in each solution. In this example, some **A** modified nucleotides were added in purple solution, and in yellow was added some **T**. This nucleotide is called dideoxynucleotide and its function is to block the *DNA polymerase* to continue extension.
5. The *DNA polymerases* binds to the *primer* and starts the replication. Normal nucleotides are added to the segment, and when a modified one is added, the process stops, leaving sequences of different sizes. In Figure 5.4, we have a demonstration of what happens in each solution (the colors do not represent the bases in Figure 5.3). Getting the **G** dideoxynucleotide, every time it's added to the sequence

Figure 5.4. Sanger step 5 example - DNA polymerases prevention extension. Adapted from [21]

end, the *DNA polymerases* stops. So we have in this example four different sequences with different lengths finished with **Guanine**.

6. All sequences are separated again, and we use the constructed strand from the previous step, which we call fragments.

7. The fragments pass through a polyacrylamide gel via electrophoresis technique. The solutions are placed in an equipment, and an electric current is applied to each one making the fragments travel by the gel pores. The smaller the sequences, the faster they go, traveling long distances in the gel. From bottom to top, the last gray box represents the smallest sequence that has **Thymine** as a terminal base, and then comes a sequence ending with **Cytosine**. Because **Thymine** terminal sequence went further, it is assumed that its length is smaller than **Cytosine** one. Therefore in the original sequence, the base **T** comes before **C**. So here, the sequence can be inferred as *TCGAATGTCAGG*, being the complementary sequence of the DNA sample of step 2.

As said, Sanger Sequencing was the starting point of the First Generation of sequencers. The result from all these steps produces a sequence called a read, with at most 1000 bases (one kilobase - kb) [21]. This technique was widely used, but the need to sequence larger genomes began to emerge. An example was the Human Genome Project, which had the objective sequence of approximately 3 billion base pairs from the human

genome with the collaboration of a worldwide team, having an estimated cost of one dollar per base [29]. The project was proposed in 1984 but only started in 1990, finishing on April 14, 2003 [29]. Finished but not completed. Only 92% of the human genome was sequenced because of the technologies until that moment. They could not sequence perfectly repetitive regions like the last 8% were. [30].

To circumvent some limitations of the past generation, in 2005, the Second Generation of sequencers started to appear, like the time needed to prepare the solutions, duplicate the sequences and the high cost of these steps [33]. The principle of this new era was to generate more leads in less time and lower costs. In addition, the reads here are smaller than the reads from Sanger (up to 400 bases long), so many more sequences are generated to compensate for possible gaps during genome assembly at the end of sequencing.

One of the technologies used is Illumina. This sequencer flow is shown in Figure 5.5:

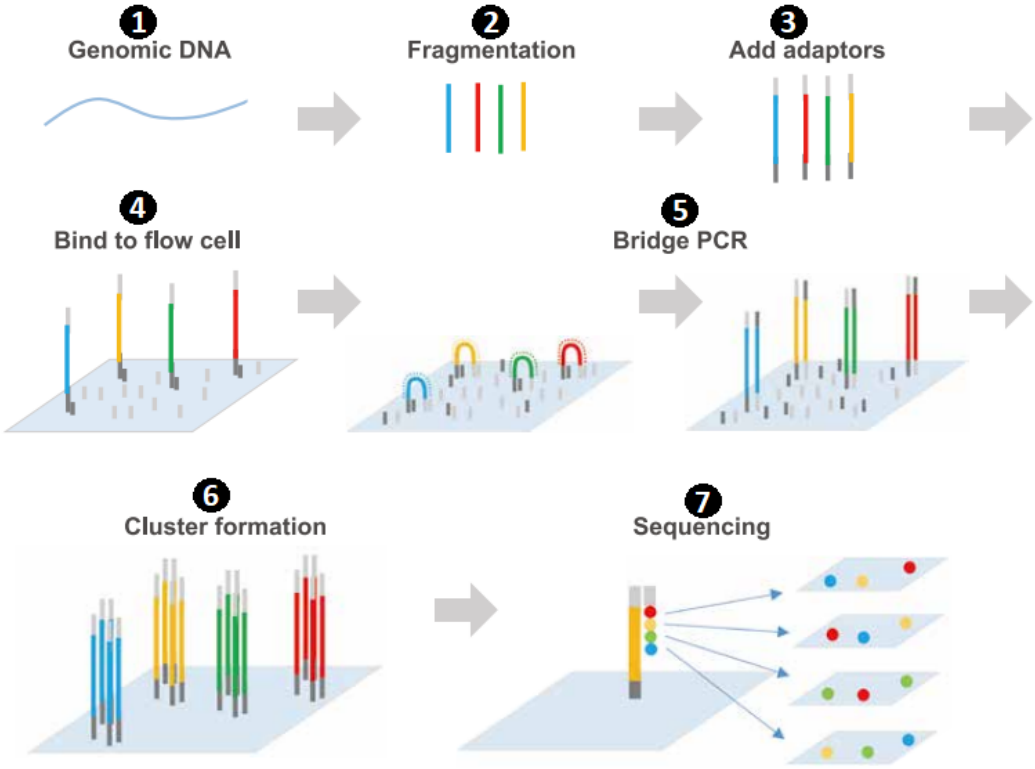

Figure 5.5. Illumina Sequencing. Adapted from [34]

1. Extract DNA strand to study.

2. Make DNA fragmentation from the last DNA strand.

3. Add adaptors to the ends of the fragments. Each adaptor is complementary to the sequence bases.

4. The adaptors will bind to the flow cell that contains others adaptors that are complementary to one of the fragments adaptors.

5. The fragment-free adaptor ends up binding to another adaptor of the cell, making a bridge between the two cell adaptors. Now, the *DNA polymerases* binds to one of the cell adaptors and starts building the complementary fragment. When the enzyme stops, the bridge is undone, and one of the adaptors is released, resulting in two sequences (the original and the copy). Creating a copy of the DNA strand is repeated thousands of times. The process is called Polymerase Chain Reaction (PCR).

6. When all fragments have already been copied, all reverse strands created are cleaved and eliminated from the flow cell. We will have a result cluster containing thousands of samples for each fragment from step 2.

7. *Primers* are attached to the free fragment end, and *DNA polymerases* starts building the complementary fragment. Each nucleotide has a different fluorescent element attached, so a light signal is emitted when the enzyme binds to the correct base. Belonging to a cluster, several signals will be released at the same time and captured by a sensor, which will identify which nucleotide was inserted. Like in Sanger, we will have the complementary sequence of the DNA fragments of step 2.

Unlike the Sanger method, which can sequence only one fragment at a time, Illumina can make the process in parallel, using the many clones of the fragments, drastically decreasing the execution time and increasing the number of reads generated [33]. As a result, nowadays is possible to use Second Generation technologies to sequence the human genome in up to 10 days and at a cost below 10.000 dollars.

Some limitations persisted, and for this reason, new improvements have been made. For instance, the reads generated until now were very small. For some genome assembly, this method can be a problem as thousands of reads need to be generated to deal with repetitive regions (best seen in Section 5.3), making the process expensive and slow. Therefore, the Third Generation came to sequence molecules giving results bigger reads. Another advantage that lower the costs is that it is not necessary to make DNA amplification and we can use the fragment alone, without clones [21].

One of the approaches and currently the most used method is the Single-Molecule real-time (SMRT), used by *Pacific Biosciences of California, Inc.* (*PacBio*) [21]. The idea of reading luminous signals is the same as the past generation. Before sequencing, the DNA sample needs to be prepared; this step takes some time. First, the double strand DNA is connected to adaptors creating a circular sample called *SMRTbell Library*. A *primer* and the *DNA polymerases* are added to the library as seen in Figure 5.6.

In Figure 5.7, we can see the PacBio flow described below.

a) Inside the PacBio machine are cells called SMRT Cell with millions of pores called Zero-mode Waveguides (ZMW). Each library created is mobilized in each ZMW. As in the Illumina method, we have labeled nucleotides with different fluorescent elements attached to them. Light is emitted as *DNA polymerases* attach these bases.

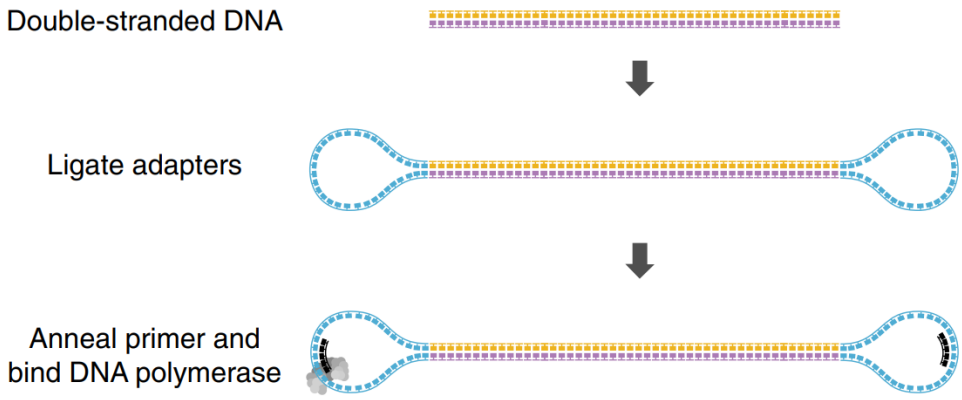

Figure 5.6. SMRTbell Library example. Adapted from [43]

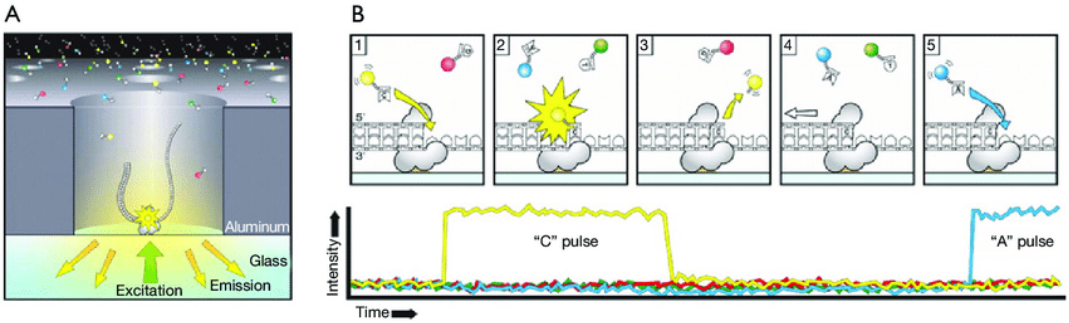

Figure 5.7. PacBio Sequencing [42].

b) With a sensor, nucleotide insertion is captured in real-time and depending on the intensity, we know the sequence of nucleotides.

The *DNA polymerases* works very quickly, and sometimes the signal is misread from the sensor or ends up not being read between one incorporation of bases and another. It is one reason for the generations of reads containing high error rates. Another problem that can occur is error/duplication in library ZMW binding. Each SMRT Cell has about 150.000 ZMW; from that, only 35.000 and 70.000 produce successful reads, with at most 60kb [44]. Without clones to make a consensus fragment, the error rate can reach 20% [17].

PacBio itself has a complementing of its method that reduces this error, using the maximum life of the *DNA polymerases* creating a clone of the sequence being sequenced, called HiFi sequencing. In Figure 5.8, The *DNA polymerases* is creating a Circular Consensus Sequence (CCS). An overlap of the CCS on itself to create a consensus sequence is done, decreasing the error rate. However, depending on the *DNA polymerases* lifetime, the CCS can be much smaller than the read without this technique, and the consensus

sequence can also decrease. With that, the experiment can become more expensive due to the increase in operations necessary to cover a region in the genome and the time spent in the CCS generation as well [44].

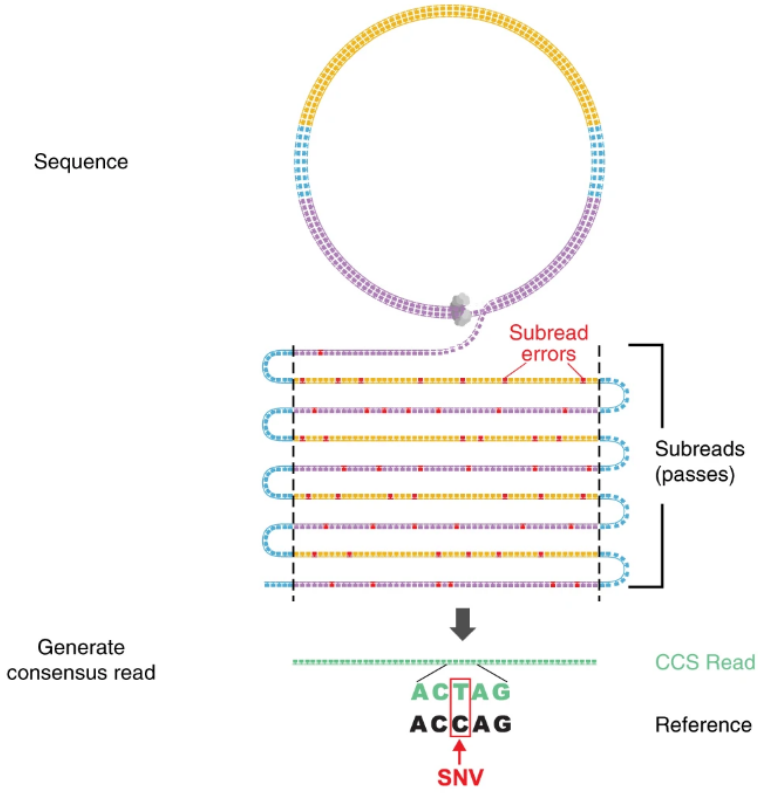

Figure 5.8. HIFI reads. Adapted from [43]

In Table 5.1, we can see a comparison between the technologies explained and all these details should be in mind before starting a study.

Table 5.1. Performance comparison of sequencing platforms of various generations. Adapted from [44].

	Sanger ABI 3730×l	Illumina HiSeq 2500 (High Output)	Illumina HiSeq 2500 (Rapid Run)	PacBio RS II: P6-C4
Read length (bp)	600–1000	2×125	2×250	$1.0–1.5\times10^4$ *onaverage*
Error rate (%)	0.001	0.1	0.1	13
Reads per run	96	8×10^9 (*paired*)	1.2×10^9 (*paired*)	$3.5–7.5\times10^4$
Time per run	0.5–3 h	7–60 h	1–6 days	0.5–4 h
Cost per million bases (USD)	500	0.03	0.04	0.40–0.80

Using long reads with a low error rate (from more modern sequencers or techniques that correct reads), it is possible to perform studies that were not possible in the

past, as some related to those of the human genome. For instance, in 2022, almost 20 years after the human genome sequencing, the unremitting regions that were not known were completely sequenced [30].

5.3. Error correction, Sequence assembly and Analysis

In this section, we discuss hybrid and self-error correction methods, providing examples, the basic algorithm for reading correction, and examples of sequence assembling and analysis for information discovery.

5.3.1. Error correction method

Depending on the type of experiment being performed and/or the organism being studied, some methods of sequencing from past generations may be better alternatives than current methods, having a reasonable cost benefit. The error rate and the resulting reads are low, and when used in smaller genomes, assembling the sequences is easier than when performed in larger genomes [13]. Getting the human genome, for example, when we try to assemble its regions by overlapping the short reads, we cannot make a consensus sequence due to many regions that are repetitive and complex [14].

As we can see in Figure 5.9, when using a small number of reads in a region much more extensive than them, may occur gaps in the consensus sequence. To work around this, we can generate a massive amount of short reads to increase the overlapping area, until we can assemble a complete consensus sequence. However, with that, we have an expensive experiment.

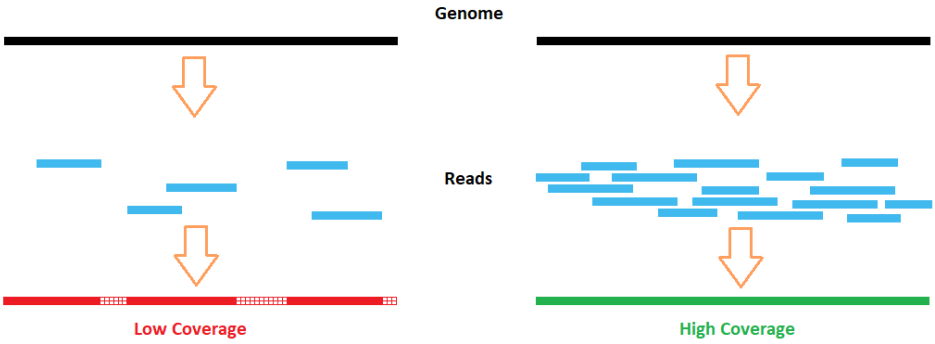

Figure 5.9. Example of a low and a high coverage consensus sequence

Thinking of another alternative, we can use the Third Generation Sequencing, like PacBio, that produces long reads than the previous generations [14]. With this, instead of using 100 small reads to make a consensus for a region, we can sequence this region as a long read with a lower cost but with the disadvantage of having a higher error rate. This is a problem because if we need to use these long reads with others to assemble another more extensive region, the present errors can make the process more complicated and increase the computational cost, in addition, to be imprecise in some repetitive areas [17].

To facilitate studies using larger genomes and allow research centers with low

investment to perform some experiments, we have methods to perform error corrections for long reads, decreasing the error rate. This is the first thing to think about when using long-read sequencing. New versions of PacBio have in it sequencing finalization steps to prevent high error rates (HiFi). However, this method ends up making the experiment more expensive, for example, making it impracticable. For this reason, other software works get the reads with a high error rate and process it, trying to correct the so-called wrong bases, using two main approaches [15].

5.3.1.1. Hybrid correction

This type of correction uses short reads from a known genome to work as a reference. Many techniques can be performed, like comparing the k-mers from a reference genome against the long reads. If we have the genome of the nematode *Caenorhabditis elegans* (*C. elegans*) organism and want to study it, for example, we can sequence it with PacBio. However, as we know, the resulting long reads may have an error rate up to 30%. The *C. elegans* was the first multicellular organism to have its approximately 100 Mega bases completely sequenced. Using the studied and labeled nematode genome as a reference to the experiment is a good approach because we can consider it without errors [16]. Nevertheless, suppose we do not have an organism completely sequenced. In that case, we can use short reads (like the ones from Illumina) as our reference genome because the error rate is minimal, so it is very close to the real genome [17].

For this, we have to create the k-mers from the reference genome. K-mers are substrings with length K inside a string, being this string a nucleotide sequence. For example, in Figure 5.10, we show k-mers of size 16 in a sequence. If the sequence has size L, we will have L - K + 1 k-mers in total.

Figure 5.10. Example of 16-mer in a sequence

If we want to know which organism a studied sequence belongs to, we just break it down in k-mers and compare their frequencies with the reference genome. The higher the frequency, the closer these organisms are.

K-mers with errors do not often appear, so their frequency will be minimal. Most of the time, they will not be present in the reference genome (in some cases, if we use other K values, some k-mers with errors can appear in different frequencies and match with any k-mer of the reference).

First, we have to determine the size of the k-mer (in this example, we will use 24)

and create a list with all k-mers of the reference genome. Now, let us suppose we have the following read with some error in it.

AAAAACCGAAAAAAGTGTGGACTTCCCGCGTGAAAAC(...)

In this example, K is 24 and we want to correct this read.

AAAAACCGAAAAAAGTGTGGACTT CCCGCGTGAAAAC

Running the Algorithm 1, a new k-mer is formed by sliding one base from the previous state. The result will be:

A AAAACCGAAAAAAGTGTGGACTTC CCGCGTGAAAAC

Now, we check if this new k-mer is present in the genome reference k-mer list. The algorithm continues sliding the k-mer window until it finds a k-mer that is not in the reference list.

AA AAACCGAAAAAAGTGTGGACTTCC CGCGTGAAAAC

AAA AACCGAAAAAAGTGTGGACTTCCC GCGTGAAAAC

AAAA ACCGAAAAAAGTGTGGACTTCCCG CGTGAAAAC **Not in k-mer list!**

When the problematic k-mer is found, test all possible errors that may have occurred. For each check, verify if the new k-mer is valid.

- **Insertion**: Remove G
- **Deletion**: Substitution of G by AG, CG, GG, TG
- **Mismatch**: Substitution of G by A, C, T

If only one valid k-mer is found, this is the corrected sequence. We go to more complex correction steps if 0 or more than one is found.

If the previous step fails, we can align the sequence with the reference genome. For this, the last valid k-mer found (the one before the problematic k-mer) has a complementary of it in the real genome. For example, if the reference genome is `AAACCCGGGTTT` and K is from size 6, the complementary of the 6-mer green is the 6-mer in cyan, as shown below.

Algorithm 1 Simple read correction

```
0: sum ← 0
0: C ← NULL
   for i ← k + 1 to L do
      kmer ←read[i :i+k]
      if not (kmer in reference genome) then
         newKmer ←checkInsertion(kmer, i)
         if newKmer in reference genome then
            sum ← sum + 1
            if sum == 1 then
               C ← newKmer
            end if
         end if
         for base in ['A', 'T', 'C', 'G'] do
            newKmer ←checkDeletion(kmer, i, base)
            if newKmer in reference genome then
               sum ← sum + 1
               if sum == 1 then
                  C ← newKmer
               end if
            end if
         end for
         for base in ['A', 'T', 'C', 'G'] do
            newKmer ←checkMismatch(kmer, i, base)
            if newKmer in reference genome then
               sum ← sum + 1
               if sum == 1 then
                  C ← newKmer
               end if
            end if
         end for
         if sum != 1 then
            C ← NULL
         end if
         return C
      end if
   end for
   return C =0
```

AAACCC GGGTTT

This works like a prefix and a suffix. For example, the green k-mer is the prefix of cyan k-mer. The reference suffix is aligned with the k-mer being analyzed plus its suffix. If the referenced prefix has more than one suffix, the operation occurs with all of it, and the corrected sequenced is the one with optimal alignment.

5.3.1.2. Self correction method

This method is more computationally expensive. A reference genome is not required and therefore is necessary to perform an alignment between all the reads with each other [47].

In Figure 5.11, we can see the workflow of CONSENT, a combination of self-correction methods from the state-of-the-art [47]. First, an overlap of some read's region is computed by an external tool. Then, CONSENT select a template read to correct. Every read region is not part of the template window. CONSENT is removed from the next steps because this region will not be used for correction. Next, windows are defined, and each of them is aligned. When the consensus of the windows is generated, this sequence is polished (corrected) by a local *de Bruijn graph*. In the end, the consensus sequence is aligned with the original template read to perform the correction of it.

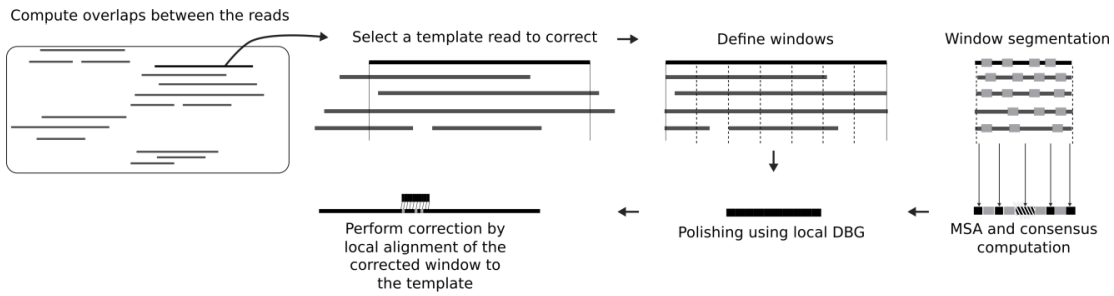

Figure 5.11. Overview of CONSENT workflow. [47]

5.3.2. Sequence assembly

When we have all the reads from the sequencing method, it is time to start reconstructing the original sequence. The sequencers only produce reads from random sizes (within a specific range depending on the technology used). For this reason, we need to use other strategies to overlap these reads and assemble them, as shown in Figure 5.12, in another continuous sequence called *contig*. This *contigs* are ordered, oriented and grouped together to form *scaffolds* [18].

Depending on the type of sequencer used, the number of reads generated per run can be considerable, as seen in Table 5.1. If the study focuses on a more complex organism and needs more extensive genome coverage, this implies a larger data storage. One example is a genome size of approximately 100MB with 40x of coverage, which gives us

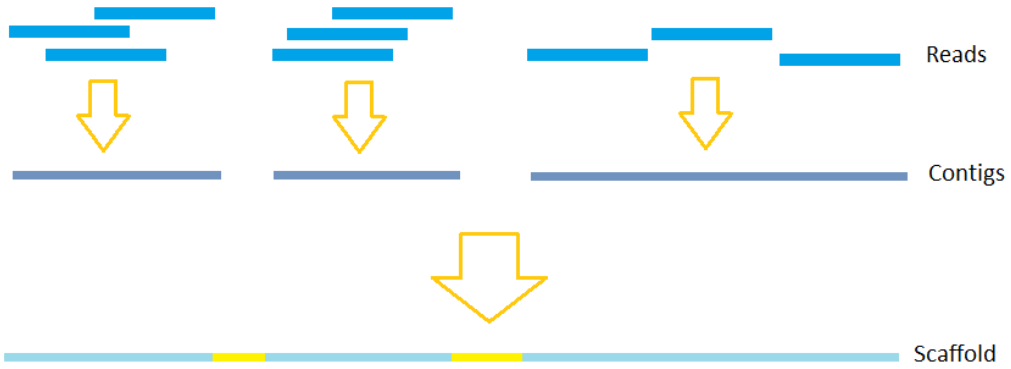

Figure 5.12. Example of assembly

a raw output of 4.5GB [45]. The larger the genome and/or the coverage, the greater the output. Genomes from approximately 3GB and 100x of coverage can reach terabytes of data [18].

All reads from a region need a good overlap with those of the same region to have the best assembly. If this doesn't happen, some *contigs* may not accurately build *scaffolds* and the result will be very fragmented [18]. Some software can help to find low-quality reads and regions, for example, from a database with reference genomes that have already been studied and sequenced by another sequencer [18].

So when we want to reconstruct the genome from an organism, short reads can lead to gaps in assembled sequences (using more short reads can work around this problem). While long reads with a low error rate can map certain regions by themselves, depending on the number of reads generated, making assembly with fewer gaps.

If the genome to be generated is new and/or does not have another sequence as a reference as a template, the approach used is the *De-novo* assembly. One of the algorithms uses the method of overlap. In short, each read is compared to the others to find the closest match and is merged to form a *contig*. The process is repeated until necessary, creating new *contig* and generating *scaffolds* at the end. This process is computationally expensive, since each assembly tool has a specific depending on the study to be carried out. Making it difficult to predict which one is the best to use [18]. If we have a template sequence, we can align the reads against it, and we will have a faster process with less memory consumption. The two types can be seen in Figure 5.13.

5.3.3. Analysis

A new sequence does not necessarily represent a discovery. It is necessary to analyze this sequence. Analyze the new sequence that leads to inferences and discoveries about it.

Sequence analysis started in the 1980s to compare two DNA sequences and find if they are homologous to each other (i.e., if the sequences share common areas) [48].

In the beginning, only a superficial analysis could be performed. After the 1990s, more complex analyses were performed with increased processor speed and storage ca-

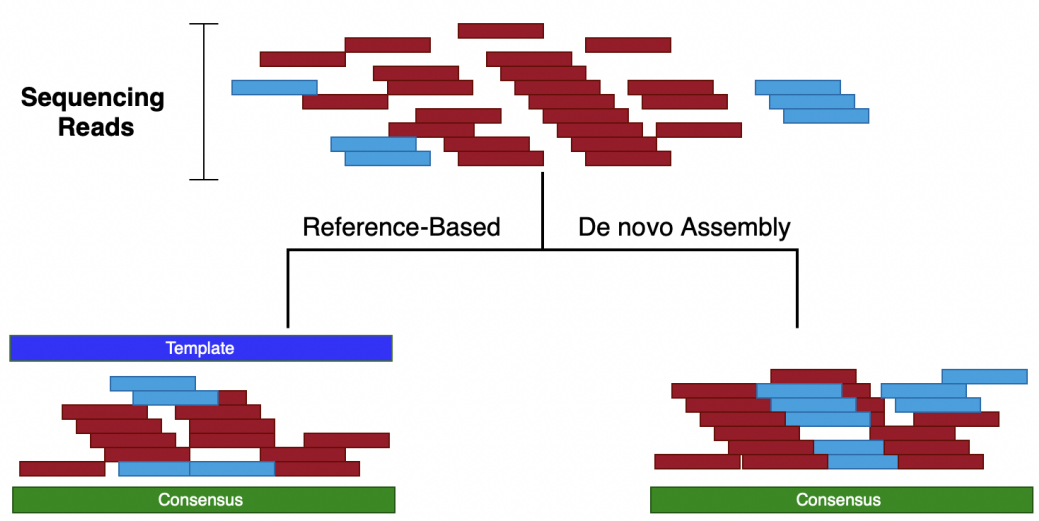

Figure 5.13. Types of sequencing assembly. [46]

pacity. For instance, analyze the family's histories [48]. Also, the software becomes more available for many researchers, enabling studies from different disciplines with a vast number of individual sequences being compared. Quantifying a specific characteristic became possible due to the advancement of technology.

There are vast applications for sequence analyses, such as finding similar sequences in a database, finding variations in the DNA such as point mutations and single nucleotide polymorphism (SNP), map evolution, and genetic diversity [48]. Nowadays, even with servers and supercomputing centers making it possible to do some analyses that seemed impossible in the past, the analysis's complexity and urgency seem to grow more each year. For example, the ancestry exams and the ones that can predict the risk for certain types of diseases are more frequent, increasing the demand for computing resources.

The demand to increase the efficiency of this analysis is evident. The most commonly used algorithm is sequence alignment, in all its variations.

The following subsection describes the sequence alignment, which can be used in many bioinformatics procedures. Also, we detail the different types and discuss the possible improvements in Smith-Waterman and CLUSTAL W algorithms.

5.4. Sequence alignment

After a new gene is found by scientists investigating a disease, the next step consists of finding the gene function. The typical approach used to infer a function is comparing the new one find with all already known gene sequences [8].

This approach has many successful cases. We will describe the first two reported in the literature. In 1984, scientists discovered a new *cancer-causing v-sis oncogene*[1]. Af-

[1]Oncogenes are genes capable of causing cancer, they can be mutated genes or an over-expressed gene. [8].

ter the discovery, they performed a computation technique to compare the new discovery with all the know genes until that moment.

At the first moment, we imagine that the result matches this new gene and another cancer-related gene. However, this is not what the scientists found.

Surprisingly, the gene show similarities with Platelet-Derived Growth Factor (PDGF), a gene involved in growth and development. This discovery was essential for scientists to understand cancer as an over-expressed gene [8].

Another example date back to 1989. The result of an alignment made it possible to associate the cystic fibrosis genes with an Adenosine Triphosphate (ATP) binding protein gene.

Known as the salty kiss disease, this association seen in the genes could explain the salty skin presented by people with cystic fibrosis. In contrast, the ATP binding protein gene is responsible for expanding the membrane cell multiple times to transport the sodium ions [8].

A current example of sequence alignment use is the ancestry exams. Most companies that provide this type of exam create a database with the DNAs sequences of their clients. Then, when a new client submits the collected DNA they compare, using sequence alignment, the new sequence against all the sequences in the built database. Therefore, proving information about the ancestry of the DNA sequence. Using this approach is also possible to find relatives by comparing the DNA sequences [50].

It is possible to notice that sequence alignment was essential for scientific discoveries, including understanding many diseases, helping to create a treatment, or even getting close to the cure. Therefore, sequence alignment is an indispensable procedure in the bioinformatics area.

Sequence alignment compares two or more sequences, which can be either an amino-acid (i.e., protein) or a nitrogenous base (i.e., DNA or RNA) sequence. When we compare only two sequences, we procedure a pair-wise alignment. In contrast, when there are more than two sequences is called a multiple sequence alignment [7].

Sequence alignments can also be separated into two categories, global and local alignment. On the one hand, the global alignment tries to align the entire sequence, considering all the amino-acids or nitrogenous bases until the end of the sequence analyzed. On the other hand, a local alignment aims to find the best subalignments or islands of matches from an aligned sequence. In this case, only a fraction of the sequence is aligned. The result, in this case, is the fraction that represents the highest number of matches. The following subsection discusses in detail each, including examples and applications [7].

In a simplified way, two sequences are aligned by writing one below another in two separate rows. The alignment is procedure character by character. We have a match if the two characters in the same column are identical. We have identical sequences if all the characters from both sequences are equal.

Otherwise, if the character is nonidentical, they can be characterized as a miss match, or a gap can be inserted. Gaps are used to optimize alignments, performing a sequence shift. An optimal alignment can include the addition of gaps to reach as many

matches as possible [7].

5.4.1. Global alignment

A global alignment can be executed in a protein, DNA, or RNA sequence. The primary process is the same for both types of sequences. In a protein, amino acids are aligned and in DNA and RNA sequences, nitrogenous bases are aligned [7].

The global alignment is stretched over the length of the entire sequence. The aim is to reach the highest number of matches in the sequences, namely the alignment score. Gaps can be added in any of the sequences, when a gap is added, the score suffers a penalty. The global alignment is performed up to and including the end of the sequences [7].

We analyze an alignment from column to column, comparing the characters presented in each sequence. If they are identical, we have a match. On the other hand, if the characters presented in a column are nonidentical, we have a miss match. Also, in this case, a gap can be added to improve the number of matches, is essential to notice that gaps added need to respect the length of the sequences, not exceeding it[7].

Figure 5.14 illustrates an example of global alignment in a hypothetical protein sequence. We can see two sequences and the matches and miss matches from each column. A match is observed in the first column, amino-acid L, which stands for Alanine. The second column is a miss match. Next, a gap is added in the third column of the second sequence, and we also see a gap in the 13th column of the first sequence.

Analyzing the entire sequence we observe seven matches (Alanine(**L**) in 1th column, Lysine(**K**) in 6th and 10th column, Glycine(**G**) in 9th and 11th column, Arginine(**R**) in 15th column and Aspartic acid(**D**) in 18th column) and eleven miss matches. Furthermore, two gaps were added.

Figure 5.14. Example of global alignment. Adapted from [7].

In summary, when sequences are similar and have the same length a global alignment is suitable to be performed in these sequences [7]. In most cases, global alignment is performed when sequences are related along their entire length. A local alignment can be used when only a region of similarity is sought.

5.4.2. Local alignment

Similar to a global alignment, sequences from proteins, DNAs, and RNAs can also be used in a local one. Although different from the global approach, in a local alignment, only a fraction of the sequence is aligned, searching in the sequence for the fraction representing the highest number of matches.

Figure 5.15 shows the same sequence illustrated in Figure 5.14, although, as we

can see, the optimum local alignment considers only the fraction of G-K-G (Glycine-Lysine-Glycine) (i.e., the three consecutive matches).

Figure 5.15. Example of local alignment. Adapted from [7].

This type of alignment is used to align sequences with conserved regions or domains, the ones with similarity in only a part of the sequence. Additionally, since only a fraction of the sequences are aligned, they can differ in length.

5.4.3. Pair-wise and Multiple sequence alignment

Pair-wise consists of aligning only two sequences. The sequences can be proteins, DNA or RNA. An extension of the pair-wise alignment is the multiple sequence alignment when the alignment is performed in three or more sequences.

For instance, a new sequence can be aligned against a sequence database to recognize the family or species, for example, [7]. Another example is finding conserved residues and identifying structural and functional domains when aligning more than two protein sequences.

5.4.4. Alignment scoring functions

For all sequence alignment types (i.e., local or global), we need to define the values associated with each scenario in the alignment, to calculate the alignment score.

When two sequences are being aligned, four scenarios are possible: (1) a gap in the first sequence, (2) a gap in the second sequence, (3) a match in both sequences, and (4) a miss-match between sequences.

Keeping these four scenarios in mind, a scoring function must attribute a value to each possible situation.

We will consider a simple scoring function in this chapter. -1 is assigned if a gap is added or a miss-matches is observed. For a match, +1 is attributed. This simple scoring function is describe in Equation 1 [1].

$$\begin{aligned} \sigma(-,a) = \sigma(a,-) = \sigma(a,b) = -1 \ \ \forall (a \neq b) \\ \sigma(a,b) = 1 \ \forall (a = b) \end{aligned} \tag{1}$$

In a global alignment, this function is used to score the sequence alignment since this alignment considers the entire sequence. Considering the example illustrated in Figure 5.16 and Equation 1, we have seven matches, three gaps, and two miss-matches. In this case, the result of the scoring function is equal to two [1].

$$A(\mathbf{s},\mathbf{t}) = \begin{array}{cccccccccccc} \mathtt{V} & \mathtt{I} & \mathtt{V} & \mathtt{A} & \mathtt{L} & \mathtt{A} & \mathtt{S} & \mathtt{V} & \mathtt{E} & \mathtt{G} & \mathtt{A} & \mathtt{S} \\ | & | & | & | & & | & & | & & & & | \\ \mathtt{V} & \mathtt{I} & \mathtt{V} & \mathtt{A} & \mathtt{D} & \mathtt{A} & - & \mathtt{V} & - & - & \mathtt{I} & \mathtt{S} \end{array}$$

Figure 5.16. Example of the score of a global alignment. Adapted from [1].

The scoring function for the example in Figure 5.16, is describe in Equation 2.

$$M(a) = 7 - 2 - 3 = 2. \tag{2}$$

The alignment score is directly associated with the function score. Therefore, the choice of the scoring function should try to reflect the biological behavior[1].

5.4.5. Substitution matrices

Substitution matrices can be used to consider multiple scoring functions and attribute different scoring for the substitution of one letter for another.

These matrices can be used for proteins or DNA and RNA sequences. The matrix shows the substitution cost between amino acids in a protein scenario. While in a DNA or RNA scenario, the cost reflects the cost of nucleotide substitution. Also, the match and gap values are shown in the matrices.

Using a substitution matrix is possible to expect ambiguous characters or even mutations and changes arising from the evolution process and assign a higher value for these changes [7].

Figure 5.17 illustrates a 5x5 matrix that can be used as a nucleotide substitution matrix. All pairs of $\sigma(a,b)$ nucleotide or gaps are shown. Note that $\sigma(-,-)$ has a not determined (**N/D**) value since an alignment between two gaps is not needed.

	A	C	G	T	−
A	+1	−1	−1	−1	−1
C	−1	+1	−1	−1	−1
G	−1	−1	+1	−1	−1
T	−1	−1	−1	+1	−1
-	−1	−1	−1	−1	N/D

Figure 5.17. Example of a substitution matrix. Source: [1].

5.4.6. Smith–Waterman

The Smith-Waterman [19] is an algorithm that uses dynamic programming to find, in any sequences of any lengths, where an optimum alignment can be found. The sequences can be either RNA, DNA or protein sequences.

5.4.6.1. The algorithm

The main steps of Smith-Waterman consist of:

- Initialization of a scoring matrix
- Filling the matrix with the calculated scores
- Trace back the sequences to find the best alignment

Considering two sequences, the first sequence's characters are placed at the top of the matrix, and the character from the second sequence are placed vertically on the left side. Next, the alignment score for each character pair (i.e., $\sigma(a,b)$) is placed in the corresponding matrix position. The code is described in Algorithm 2.

Algorithm 2 Smith–Waterman algorithm

0: $S[0,0] \leftarrow 0$
for $i \leftarrow 1$ **to** M **do**
$S[i,0] \leftarrow 0$
end for
for $j \leftarrow 1$ **to** N **do**
$S[0,j] \leftarrow 0$
for $i \leftarrow 1$ **to** M **do**

$$S[i,j] \leftarrow \text{MAX} \begin{cases} 0 \\ S[i-1,j-1] + \sigma(x_i, y_j) \\ S[i,j-1] + \sigma(-, y_j) \\ S[i-1,j] + \sigma(x_i,) \end{cases}$$

end for
end for
return $S[M,N]$ =0

To illustrate the functionality, we will describe the alignment of the sequences *PARALLELDNAALIGNMENT* and *DNASEQUENCE*. Analyzing both sequences, we can see that the subsequence DNA is present in both. Thus, the solution for this alignment is **DNA** as described in Figure 5.18.

Figure 5.18. DNA subsequence alignment.

Since the solution is already known, we will describe step by step the Smith-Waterman execution for the two sequences as inputs. The first step is configuring the score matrix. To create it, sequences are placed on top of the matrix and vertically on the left side, as illustrated in Figure 5.19.

		P	A	R	A	L	L	E	L		D	N	A	A		L	I	G	N	M	E	N	T
D																							
N																							
A																							
S																							
E																							
Q																							
U																							
E																							
N																							
C																							
E																							

Figure 5.19. Create matrix based in the sequences.

		P	A	R	A	L	L	E	L		D	N	A	A		L	I	G	N	M	E	N	T
		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0																						
N	0																						
A	0																						
S	0																						
E	0																						
Q	0																						
U	0																						
E	0																						
N	0																						
C	0																						
E	0																						

Figure 5.20. Initialize matrix with zeros.

After is the initialization, which consists of adding zeros to the first column and first line of the matrix, the result of this step is illustrated in Figure 5.20.

Next, we will use the scoring function described in Equation 1 to score all the pairs in the sequences. The result of this step is shown in Figure 5.21.

		P	A	R	A	L	L	E	L	D	N	A	A	L	I	G	N	M	E	N	T
		0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	1	0	0	1	0
A	0	1	0	1	0	0	0	0	0	0	0	3	1	0	0	0	0	0	0	0	0
S	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
N	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0
C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0

Figure 5.21. Score function calculation.

The last step consists of a trace-back procedure, starting with the highest element observed in the table of Figure 5.21 until a zero element in the trace-back path [1]. The

highest element in this example is three. Performing a trace-back, we find element two (i.e., a match between character N and character N). The next step is element one, the result of the match between D and D, and the next element is a zero, which stops the process. The result of the trace-back is shown in Figure 5.22.

		P	A	R	A	L	L	E	L	D	N	A	A	L	I	G	N	M	E	N	T
	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	1	0	0	1	0
N	0	1	0	1	0	0	0	0	0	0	3	1	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
S	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
E	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Q	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
U	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0
E	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0
N	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0
C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0

Figure 5.22. Trace-back procedure.

After all the steps, the alignment described in Figure 5.18 is reached. This way, finding the optimum alignment for the sequences.

5.4.6.2. Parallel solutions

It is possible to notice that Smith-Waterman is a time-consuming algorithm associated with all the complex procedures and comparisons. When we consider two sequences of lengths m and n. The computational and space complexity equals $O(mn)$ and $O(m)$, respectively. When we consider multiple alignments, the execution time gets even more significant, where $O(mn)$ is multiplied by the number of sequences being compared.

Many parallel solutions were developed to accelerate the algorithm, using technologies such as vector-level parallelism, thread-level parallelism, process-level parallelism, and heterogeneous approaches [20].

The first step when paralleling an algorithm is identifying and solving the dependencies. The Smith-Waterman algorithm has a dependency on the previously calculated scores. As described in Equation 3.

$$MAX \begin{cases} 0 \\ S[i-1, j-1] + \delta(x_i, y_j) \\ S[i, j-1] + \delta(-, y_j) \\ S[i-1, j] + \delta(x_{i,}) \end{cases} \quad (3)$$

The current position depend on the S[i-1, j-1], S[i-1, j] and S[i, j-1] positions, this dependency is illustrated in Figure 5.23. The first step in implementing a parallel solution is to solve this dependency so that the position calculation for the score matrix can be separated into groups.

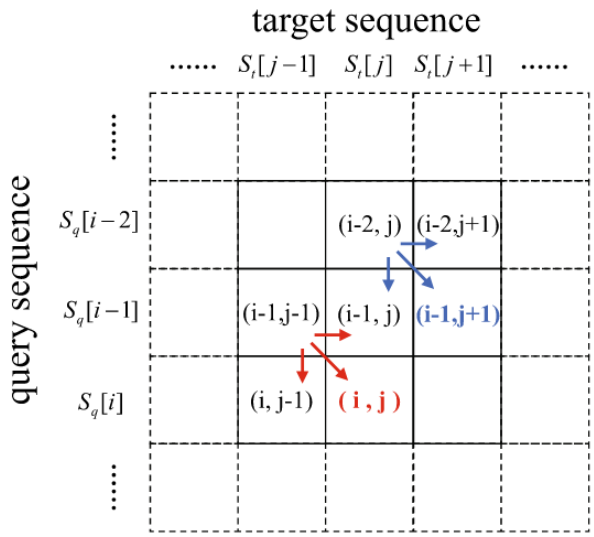

Figure 5.23. Existing dependencies while calculating the score matrix [20].

One possible technology for active parallelism is vector-level parallelism, also known as Single Instruction Multiple Data(SIMD), this type of parallelization performs the same operation in a group of data, namely spatial parallelism. For example, using a controller to control multiple processors, executing one instruction in multiple data [20]. Figure 5.24 shows the difference between vector and scalar operations.

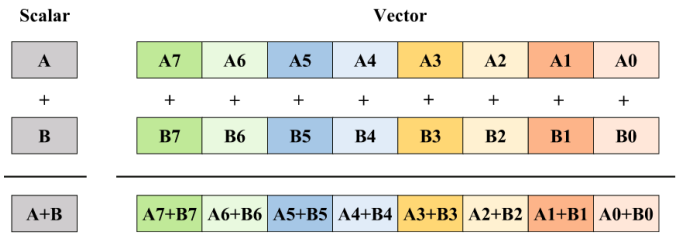

Figure 5.24. Vector and scalar operations [20].

When we consider solutions in vector-level parallelization, some possible strategies are reported in the literature to solve the dependencies.

In inter-sequence, the parallelization is performed in a pair of sequences. The available solutions are organized in anti-diagonal, sequential, or striped, described in Figure 5.25.

Anti-diagonal layout [26] works in the observation that calculation cell S[i,j] and S[i-1, j+1] is an independent task, thus possible to be calculated in parallel. Despite that, this approach does not change computation and space costs.

Another layout is the sequential [27]. Algorithm 3 describes the pseudo-code for the Lazy-F evaluation, a solution using the sequential layout to solve the data dependencies using auxiliary vectors.

Algorithm 3 Sequential-layout for Smith-Waterman algorithm

0: VECTOR4 vGo = $[\Delta,\Delta,\Delta,\Delta]$
0: VECTOR4 vGe = $[\delta,\delta,\delta,\delta]$
for $j \leftarrow 0$ **to** n/4 **do**
  $Hb \leftarrow [0,0,0,0]$
  $Eb \leftarrow [0,0,0,0]$
end for
for $i \leftarrow 0$ **to** m **do**
  $vHx \leftarrow [0,0,0,0]$
  $vF \leftarrow [0,0,0,0]$
  for $j \leftarrow 0$ **to** n/4 **do**
    $vH \leftarrow Hb$
    $vE \leftarrow Eb$
    $vTem1 \leftarrow vH >> 3$
    $vH \leftarrow (vH << 1)|vH$
    $vH \leftarrow vTem1$
    $vH \leftarrow vH + M[i][j]$
    $vH \leftarrow max(vH, vE)$
    $vF \leftarrow (vH << 1)|(vF >> 3)$
    $vF \leftarrow vF - vGo - vGe$
    if any values in vF > 0 **then**
      $vTem2 \leftarrow vF$
      while any values in vTem2 >0 **do**
        $vTem2 \leftarrow (vTem2 << 2) - vGe$
        $vF \leftarrow max(vF, vTem2$
      end while$vH \leftarrow max(vH, vF)$
      $vF \leftarrow max(vH, vF + vGo)$
    else
      $vF \leftarrow vH$
    end if$Hb \leftarrow vH$
    $Eb \leftarrow max(vH - vGo, vE) - vGe$
    $vMAX = max(vMAX, vH)$
  end for
end for=0

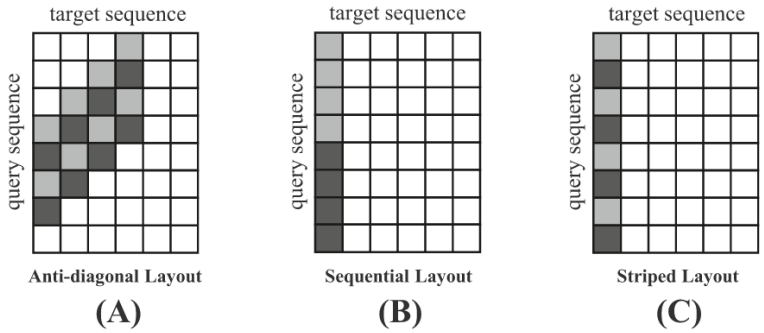

Figure 5.25. Inter-sequence parallel layouts [20].

In the sequential-layout, one character of the target sequence is aligned to the entire query sequence, as illustrated in Figure 5.25. The parallel solution came from dividing this process into equal parts and executing each part in parallel. For instance, the process is separated into four parts in the Algorithm 3. This number could be adapted for the SIMD technology used[26].

This solution still uses a computation of $O(mn)$ and space of order $O(m)$, nevertheless is faster than the previous solutions.

Finally, the striped layout [31] is a refined version of the sequential layout. The query elements are reorganized in this approach and, similar to sequential layout computation, remain O(mn).

Another way to achieve parallelism is using shared memory architectures to perform thread-level parallelism. In Figure 5.26 (A), we illustrate a shared memory architecture, in which the cores share access to the memory through an interconnection, differently from a distributed architecture, illustrated in Figure 5.26 (B), where each core has its own memory.

Some tools that provide parallelization using threads are POSIX Threads (Pthreads) [23] and OpenMP [24].

When we have the Figure 5.26 (b) architecture, we can reach parallelization using a process level. In this case, each process has an independent code segment and memory, which means the need to exchange information from one process to another. This exchange is done by communication. It is essential to notice that passing messages from one process to another represents an extra overhead to this technique. A commonly used protocol is Message-Passing Interface (MPI), a programming interface available for many programming languages.

Therefore, solutions that use shared memory in thread-level parallelism are available in the literature. For instance, KSW and KSW2 [51, 52], libssa[53], SeqAn[54], SWIPE[55], etc. One solution applied in thread parallel versions of Smith-Waterman is similar to the abovementioned solutions. Dividing the sequence to be aligned in subsets, normally equal to the number of threads available, each thread is responsible for its sequence [20].

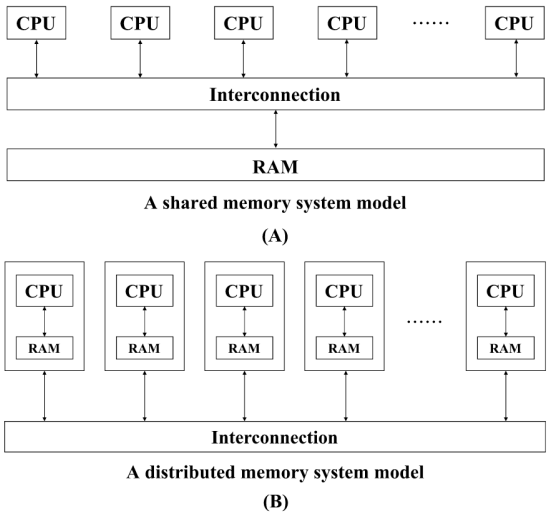

Figure 5.26. (A) Shared memory architecture; (B) Distributed memory architecture [20].

Nowadays, graphic processing units (GPUs) are becoming more common in personal computers. For that reason, heterogeneous parallel solutions, including the Central Processing Unit (CPU) and different accelerators such as GPUs are used [20].

Furthermore, in Python, a package named Biopython, have the alignment algorithm already implemented. Therefore, the algorithm automatically chooses the appropriate alignment algorithm between Needleman-Wunsch, Smith-Waterman, Gotoh, and Waterman-Smith-Beyer global. The decision is made based on the values of the gap scores observed in a pair-wise alignment.

Another widely used alignment algorithm is the one in which Smith-Waterman is based: Needleman-Wunsch. While Smith-Waterman procedure a local alignment, Needleman-Wunsch performs a global alignment.

5.4.7. CLUSTAL W

CLUSTAL W [32] is the most cited algorithm in bioinformatics. It is a global multiple alignment algorithm applied for three or more sequences, with a computational complexity of $O(n^2)$. For this reason, there is a need to improve CLUSTAL W performance since it is high complexity and vastly used algorithm. The algorithm uses dynamic programming to align the sequences, both amino acid and nucleotide sequences are accepted. The algorithm is divided into three main steps:

a) Calculate the distance matrix for each pair of sequences.

b) Determinate the topology of the progressive alignment.

c) Obtain the multiple alignments progressively.

In the literature, we have solutions in both shared and distributed memories. For instance, in shared memories, a solution used in most of the CLUSTAL W servers is proposed by Mikhailov et. al. [35], which uses OpenMP.

When analyzing distributed memory, a solution proposed by Liu [36] uses MPI to improve the average performance. All three algorithm steps were parallelized using the fixed-size chunking strategies, combining fine and coarse-grained division of work. They are reaching a speedup of 4.3% when 16 cores are used.

Another widely used algorithm is T-Coffee. This algorithm is also a multiple sequence aligner; although different from CLUSTAL W, it can be used for global or local alignment.

5.5. Open challenges

The importance of bioinformatics in our life is undeniable, and improvement and development still need to be done in many problems in this area. This section describes the complexity associated with the most used algorithms and the remaining challenges in bioinformatics studies.

One topic of study is redundancy removals in raw reads resulting from the DNA sequencing process. The first example is the MapReduce Duplicate Removal tool (MarDRe) [38], a tool to remove duplicate and near-duplicate DNA reads through the clustering of single-end and paired-end sequences from the FASTQ/FASTA dataset. Madre shows a computational complexity of O(logn). Similar to FastUniq [39], a tool with a functionality similar to MarDRe and an O(log N) complexity. Since the data volume is increasing, faster solutions are needed to process increasingly complex data. Improving these algorithms' performance is an open area.

Another new approach used in bioinformatics is machine learning. This technique could be used in many solutions, from predicting a gene function to analyzing MRI images. For instance, machine learning could be a tool to facilitate research or doctor analyses and anticipate some conditions or even a disease that still does not show symptoms.

In addition, considering all the areas listed above, the high-performance computing (HPC) area is essential in all of them. If a solution or algorithm cannot be executed in a feasible time, independently of how useful or innovative it may be, it can not be used.

In conclusion, all the bioinformatics problems and algorithms could be seen as a target for HPC since time is one of the most critical points in this area. Also, considering the increase in the data volume and complexity, even the faster algorithms for our current data could be insufficient in the future.

5.6. References

[1] Cristianini, N. and Hahn, M.W., 2006. Introduction to computational genomics: a case studies approach. Cambridge University Press.

[2] Stevens, Tim J., and Wayne Boucher. Python programming for biology. Cambridge University Press, 2015.

[3] Klaczko, Louis B., and Blanche C. Bitner-Mathe. "On the edge of a wing." Nature 346.6282 (1990): 321-321.

[4] Clark, Dominic A., Geoffrey J. Barton, and Christopher J. Rawlings. "A knowledge-based architecture for protein sequence analysis and structure prediction." Journal of Molecular Graphics 8.2 (1990): 94-107.

[5] Ma, Jianzhu, and Sheng Wang. "Algorithms, applications, and challenges of protein structure alignment." Advances in Protein Chemistry and Structural Biology 94 (2014): 121-175.

[6] Baldi, P. and Brunak, S., 2001. bioinformatics: the machine learning approach. MIT press.

[7] Gollery, M., 2005. Bioinformatics: sequence and genome analysis. Clinical Chemistry, 51(11), pp.2219-2220.

[8] Jones, N.C. and Pevzner, P.A., 2004. An introduction to bioinformatics algorithms. MIT press.

[9] Lehninger, A.L., Nelson, D.L., Cox, M.M. and Cox, M.M., 2005. Lehninger principles of biochemistry. Macmillan.

[10] Zomaya, A.Y., 2005. Parallel computing for bioinformatics and computational biology. Wiley.

[11] Preetha J. Shetty, 2020. The Evolution of DNA Extraction Methods. American Journal of Biomedical Science & Research.

[12] Anandika Dhaliwal, 2013. DNA Extraction and Purification. Materials and Methods volume 3.

[13] Immy Mobley, 2021. https://frontlinegenomics.com/dna-sequencing-how-to-choose-the-right-technology/.

[14] Immy Mobley, 2021. https://frontlinegenomics.com/long-read-sequencing-vs-short-read-sequencing/.

[15] Morisse, Pierre and Marchet, Camille and Limasset, Antoine and Lecroq, Thierry and Lefebvre, Arnaud, 2020. ONSENT: Scalable long read self-correction and assembly polishing with multiple sequence alignment. Cold Spring Harbor Laboratory.

[16] C. elegans Sequencing Consortium, 1998. Genome sequence of the nematode C. elegans: a platform for investigating biology. Science (New York, N.Y.), 282(5396), 2012–2018.

[17] Carvalho, Antonio Bernardo and Dupim, Eduardo G. and Nassar, Gabriel, 2016. Improved assembly of noisy long reads by k-mer validation. Cold Spring Harbor Laboratory.

[18] Ekblom, R. and Wolf, J.B.W. (2014), A field guide to whole-genome sequencing, assembly and annotation. Evol Appl, 7: 1026-1042

[19] Smith T.F., Waterman M.S., and Burks C. 1985. The statistical distribution of nucleic acid similarities. Nucleic Acids Res. 13:645-656.

[20] Xia, Zeyu, et al. "A review of parallel implementations for the smith–waterman algorithm." Interdisciplinary Sciences: Computational Life Sciences (2021): 1-14.

[21] James M. Heather, Benjamin Chain. The sequence of sequencers: The history of sequencing DNA, Genomics, Volume 107, Issue 1, 2016, Pages 1-8.

[22] Alice Maria Giani, Guido Roberto Gallo, Luca Gianfranceschi, Giulio Formenti, Long walk to genomics: History and current approaches to genome sequencing and assembly. Computational and Structural Biotechnology Journal. Volume 18, 2020, Pages 9-19

[23] Butenhof, David R. Programming with POSIX threads. Addison-Wesley Professional, 1997.

[24] Chandra, Rohit, et al. Parallel programming in OpenMP. Morgan kaufmann, 2001.

[25] Gauthier, Michel. (2007). Simulation of polymer translocation through small channels: A molecular dynamics study and a new Monte Carlo approach.

[26] Wozniak A (1997) Using video-oriented instructions to speed up sequence comparison. Bioinformatics 13(2):145–150. https://doi.org/10.1093/bioinformatics/13.2.145

[27] Rognes T, Seeberg E (2000) Six-fold speed-up of smith-waterman sequence database searches using parallel processing on common microprocessors. Bioinformatics 16(8):699–706. https://doi.org/ 10.1093/bioinformatics/16.8.699

[28] Rognes T (2011) Faster smith-waterman database searches with inter-sequence simd parallelisation. BMC Bioinform 12(1):1–11. https://doi.org/10.1186/1471-2105-12-221

[29] H. Zwart (2015) “Human Genome Project: history and assessment”. In: International Encyclopedia of Social Behavioral Sciences. 2nd ed. Oxford: Elsevier, 311–317.

[30] Sergey Nurk, et al. The complete sequence of a human genome (2022). Science 376(6588):44-53. https://doi.org/10.1126/science.abj6987

[31] Farrar M (2007) Striped smith-waterman speeds database searches six times over other simd implementations. Bioinformatics 23(2):156–161. https://doi.org/10.1093/bioinformatics/btl582

[32] Thompson,J.D., Higgins,D.G. and Gibson,T.J. (1994) CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. Nucleic Acids Res., 22, 4673–4680.

[33] Gut, I.G. New sequencing technologies. Clin Transl Oncol 15, 879–881 (2013). https://doi.org/10.1007/s12094-013-1073-6

[34] Dr. Josh Wang, Advancing genomics, medicine and health together – by semiconductor DNA synthesis technology, https://www.genscript.com/advancing-genomics-medicine-and-health-together-by-semiconductor-dna-synthesis-technology-summary.html

[35] Mikhailov,D., Cofer,H. and Gomperts,R. (2001) Performance opti- mization of Clustal W: parallel Clustal W, HT Clustal, and MUL- TICLUSTAL, White papers, Silicon Graphics, Mountain View,CA

[36] Li, Kuo-Bin. "ClustalW-MPI: ClustalW analysis using distributed and parallel computing." Bioinformatics 19.12 (2003): 1585-1586.

[37] Henikoff, Steven, and Jorja G. Henikoff. "Amino acid substitution matrices from protein blocks." Proceedings of the National Academy of Sciences 89.22 (1992): 10915-10919.

[38] Expósito, R. R., J. Veiga, J. González Domínguez and J. Touriño, 2017. Mardre: Efficient mapreduce based removal of duplicate DNA reads in the cloud. Bioinformatics, 33(17): 2762 - 2764.

[39] Xu, H., X. Luo, J. Qian, X. Pang, J. Song, G. Qian, J. Chen and S. Chen, 2012. Fastuniq: A fast de novo duplicates removal tool for paired short reads. PloS one, 7(12): e52249.

[40] Costa, Gustavo GL, et al. "The mitochondrial genome of Moniliophthora roreri, the frosty pod rot pathogen of cacao." Fungal Biology 116.5 (2012): 551-562.

[41] Watson, James D. "The human genome project: past, present, and future." Science 248.4951 (1990): 44-49.

[42] Xiao, Tiantian Zhou, Wenhao. (2020). The third generation sequencing: The advanced approach to genetic diseases. Translational Pediatrics. 9. 163-173. 10.21037/tp.2020.03.06.

[43] Wenger, A.M., Peluso, P., Rowell, W.J. et al. Accurate circular consensus long-read sequencing improves variant detection and assembly of a human genome. Nat Biotechnol 37, 1155–1162 (2019). https://doi.org/10.1038/s41587-019-0217-9

[44] Anthony Rhoads, Kin Fai Au, PacBio Sequencing and Its Applications, Genomics, Proteomics Bioinformatics, Volume 13, Issue 5, 2015, Pages 278-289, ISSN 1672-0229, https://doi.org/10.1016/j.gpb.2015.08.002.

[45] PacBio, New Chemistry Boosts Average Read Length to 10 kb – 15 kb for PacBio® RS II, 2014, https://www.pacb.com/blog/new-chemistry-boosts-average-read/

[46] By Erekevan - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=116777958

[47] Morisse, Pierre and Marchet, Camille and Limasset, Antoine and Lecroq, Thierry and Lefebvre, Arnaud, 2020, CONSENT: Scalable long read self-correction and assembly polishing with multiple sequence alignment, bioRxiv - Cold Spring Harbor Laboratory. https://doi.org/10.1101/546630

[48] Brzinsky-Fay, Christian, and Ulrich Kohler. "New developments in sequence analysis." Sociological methods research 38.3 (2010): 359-364.

[49] Hu, Y., Colantonio, V., Müller, B.S.F. et al. Genome assembly and population genomic analysis provide insights into the evolution of modern sweet corn. Nat Commun 12, 1227 (2021). https://doi.org/10.1038/s41467-021-21380-4

[50] Jorde, Lynn B., and Michael J. Bamshad. "Genetic ancestry testing: What is it and why is it important?." JAMA 323.11 (2020): 1089-1090.

[51] Li H (2018) Minimap2: pairwise alignment for nucleotide sequences. Bioinformatics 34(18):3094–3100. https:// doi. org/ 10.1093/bioinformatics/bty191

[52] Suzuki H, Kasahara M (2018) Introducing difference recurrence relations for faster semi-global alignment of long sequences. BMC Bioinform 19(1):33–47. https:// doi. org/ 10. 1186/ s12859-018-2014-8

[53] Rognes T (2011) Faster smith-waterman database searches with inter-sequence simd parallelisation. BMC Bioinform 12(1):1–11. https://doi.org/10.1186/1471-2105-12-221

[54] Rahn R, Budach S, Costanza P, Ehrhardt M, Hancox J, Reinert K (2018) Generic accelerated sequence alignment in seqan using vectorization and multi-threading. Bioinformatics 34(20):3437– 3445. https://doi.org/10.1093/bioinformatics/bty380

[55] Rognes T (2011) Faster smith-waterman database searches with inter-sequence simd parallelisation. BMC Bioinform 12(1):1–11. https://doi.org/10.1186/1471-2105-12-221