POLITECNICO DI TORINO

III Facoltà di Ingegneria

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea Magistrale

Novel Approaches and Algorithms for the Alignment of Third Generation Sequencing Long-Reads

Relatori:
Prof.sa Elisa Ficarra
Ph.D. Gianvito Urgese

Emanuele Parisi

Aprile 2018

Acknowledgments

Sono molte le persone senza le quali questo lavoro non avrebbe mai visto la luce. Innanzitutto desidero ringraziare la Prof.sa Elisa Ficarra per avermi dato la possibilità di lavorare al fianco di persone disponibili e competenti e per aver indirizzato questo lavoro verso lo studio di tecnologie di grande attualità, che promettono di rivoluzionare il campo della genomica.

Un ringraziamento particolare a Gianvito per essere stato sempre disponibile nei miei comfronti e per aver messo a mia completa disposizione la sua grande esperienza nel campo della bioinformatica.

L'amore dei miei genitori, Antonio e Lucia, che sono sempre stati al mio fianco e mi hanno sostenuto ed incoraggiato sia nei momenti di serenità che in quelli di sconforto, assecondando tutte le mie scelte.

In ultimo, impossibile non ringraziare Davide, Maria, Samuele, Gloria, Michele e Davide per gli innumerevoli pomeriggi di studio passati in loro compagnia: il loro supporto è stato centrale per tutta la durata di questo lungo percorso di studi.

Summary

During the last few years novel sequencing machine appeared on the market, exploiting different protocols for sequencing and producing data exposing radically different properties with respect to previous Next Generation Sequencing machines. The main characteristics of such novel data are an enhanced read length, approaching hundreds of thousands of base-pairs long reads, and a higher error rate, over 30% for some technologies. These new technologies requires rethinking the architectures of bioinformatics pipelines built for managing the data they produce: in particular, read alignment tools experienced the major changes with respect to the past, exploiting new representations of biologic sequences for the sake of read mapping, based on sequence *fingerprints*, a concise way of representing the informative content of a sequence of nucleobases as a set of features, still able to detect the best mapping positions over the reference sequence.

This thesis work has two main targets: defining the main characteristics a third-generation sequencing alignment tool should have, in terms of data representation, data structures exploited for indexing and mapping algorithms and trying to extend the approaches already present in literature, proposing new algorithms for the management of read fingerprints.

Alignment tools pipelines traditionally work on two distinct steps: the former is called mapping and aim at finding the reference regions which are

most promising for read mapping, i.e. the regions exposing the highest degree of similarity with the sequence to be mapped; the latter step is not always required, and consists in a detailed base-per-base alignment procedure, exploiting Dynamic Programming for discovering SNP or other local variants. All third-generation sequencing alignment algorithms present in literature up to now rely on well-known Dynamic Programming algorithm for base-per-base alignment, introducing novel approaches for read mapping, making this field more interesting to analyze with respect to the previous one.

A preliminary literature review underlines that state-of-the-art tools can be divided into two families regarding the way they perform mapping: algorithms that exploits traditional tree-based data structures for performing fast exact-match seed search and clustering, selecting regions of the reference exposing a denser exact matches and approaches that transforms a sequence into list of its overlapping sub-strings of a given, fixed, length, hashes them and exploits the *Winnowing* procedure for selecting the most relevant among the so obtained list of hash values, called features, that are then looked for in the reference sequence. Analyzing run-time, accuracy and precision of *BLASR* and *MashMap*, two modern tools exploiting old-fashioned tree data structure, the former, and a Winnowing-driven reduced data representation, the latter, a first important result emerges: the run-time of fingerprint-features based tools is always lower than the one of traditional tools still retaining similar accuracy for every kind of read length and error rate, assumed lower than 20%; in particular, indexing steps of novel methods is up to six times faster than building suffix arrays for the small *Escherichia coli* genome and up to eighteen time faster for the Human genome, meaning that such novel approaches worth being investigated.

Exploiting the previous results, assuring that fingerprint-based sequence

representation holds sufficient information for the sake of read mapping, two novel methods are investigated. The first approach exploits Bloom filters, a probabilistic data structure for concise set representation, for compressing sets of hash values computed resorting to the well-defined *Winnowing* algorithm, obtaining an even more concise representation of the features of a sequence. The key idea is to translate a sequence of nucleotides into a sequence of hash values first, through the Winnowing procedure, and into a sequence of Bloom filters next, representing a certain number of hashes, decided at design time, as an array of bits; the highest similarity regions between to sequences represented in such a way can be found by cross-correlating one sequence against the other, substituting the product operation, typical for the cross-correlation of two arrays of integers, with a mathematical relationship able to express how much the sets two Bloom filters represent are similar, by looking only at the bits of the filters they are represented by. This method was implemented in C++ exploiting a library called *SeqAn*, suitable for easily manage file formats and data types typical for programs aiming at managing biological sequences. This approach proved to work in principle, as there exists parameters able to reach near perfect accuracy; however, it suffers from two major drawbacks: it depends on four different parameters, two regarding the Winnowing procedure and two for the designing of Bloom filters, making difficult to analytically describe the behavior of the tool. Moreover, Bloom filter sequences proved to be difficult to index, making this tool relies on a brute-force approach for finding the best mapping positions, making it impossible to be used as it is, due to high runtime. The second approach proposed is to exploit traditional seed-and-chain algorithms, already proven to work by previous second-generation alignment tools, substituting the concept of seed with the one of hash feature. First of all, the reference sequence

is indexed: its fingerprint is built from the Winnowing procedure, and a hash table is generated, having the hashed features as keys and the positions in the reference sequence where they occur. Whenever a query sequence has to be mapped it is fingerprinted with the same procedure the genome were subject to, keeping track of what feature comes from what query position: then the index is queries for each of query hashes, obtaining, at the end, a set of points of match, between the query and the reference, made of two coordinates: a position on the reference and a position on the query. The chaining procedure consists in joining together points that are approximately co-linear, meaning they relies on the same diagonal, from the geometric point of view, assuming that a set made of a given amount of co-linear points indicates a region of similarity between the query and the reference sequence. This method was again implemented in C++, exploiting the same library for sequence management, obtaining much more encouraging results: for every error rate it is able to reach near 100controlled by the knobs parameter of the algorithm. Moreover, is very easy to build up indexes for such approach, making it more suitable for real-life applications. However, it proved to be still not as fast as competitor tools, meaning that an in-depth profiling of the code is needed before the tool to be completed. The above analysis on the state-of-the-art tools for third-generation sequencing long reads mapping and the results of the two approaches proposed in the context of such work allow to define a set of characteristics any bioinformatics pipeline aiming at mapping or aligning long-reads should have. The great homogeneity of read lengths in typical real-life third-generation sequencing datasets should be taken into account for the sake of mapping the shortest among the sequences in the datasets, whose number is often non-negligible. For mapping only tools, the most important error parameter for algorithms parametrization is the overall error

rate, meaning that the mapping phase can be accomplished not taking into account the different rates of insertion, deletion and substitution errors that characterized the different technologies. The concept of *fingerprint*, i.e. the idea of using only a part of the nucleobases composing a sequence for finding high similarity regions between two sequences proved to work, meaning that such representation still maintains a sufficient amount of information for the sake of mapping; moreover, for the management of *fingerprints* an indexing scheme based on storing the positions in the reference where each feature can be found in an hash table proved not only to work, but also to be a very efficient method. Concerning mapping approaches, up to now, hashes exact match search, followed by a chaining step seem to be the best way of solving the long-reads mapping problem, assuming the dataset to be mapped is a copy of the reference sequence, perturbed only by technological error, not containing any kind of biological events able to make mapping more difficult. On the other hand, fingerprint compression using Bloom filter, while being a promising method, needs more investigation before being used extensively in a real bioinformatics pipelines.

Contents

Chapter 1

Introduction

We as humans, together with all the remaining organisms on the Earth, can be referred to as *living being*. This sentence, that could seem innocent, can soon demand for further considerations as we ask ourselves a question the whole mankind asks itself from almost the very beginning of its existence: *what is life ?*. The human history is full of possible answers to such a issue, coming from philosophy, literature or religion, but starting from the early years of the nineteen century, the science approached such dilemma proposing possible answers starting dissociating from the approaches taken by other disciplines so far. In 1839 Schleiden and Schwann formulated the so called *cell theory*, claiming that the cell is the basic unit of structure and organization in organisms and that all living organisms are composed of one or more cells, suggesting that answer any question about life, from a scientific point of view, would have required a deeper understanding of what these organisms basic blocks are and how they work. Subsequent advances in biology and chemistry made clear that the activities an organism can take, that are often used for defining life, such as growing, reproducing and responding to external stimuli, are accomplished first of all at the cell level. Moreover, the

relationship between characterizing the cell behaviour and understanding its inner bio-molecular interactions was becoming more and more clear as Francis Creek, one of the scientists that won the Nobel prize for discovering the DNA structure in 1953, in 1988 claimed[4]:

> "We also now appreciate that molecular biology is not a trivial aspect of biological systems. It is at the heart of the matter. Almost all aspects of life are engineered at the molecular level, and without understanding molecules, we can only have a very sketchy understanding of life itself. All approaches at a higher level are suspect until confirmed at the molecular level.".

What Crick stated in the quotation above is the impossibility for modern biology these days to reach a deep understanding of the remarkable properties of processes which defines life, without investigating them at the biological level.

At first sight, much of the cell content can be considered as a soup made of water, ions and small molecules. One of the most studied small molecule is adenosine-triphosphate, ATP in short, a readily available source of energy able to power most of the energy-requiring processes which continuously take place in the cell; other small molecules are in charge of regulating the response to external stimuli or carrying on extra-cell signal transmissions, such as hormones and neurotransmitter[15]. Macromolecules are large assembly of small repeating units that can be classified into four groups: *polysaccharides*, *lipids*, *proteins* and *nucleic acids*, depending on the chemical type of the small units composing them. Polysaccharides consist in complex linear chains of simpler sugar molecules called *monosaccharides*; they are mostly used by cells as power storage, along with lipids, which, in addition, are also involved in some cell signaling processes and constitute the main brick

cellular membranes are made of. From the structural point of view, proteins are strings of aminoacids, organic compounds containing amine and carboxyl functional groups plus a side chain specific to each of them; among the macromolecules those are the one carrying on most tasks within the cell environment: they can serve as structural components for the cell, detector for changes in temperature or ions concentration, enzymes, catalyst and extra-cellular transmitter. The ability of a single molecular family of accomplishing such a great variety of functions is justified by the exponential grow of the number of possible protein sequences that can be assembled starting from the twenty different kind of amino acids able to join together: an average sized chain made of four hundred amino acids can results in 20^{400} different proteins, ensuring almost infinite possibilities in the shape and dimension such molecules can take. Given the great importance of proteins for the completion of cells most important tasks, the even more important role of deciding what kind of protein has to be synthesized, when and in what quantity, is performed by nucleic acids, whose most well-known example is the deoxyribonucleic acid, better known as DNA[15].

1.1 The information content of DNA

Deoxyribonucleic acid, or *DNA*, is a bio-molecule consisting in a double-stranded chain of nucleotides. A nucleotide is a smaller particle structurally divisible into three parts: a sugar, a phosphate group and a base. The sugar is a ring of carbon atoms that, for notation convenience, will be addressed by their positional number, from 1' to 5' counting from the one on the right of the oxygen on the ring, proceeding clock-wise; it is called deoxyribose, because it is structurally similar to ribose, except for the presence of a single

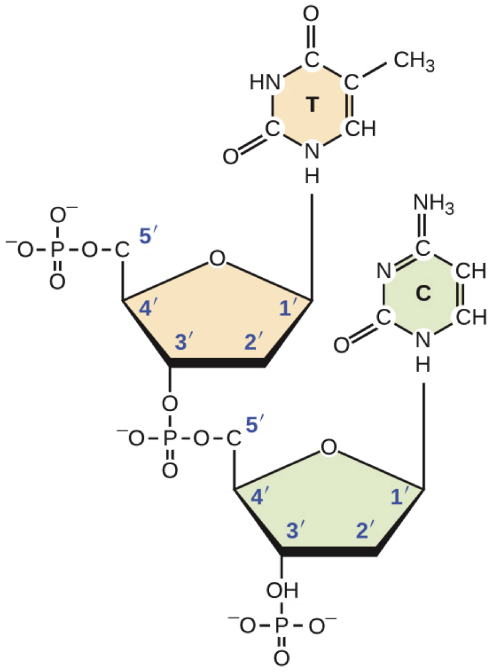

Figure 1.1: Two nucleotides

hydrogen atom, instead of an hydroxy group on the 2' ring position. The phosphate group is attached to the 5' carbon, replacing another hydroxy group, and, along with the deoxyribose constitutes the so called *backbone* of the DNA. The last group is the base which attaches to the 1' carbon atom of the deoxyribose and can be one of four different kinds: *adenine*, *cytosine*, *guanine* and *thymine*, usually shorten with their leading letter: A, C, T or G. A single strand of DNA can be seen as a long chain of nucleotides linked together through the 3' carbon, with a precise reading order, from the 5' position toward the 3' one. An example of how nucleotides bound together is shown in figure 1.1. The directionality characterizing the way bases are chained is important, as long as the sequence cannot be reversed: enzymes that attaches to DNA for replication or other in-cell processes always traverse the chain from 3' to 5'. The whole DNA molecule consists of a double

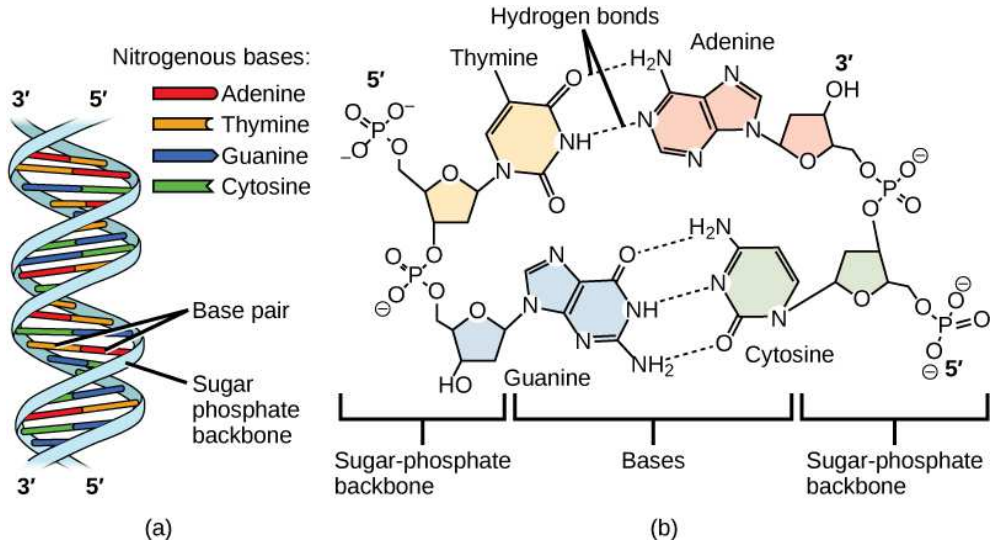

Figure 1.2: Example of DNA double strand

helix joining the two strands together, pairing the nucleotides base by base following a very strict rule: a molecule of *adenine* can be paired only with a *thymine* one and the same is valid for *cytosine* and *guanine*; chemically speaking, those are the only stable ways in which the pairs can be chained by effective hydrogen bonds. Last, the two strands run in opposite directions, but each of them is always considered from 5' to 3'.

The DNA is sometimes called the "code of Life" because the information central to our whole existence, from how we look like up to the way inner processes, taking place within our cells, are regulated is actually stored in the genome, namely the unique sequence of nucleotides composing our own DNA. Even the information about our genetic heritage have to be looked for within this context, as long as part of our genetic code is inherited by our parents, following the very same rules Mendel started devising during the middle of nineteen century[12]. This information flow is one-way only and was formalized in 1970 as the *central dogma of biology*[3]:

> "The central dogma of molecular biology deals with the detailed

> residue-by-residue transfer of sequential information. It states that such information cannot be transferred back from protein to either protein or nucleic acid."

At the molecular level the process of synthesizing a protein starting from a genome portion is called *gene expression* and is accomplished in two steps: *Transcription* and *Translation*. The Transcription is the mechanism that starting from a portion of DNA, builds a molecule of *messenger RNA*, or *mRNA*. RNA, which stands for Ribonucleic acid, is a nucleic acid like DNA, with some differences:

- It is a single stranded molecule.
- The sugar contained in its backbone is a ribose, having an -OH group in position 2' instead of a hydrogen atom.
- The complementary base to Adenine is a base called Uracil, which in DNA molecules does not exist.

The process is carried out by a group of enzymes which attaches to a particular region of DNA, called *promoter*, in charge of starting the transcription process lead by an enzyme called RNA polymerase by base complementing one of the two DNA strands, called *template*. Once the mRNA molecule is synthesized it flows out of the cell nucleus toward ribosomes, where the Translation process can occur. During Translation a set of enzymes translates each triplet of bases in messenger RNA into a specific amino acid; those are chained together creating the protein structure step by step until a specific triplet is reached, also known as *stop codon*. The correspondence between a base triplet and each amino acid is universal and it is at the core of the information transmission mechanism each living being can realize; this correspondence is better known as *genetic code* and is shown in figure 1.3.

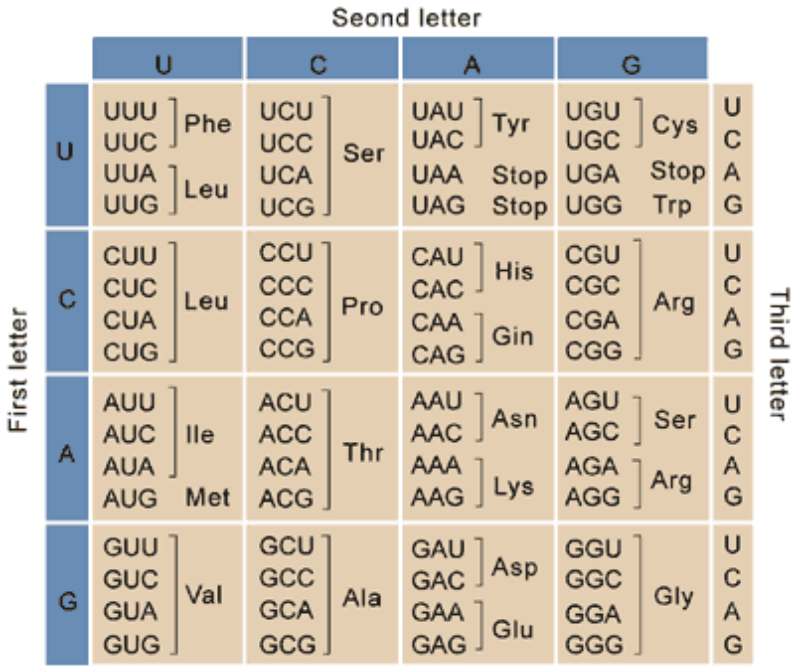

Figure 1.3: The genetic code

1.2 The sequencing process

Given the centrality of nucleic acids in the cell biology, most of the studies aiming at discovering or characterizing a certain process at the bio-molecular level cannot even start without discovering the sequence of nucleotides composing at least a portion of the DNA regions of the organism under study, or the RNA sequences it transcripts as part of that process. Most of the researches undertaken these days, focusing their attention on events taking place at the genic level, were effective in detecting which genes are most involved in the appearance of certain diseases, as well as discriminating between harmless mutations in the sequence of bases and dangerous one. The technological process aiming at determining the nucleotides composing a genome, or a part of it, is called sequencing; from the bioinformatics point of view, each sequencing technology can be characterized by two metrics: the length

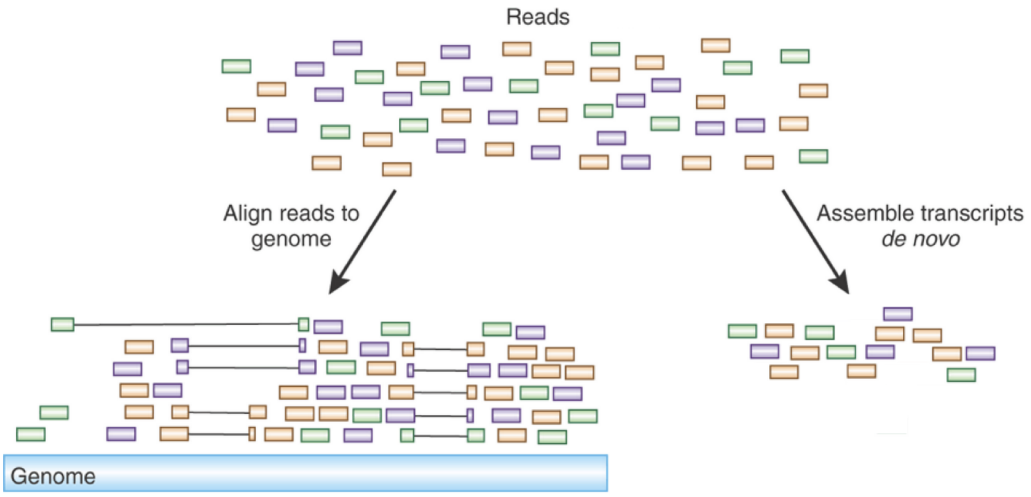

Figure 1.4: Assembly and Alignment bioinformatic pipeline workflow

distribution of the nucleotides fragments, called *reads*, the sequencing process is able to produce, and the amount of errors, and their model, it introduces during sequencing. The first metric has to do with the inability of any technology up to now to determine the whole sequence of bases composing a nucleic acid chain at once: even recent technologies, providing very long reads, can cover a minor fraction of the genome length even from small organisms such bacteria. The second metric is useful for measuring how much we can trust the data coming from the sequencing machine, when we have to decide what reads can overlap, or are most similar to a certain target chromosome portion.

There exists two families of sequencing projects that leads to different choices in the bioinformatic pipelines that have to deal with the data they generate: *de novo sequencing* and *resequencing*. De novo sequencing aims at determining the ordered sequence bases composing the chromosomes of an organism belonging to a species whose genome is unknown, or at detecting large variants in the genetic material of the target organism, such that exploiting the already known genome of its specie would be unfeasible

because of the differences. The bioinformatics problem behind de novo sequencing is called *assembly* and consists in reconstructing the genome of the organism under study by overlapping the reads given back by the sequencing machine for determining longer and longer fragments, until the whole genome is covered. Resequencing is the task of determining the genome of organisms belonging to species whose genetic material is already known and characterized. If the settings of the experiment suggests that few modifications can be expected between the sampled genome and a reference coming from any target of the same species we can avoid finding the best overlapping between all the reads, but looking for the best location where each read the sequencing machine provides can be mapped, over a golden-sequence called reference. This problem is solved using very different algorithms and data structures with respect to the assembly problem, and, in facts, is often referred to as *alignment*. Figure 1.4 shows a schematic representation of the two bioinformatic pipelines.

Several technologies, procedures and protocols followed one another over the years, each different from the others in their cost per sequenced base, throughput, error rate, average read length and chemical reactions involved in the process. These technologies, that start appearing in the last twenty years of the twentieth century can be roughly divided into three generation of sequencing machines, whose most important features will be discussed in the next chapter.

Chapter 2

Background

First and Second generation sequencing technologies were the main actors of the first fifty years of modern genomics from when Watson and Creek solved the structure of the DNA. These two technologies exploited different principles for biological sequences sequencing: while Sanger method read the nucleotides by chain-terminating them with a radio-labeled ddNTP, most of the so called Second Generation Sequencing technologies relied on a *wash-and-scan* approach, which worked by releasing in a solution containing DNA polymerase and biological samples a given amount of a kind of nucleobase, detecting if any incorporation takes place, generally with a camera and a light sensor, exploiting the ability of the chemical process to produce light. If no incorporation takes place the solution is cleaned up and a different kind of nucleotide is released in solution for being incorporated. Such approach allows a massive parallelism of the sequencing process, but allow sequencing only short read, due the fact that a sufficient signal-to-noise ratio is needed for the camera to detect the correct base incorporation. Third Generation sequencing technology uses partially or totally different chemical and physical principles for the purpose of sequence base-calling, which is the root cause

why such technologies produce a radically different kind of sequencing data with respect to previous technologies. The different kind of approaches born for solving the problem of aligning third generation sequencing long-reads have their roots in the novel properties the data generated by such machines; for better understanding this point a brief introduction regarding the main sequencing protocols used by the novel generations of machines is due, before start describing new algorithms for read mapping. Instead, details regarding how first and second generation sequencing machines work can be found in appendix A.

2.1 Third generation sequencing

The introduction of second generation sequencing technologies, allowing massive DNA sequencing for a few amount of dollars had a crucial impact on genomic research. However, the short length second generation reads are characterized by, was not suitable for handling some biological problems: detection of large structural variants in resequencing projects proved to be still difficult and de novo genome assemblies empliting short reads only may lack of entires genes and be extremely fragmented bringing the need for a new kind of machines able to overcome such limitations[14]. Defining a clear border between second and third generation technologies is not an easy task, and no straightforward answers can be provided, given the speed of developments in such field[7]. There exist two criteria for deciding whether a technology can be said to belong to third generation sequencing:

- Classifying technologies looking at their capability of sequencing whole molecules not halting the process between a base incorporation and the next, without the clonal amplification step, but still involving some

kind of enzymatic reaction[7][25].

- Classifying technologies looking at their capability of sequencing whole molecules, without involving any kind of enzymatic reaction[18].

A long debate could be settled up, but, for the sake of the following discussion, the former choice will be the preferred one; being bioinformatics a branch of computer science, it is more interested in the properties of the data it has to manage, than in their provenience or in the chemical reactions they are obtained by: choosing the latter option would mean excluding from the analysis a recent technology called *Single Molecule Real Time Sequencing*, that despite exploiting enzymatic reaction, produce a kind of reads whose length and error model are much more similar to the third than to the second generation sequencing standards. In general, we could say that bioinformatics introduce a third way of classifying sequencing machines, focusing on the kinds of data they provide: all technologies whose sequencing protocols come up with multi kilo-bases long reads, affected by a randomly distributed errors whose rate lies between 10% and 30% can be considered as third generation technologies, because the radically different property of the data they generate, with respect to previous protocols, needs to be explicitly taken into account when designing novel pipelines for managing them. The three main technologies compatible with the above requirements are *Single Molecule Real Time sequencing*, developed from Pacific Biosciences, available from 2012, *Nanopore sequencing*, an approach mainly exploited by Oxford Nanopore Sequencing up to this moment and *TruSeq Synthetic Long-Reads*, developed by Illumina and available from 2014, even if it is more a way of building longer reads up from an early assembly step of short reads, than a real long read sequencing technology.

2.1.1 Single Molecule Real Time sequencing

Single Molecule Real Time sequencing, SMRT in short, is a technology developed by Pacific Biosciences in 2012, aiming at sequencing a strand of DNA by direct observation of the base incorporation processes catalyzed by the polymerase enzyme during DNA template replication. The nucleotides present in solution are chemically modified such that fluorescence is released during the reaction once they are incorporated in the growing strand if excited by a proper source of light.

The Zero-Mode Waveguide structure

The process is based on a quantum phenomena taking place in a particular nano-structure called *zero-mode waveguide*, also known as ZMW, having cylindrical shape and being constrained to be smaller than 100 nm in its horizontal size. A molecule of DNA polymerase is confined at the bottom of its volume such that whenever a DNA fragment in added in solution, along with the correct chemical concentration of fluorescently-tagged nucleotides, it can start synthesizing DNA at a speed and processivity similar to those of in-cell DNA replication. Each time the a nucleotide is incorporated by the polymerase in the growing strand its fluorescent tag is released and excited by a laser beam from the bottom of the ZMW so that a certain amount of visible light is emitted and captured by a proper sensor on the top of the nano-structure; as long as each bases is tagged with a different fluorescent die, the sequencing process can take place, by registering what fluorescence is radiated at any time. There are two main factors which make this technology work: first of all, the fluorescence wavelength is similar in size to the ZMW diameter, while the laser one is much longer making the laser light intensity decaying exponentially, allowing it to illuminate only the bottom

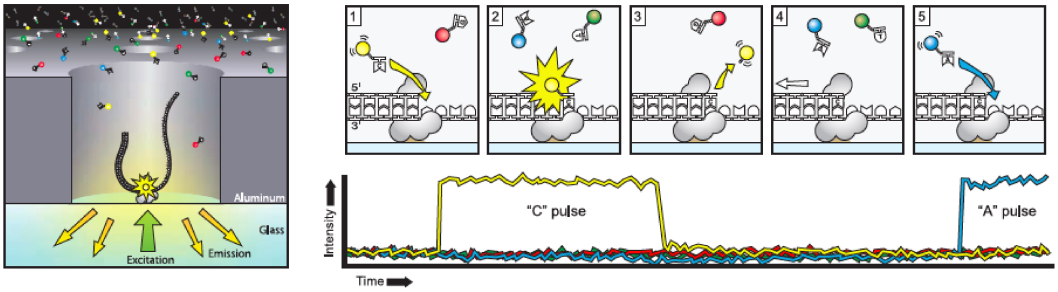

Figure 2.1: SMRT sequencing steps[5]

of the volume, where the DNA polymerase is, while fluorescences light can actually escape the ZWM being visible from the sensor posed at the top of the structure, with a signal-to-noise ratio sufficient for the purpose of base calling. The second crucial point stands in the kind of fluorescent dNTP used: in SMRT proprietary nucleotides the fluorescent dye is attached to the phosphate chain of the molecule, instead of to the base, meaning that once the polymerase incorporates the base, the dye is removed, and the strand remains as similar as possible as it would have remained in in-cell processes. Instead, in second generation technologies tagged fluorescent dyes remained attached to the base, forcing the polymerase to interrupt the reactions after few inclusions[7]. A graphic representation of the whole sequencing procedure is given in figure 2.1.

CLR and CCS reads

An interesting property of SMRT sequencing is that the technology has a built-in mechanism for trading-off reads length for accuracy. The template from which the DNA is sequenced is called *SMRTbell*, shown in figure 2.2, and consists of a closed single-stranded ring created by ligating hairpin adaptors at both ends of the double-stranded molecule to be sequenced. Because the *SMRTbell* forms a closed loop, once the DNA polymerase replicates the

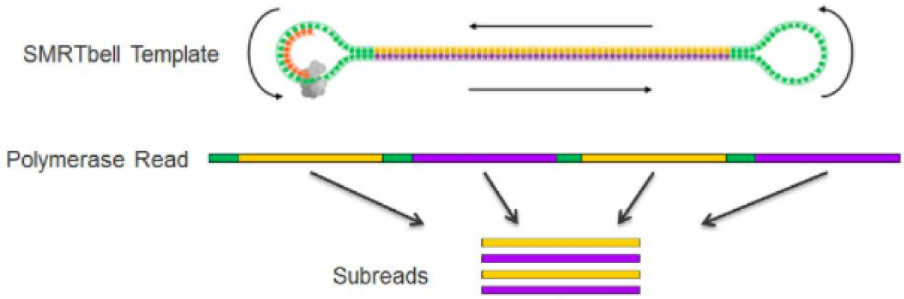

Figure 2.2: SMRTbell template[29]

forward strand it can potentially continue incorporating nucleotides, sequencing the hairpin adapter first and the reversed strand of the molecule then. If this procedure is repeated multiple time, a single ZMW can read the same sequence more than once, enabling the possibility of a built-in consensus step early in the analysis pipeline, increasing the accuracy of the reads being sequenced in such a way to more than 99%, depending on how many times the enzyme runs along the template: the greater the number of passes, the better the consensus accuracy is, as shown in figure 2.3. Each sequenced fragment can be interpreted as a *continuous long read*, also called CLR, if the polymerase travels only the first strand once, or as a *circular consensus sequence*, or CCS, if both strands are swept more than once. The trade-off between CCS reads and CLR stands in the fact that the DNA polymerase which replicates the template at the bottom of the ZMW has a limited lifetime: assuming that the enzyme runs along the template at constant speed and that the number of passes does not influence the polymerase lifetime, sequencing the same template multiple time necessarily constraint the library preparation chemistry in selecting shorter target fragments for the sake of sequencing[22].

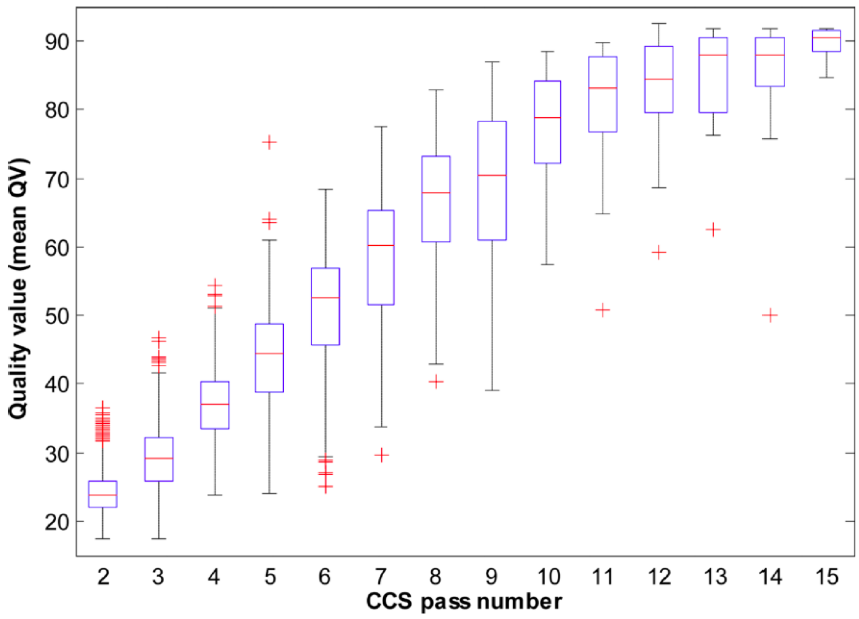

Figure 2.3: CCS accuracy from SMRT reads depending on the number of *passes*[13]

2.1.2 Nanopore sequencing

Nanopore sequencing is a technology powered by a radical paradigm shift in the way sequencing is conceived: instead of detecting the kind nucleotides chained within a DNA strand exploiting chemical enzymatic reaction, a simple current measurement is performed by taking advantage of a particular structure called *nanopore*. Such a sequencing strategy is extremely simple in principle, and, interestingly, was already proven to work in 1996, in the middle of first generation sequencing machines development, far earlier than the introduction of second generation sequencing protocols[7].

Nanopore structures

The key player for this approach is the *nanopore*, a nano-dimensional structure able to make molecules pass through a pore in its middle. Such unit

is embedded in a membrane, immersed in a salt solution, and is able to make ions flow through it, as an appropriate current is applied to the pore. Its key property is that whenever a charged bio molecule, such as a DNA fragment, start traversing the pore, an electrical resistance is introduced in the system and the current caused by the ions flow decreases, as shown in figure 2.4; moreover, differently sized molecules affect the current in various ways, such that, concerning DNA, it is possible to discriminate the type of nucleotide passing though the nanopore analyzing how the current evolves through time. Nanopore sequencing technologies can be classified in two families: those using biological nanopore and those exploiting state-of-the-art nano-technologies for developing a custom solid state structure[18]:

- Biological nanopore are easily modifiable and can be produced in mass remaining homogenous in size and structure. An example of molecule being used as biological nanopore is *α-hemolysin*, whose structure was proven to be effective in discriminating all the four DNA nucleotides after a genetic modification and the addition of an adapter molecule of *cycledextrin* on its top, needed for continuous base identification.

- Solid state technologies promise to enable the production of more stable nanopores, controllable in their size and length. An interesting material for such a purpose is *graphene*, exhibiting the possibility of adjustable surface properties and granting a greater potential for in-devices integration with respect to the biological counterparts.

Base-call procedure

A notable and already available on the market nanopore sequencing protocol is the one used by the MinION machine, a miniaturized 90g device produced

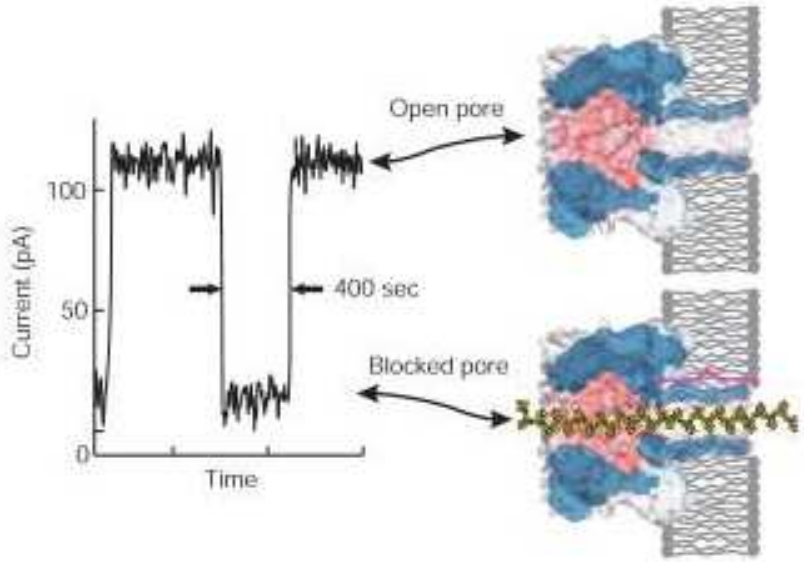

Figure 2.4: Nanopore structure, and current evolution

by Oxford Nanopore Technology. The library is constructed from double-stranded DNA by ligating two kinds of adapters at the molecule far ends: the first is called "leader adapter" and consists of two oligos forming a Y-shaped structure when annealed, while the second is referred to as "hairpin adapter". The actual sequencing process begins at the single stranded 5' end of the leader adapter. Once the double stranded region is reached, the DNA fragment is unzipped, allowing the first strand of the molecule to be passed into the nanopore one base at a time, while a sensor performs the current measurement described above, for guessing the bases being pushed through the nanopore. Once the second hairpin adapter is reached, the complementary strand is allowed to flow through the nanopore in the same fashion, actually sequencing the same molecule twice, first the forward strand and then the reversed one. The two strands sequenced separately are called *single-direction*, or *1D*, reads and are usually characterized by an average accuracy of about 70%. If the two *1D* reads have approximately the same

length a consensus procedure is performed, executing a *two-direction*, or *2D* base call: if the resulting sequence estimated quality is sufficient the so obtained read is marked as *"pass"*. A read is instead marked as *"fail"* if the *2D* does not result in a sufficient quality scored data, of if the *2D* base call is not performed. However both the classes of reads are reported to the end user[16]. A successful *2D* consensus procedure is expected to output data with an accuracy of about 15%, similarly to the output of CLR *Sigle Molecule Real Time* sequencing data. Concerning the read length distribution, most of the reads are multi kilo-bases long, with a percentage of extremely long data as expected from third generation sequencing technology, even if most of the reads are not that long, factor that have to be taken into account in designing tools for following analysis.

2.1.3 Illumina TruSeq Synthetic Long-Reads

Moleculo protocol, better known as Illumina *TruSeq Synthetic Long-Reads*, is a sequencing strategy that generates long reads, around 10 kilo base-pairs long by assembling short reads sequenced using a standard second generation Illumina machine. The key of this approach stands in the way short reads are bar-coded during library preparation, which consists of the following steps:

- DNA is fragmented into about 10 kilo base-pairs long fragments and appropriate amplification adapters are ligated
- Fragments are diluted into a 384-wells plate and a number of PCR cycles take place within each well; then, each sample is fragmented again and bar-coded

Once each well contains bar-coded fragments, about 600 base-pairs long, their content is pooled together and sequenced on a second generation Illumina

machine. The bar-code previously ligated is exploited for demultiplexing data coming from individual wells: once the source well of each read is traced back, short reads can be clustered by provenience and long reads can be assembled. The advantage of such approach is that the data produced are very precise, approaching 99.9% accuracy due to the intrinsic high level of accuracy of second generation protocols and the inner error correction step provided by the assembly tool. However, because this technology relies on long range amplification typical of second generation protocols the read produced are shorter than other third generation technologies and are subjected to biases in any region where the Illumina chemistry is biased[14]. Recently a drosophila melanogaster resequencing project showed the inability of such technology of exposing constant coverage, especially in regions characterized by high repeat content, where the synthethic long read assembly procedure is most likely to fail[17]. Moreover, high coverage sequencing projects exploiting this technology can be expensive: more that 900x short reads coverage may be necessary for reaching 30x synthetic long reads coverage[14].

2.1.4 Impact of TGS data

The most appealing feature of third generation sequencing is the ability of producing reads potentially more than tenths of thousand of base-pairs long. Recent de novo sequencing researches show a direct link between enhanced reads length and assemblies quality, leading to a better representation of genes, regulatory regions and other genomic elements, whose completeness is often a key trait for any subsequent study. In general, the quality of a genome sequencing project is evaluated by looking at the assembly results in terms of *contiguity*, *completeness* and *correctness*.

The capability of an assembly pipeline of assembling complete genome

elements, along with their surrounding context is called *contiguity*. For years *de novo* sequencing projects have been designed exploiting the studiy performed by Lander and Waterman in 1988, which starting from the average read length provided by the technology exploited and the sequencing coverage was able to provide a probabilistic expected value for the number of *contigs* obtained and their length. Unfortunately, these hypothesis proved not to hold for long and repetitive genome sequenced at high coverage, because in their formulation they did not take into account the presence of repetitive regions along genomes, that short read are usually not able to uncover. However, experiments show how the number of exactly repeated regions along a genome exponentially decrease with the length of the region. Figure 2.5 shows the impact of longer reads on assembly performance in function of the size of the genome studied: an average of 15 kilo base-pairs long reads is sufficient to almost perfectly assemble up to 100 mega base-pairs genomes and in general, longer sequenced reads always lead to longer *contigs*.

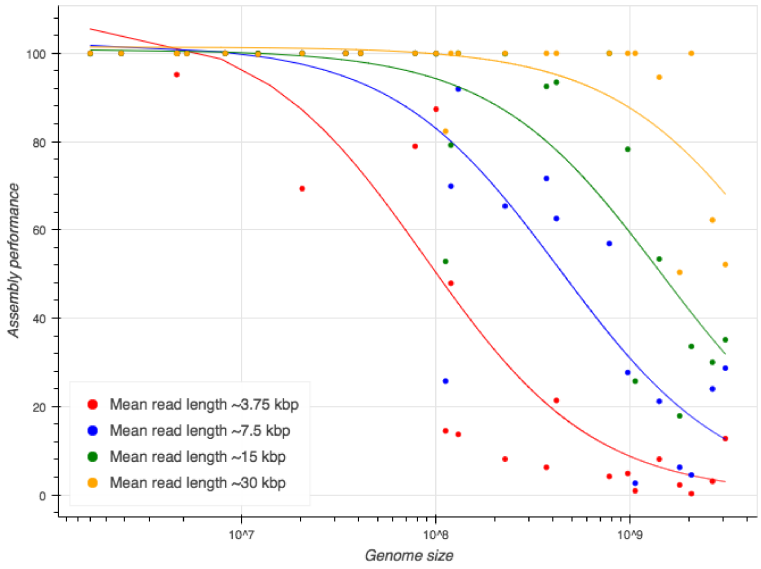

Figure 2.5: Assembly performance for a 20x coverage sequencing project[14]

Completeness is the capability of an assembled genome of representing each base of the original one; even if with a sufficiently high coverage each nucleotide in a sequence should be read at least one, usually assemblies size differ from the one of the target genomes due to artifacts introduced by assembly pipelines. Big differences between the real genome and the assembled one is very likely to cause issues in the subsequent analysis flow; figure 2.6 underline the fraction of currently known structural modification present in the Human genome that were detectable with previous version of the Human genome assembly, characterized by a shorter contig N50 size: the first Human genome draft coming from the Human Genome Project lacked of more that 10% of malicious structural variants associated with cancer or other genetic diseases. The promise of third generation sequencing is to provide the ability of assembling longer and longer *contigs*, for enabling the detection of even more complicated genome variants.

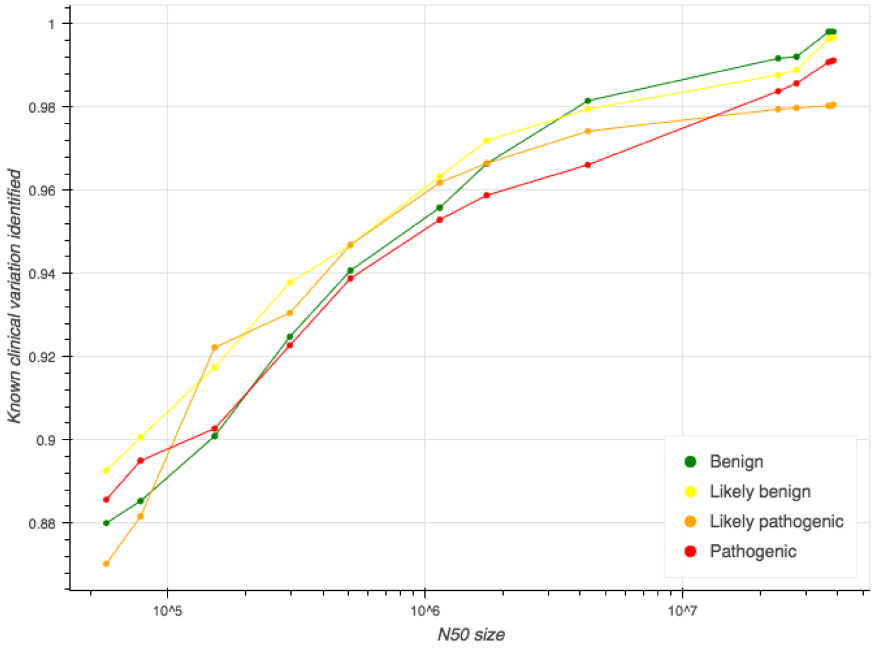

Figure 2.6: Human genome variants detectable with previous assemblies[14]

Correctness can be evaluated in terms of base-per-base accuracy the output sequence is able to grant; even if both *Single Molecule Real Time* and *Nanopore sequencing* machines an high error rate, sometimes over 20%, it does not seem to affect assembly performances: due to the random nature of errors distribution, a sufficiently high coverage can build accurate consensus sequences. Moreover some efforts were successfully carried out in developing error correction procedures, embedded in assembly pipelines or released as stand-alone tools, for forcing an accuracy comparable to previous generation data; two main approaches were proposed:

- Self-correction algorithms, where long reads are aligned one against the others and then polished with some consensus algorithm. The advantage of these techniques is that only one technology type is involved, and the random nature of errors along TGS reads makes them adapt for consensus procedures. The disadvantages are that these methods require a very high coverage, which could make the process unfeasible from the economical point of view; moreover, all-against-all overlapping makes these approaches intrinsically $O(n^2)$ in their algorithmic complexity.

- Hybrid algorithms, which increase long reads accuracy using more accurate short-reads coming from NGS data. In general, these techniques require a lower TGS coverage with respect to the former one, which makes this approach cheaper, but it can fail when applied to region of the genome not well-covered by second-generation sequencing data.

Both these approaches state the ability of assembly pipeline of dealing even with highly erroneous data, assuring that raw sequence error rate has little effects on assembled genome correctness.

2.2 Novel approaches for reads mapping

Alignment is one of the most critical task in bioinformatics: it is deeply involved in resequencing projects aiming at calling structural variants or determining SNPs but it is also used for evaluating reads overlapping in assembly tools. The core problem in aligning a *query* sequence against a *reference* is detecting the regions of the *reference* exposing a certain degree of similarity with the *query* according to some rules: such procedure is related to the origins and properties of the data it is applied to. The fact that sequences third generation technologies produce exposes radically different properties in terms of read length and error rate justifies the effort of investigating new data structures and algorithms for aligning such data. Before exploring the details of new approaches for long-reads alignment it is worth profiling two real datasets, obtained by PacificBioscience SMRT RS II machine and Oxford Nanopore MinION machine, for better assessing the property reads provided by the two most well-settled technologies in terms of read length and accuracy.

2.2.1 Assessing real datasets properties

Third generation long-reads can be generally addressed as multi kilo base-pairs sequences with an error rate between 10% and 30% depending on the technology exploited, with the exception of *Illumina TruSeq Synthetic Long-Reads* which are proven to be more than 99.9% accurate. Here, two publicly available third generation datasets will be analyzed, referenced in table 2.1 focusing on read length distribution and average accuracy, the two most important metrics when dealing with alignment. Unfortunately, as far as I know, no *Illumina* long-read dataset is freely available on the web,

so that technology is excluded from this more in deep investigation. Both the datasets considered come from sequencing runs on an *escherichia coli* sample and take advantage of recent library preparation procedures: the SMRT dataset exploited the *P6C4* chemistry, while the nanopore dataset used the novel *R9* cell for sequencing. Moreover both datasets were selected for exposing long and erroneous reads without any kind of post sequencing technology provided consensus step: the *PacificBiosciences* dataset is made of *CLR* reads, while the *Oxford Nanopore* one is made of single direction *1D* reads, for profiling data coming from the worst possible case for alignment pipelines. As shown by figures 2.7 and 2.9 both technologies tend to pro-

Technology	Process	Genome	Reference
SMRT PacBio	P6C4	E.coli K-12	[21]
Oxford Nanopore Technology	R9	E.coli K-12	[20]

Table 2.1: Datasets properties and references

duce reads exposing a log-normal length distribution, some of which longer of ten kilo base-pairs; however nanopore sequencing seems to allow extremely long reads, sometimes longer than hundreds of thousands of base-pairs, while SMRT sequencing reads don't exceed 50000 base-pairs length, in general, as shown by table 2.2. An interesting point is that both technologies produce data whose length is very heterogeneous: in both cases more than 5% of the sequences reported are shorter than one thousand base-pairs, suggesting that novel tools dealing with long-reads have to take into account the presence of such shorter data either tuning their algorithms accordingly, or providing some kind of hybrid pipeline, able to manage both short and long reads differently.

For evaluating the accuracy of the datasets, reads are aligned aganist a reference *escherichia coli* genome previously assembled starting from a Third

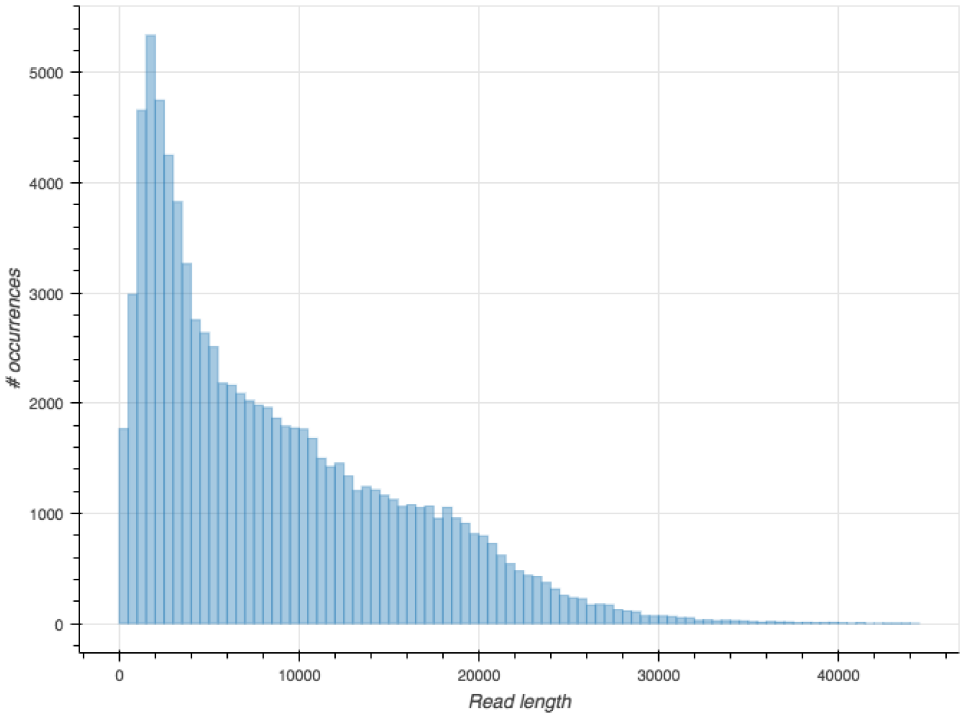

Figure 2.7: CLR SMRT reads, P6C4 chemistry

Threshold [bp]	SMRT dataset	Nanopore dataset
1000	94.568%	94.289%
2000	83.146%	82.947%
5000	58.595%	56.262%
10000	35.366%	30.917%
20000	7.899%	10.706%
50000	0%	0.579%
100000	0%	0.003%

Table 2.2: Fraction of dataset sequences longer than a given threshold

Generation dataset; the tools chosen for aligning the reads are BWA-MEM[8] and Minimap2[10]: BWA-MEM is a tool born for aligning Second Generation Sequencing reads, recently adapted for dealing with longer reads, while Minimap2 is a novel pipeline specifically designed for long reads mapping; both support a preset command line flag for aligning third generation datasets so

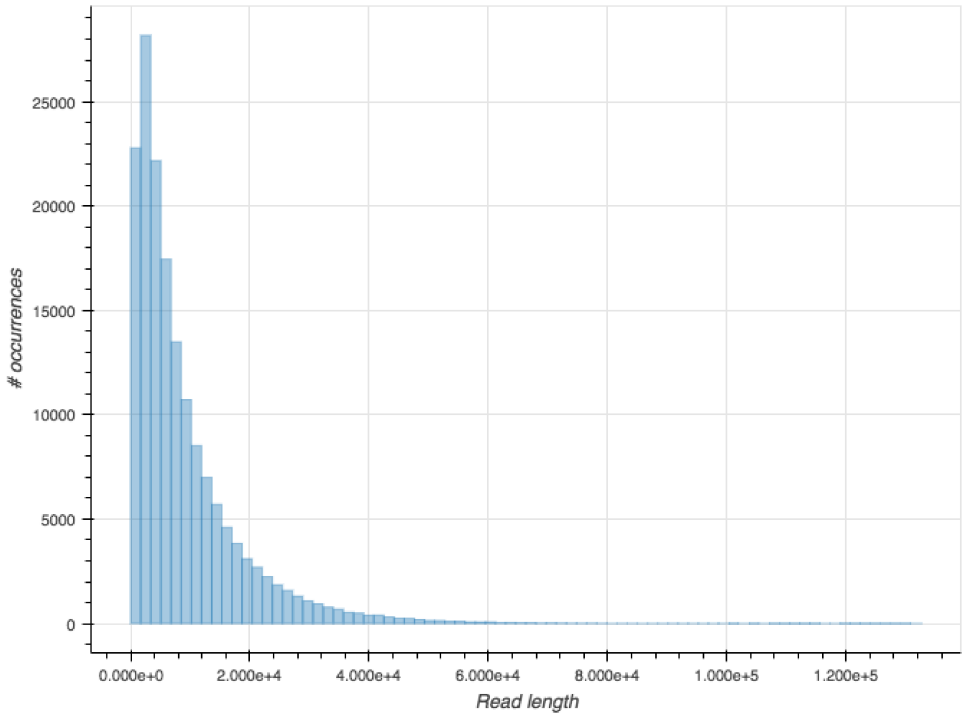

Figure 2.8: 1D ONT reads, R9 cell

no paramter tuning was performed before results evaluation. The rates of insertions, deletions and substitutions in the aligned reads were measured reading the *CIGAR* string and exploiting the estimated edit distance between the queries and the reference contained in the *NM* tag from the SAM file provided as output by both tools.

The results from the two aligner are presented in table 2.3 and are substantially concordant, confirming the assumption of the data average accuracy being comprised between 10% and 20%, underlining that *SMRT* Long-Reads are generally more accurate than those belonging from *Nanopore* datasets. Moreover, two facts worth to be underlined:

- Insertion, deletion and substitution rate are essentially technology dependent, meaning that any tool aiming at discriminating between those events during alignment for performing detailed base-per-base align-

		Error rates [%]			
Dataset	Tool	Substitution	Insertion	Deletion	Total
SMRT	BWA-MEM	1.9	7.2	2.6	11.7
	Minimap2	1.7	8.0	2.7	12.4
Nanopore	BWA-MEM	7.4	2.7	7.7	17.8
	Minimap2	6.2	3.3	8.3	17.8

Table 2.3: Different error rates devised by BWA-MEM and Minimap2

ment need to parameterize at least the dynamic programming scoring scheme with respect to the technology the reads were produced by.

- Even if accuracy is higher than 80% in the average, part of the reads in any dataset are likely to be perturbed by an higher error making it necessary, for tool aiming at aligning the majority of the reads, to tune algorithms for working with data exposing an error rate higher than the average.

2.2.2 Long-Reads mapping strategies

The most used and studied approach in bioinformatics for sequence alignment is dynamic programming. This approach is well-characterized from the mathematical point of view, several algorithms exist implementing such procedure and it proved to be extremely effective during the years. However, it suffers from a major drawback: it is extremely computationally expensive, making impossible to perform read alignment by simply applying dynamic programming to a read against the whole genome in a brute force fashion. That is the reason why almost all alignment tools work on two separate phases:

- A mapping step, where the reference regions exposing the highest simi-

larity with the query sequence are selected, using some kind of indexing data structure, allowing exact or inexact regions filtering. Such step is useful both for reducing the computational effort required by dynamic programming and because some tasks such expression analysis does not necessarily need the most suited location where a read can map.

- A detailed alignment step, where dynamic programming is selectively applied to previously selected regions, for obtaining a base-per-base alignment able to discriminate between single nucleotide variations from simple errors induced in reads by technology, before applying the final consensus step.

While the main instrument for detailed alignment remains the dynamic programming, Third Generation Sequencing opened the way to a plethora of novel approaches for read mapping with respect to the traditional seed-and-extend algorithms which are the most common approaches for selectively mapping short reads coming from previous generations technologies. There exist two families of approaches for performing long-reads mapping, discussed in the following sections: seed based algorithms and novel fingerprint based algorithms.

Seed based approaches

Seeding techniques consist in trimming long-reads into smaller seeds and resorting to traditional suffix-trie based index data structure for finding exact matches in the reference. Once the matches are found, high similarity regions are discovered by seed match clustering, or by seed-extension procedure. Examples of tools exploiting such techniques for read mapping are *BLASR* and *BWA-MEM*.

BLASR[1] was designed in 2012 by PacificBiosciences for aligning long-reads up to 20% divergent with respect to the reference, assuming an error profile compatible with the one exposed by sequences sequenced by their proprietary *SMRT RS II* machine. Given a read and a reference, all exact matches of seeds longer than a given threshold are found exploiting suffix arrays based data structures. Exact matches are clustered by grouping seeds found in intervals roughly the length of the read long and selecting non-overlapping reads increasing in both query and reference position. Once the clusters are defined, they are assigned a score, depending on how frequent the anchor sequences are in the genome, and only the top clusters are retained for the successive detailed alignment procedure, which occurs in two steps:

- A Sparse Dynamic Programming procedure, which basically repeats the previous anchoring procedure on a smaller scale, looking for short exact matches between the query and the target reference region.

- A final Banded Dynamic Programming over a set subset of dynamic programming grid cells defined by the previous sparse procedure.

BWA-MEM[8] is a *seed-chain-extend* tool designed for aligning second generation reads longer than few tenths of bases, but actually said to be able to manage up to one million base-pairs sequences. It is reported here because it is used as benchmark in various Third Generation mapper tools, and it supports command line flags for making it deals with long not accurate reads coming from third generation sequencers. It exploits Burrow-Wheeler transform for finding *super-maximal exact matches* between queries and reference. Exact matches co-linear and close each other are then greedily chained for make easier the last step of the algorithm, where each seed is sorted first by the length of the chain it belongs to and then by its length and extended

running a banded dynamic programming procedure.

Fingerprint based approaches

The main issue when dealing with seed based procedure is choosing the most appropriate length for the seed: short sequences increase the probability of the seed being a error-free copy of a certain portion of the reference, but leads to a potentially high number of matches on the reference, requiring a considerable amount of efforts for being analyzed; on the contrary, longer seeds exponentially reduces the exact matches on the target sequence, but make the probability of the seed containing a technological error or a biological mutation higher. Dealing with long and erroneous reads make this trade-off more difficult to manage, which is the reason why new "non seeding" mapping approaches were investigated for avoiding such issues. The main principle behind novel mapping approaches is to give nucleotides sequences an alternative representation, from now on called *fingerprint*, resorting to features extraction algorithms for defining a proper way of deciding weather a query is related to a certain target by looking at their *fingerprint* similarity. There exist two family of approaches in fingerprint based mapping procedures known in literature up to this moment:

- Procedures that defines an algorithm for computing fingerprints along with a function of similarity, able to decide how much two fingerprint are related: an example of tool implementing such approach is MashMap.

- Algorithms that implement a chaining-like procedure on fingerprints instead of on seeds, by looking at features in common between query and reference fingerprints: tools implementing such approach are Minimap and its recently developed successor, Minimap 2.

The core strategy for sequence fingerprinting is translating an array of nucleotides into an array of features, in general represented as integers. An algorithm very commonly exploited in already published tools is called *Winnowing*: it was developed in 2003 in the context of document copy detection and defines a rule for choosing the most significant elements among a set of features[26]. A simple way of transforming a sequence into a set of features is to split the sequence of bases into overlapped sub-strings of length k, called *k-mers*, and hashing them resorting to any kind of hash function. Assuming that a sequence is already been transformed in such a way the *Winnowing* procedure retains only the *k-mer* characterized by the smallest hash value among the ones belonging to a sliding window whose dimension is a parameter of the algorithm; if there is more than one hash with the minimum value, the rightmost occurrence is selected.

The rationale behind such approach is that no substring match shorter than the *k-mer* size is detected, which makes the k parameter acts as noise threshold; on the other hand, setting to w the size of the window for selecting minimum hash values guarantees that query and reference fingerprint share at least one *k-mer* if they expose a substring match longer than $w + k - 1$, making possible to establish a guaranteed sensitivity threshold. The winnowing algorithm is well characterized from the mathematical point of view: dealing with sequence fingerprinting, the most important parameter is the fingerprint density, meaning the number of *k-mer* selected by the procedure, with respect to the original document set, containing all the *k-mers* from the original text; Winnowing density expected value is expressed by equation 2.1.

$$d = \frac{2}{w + 1} \tag{2.1}$$

A further method for sequence fingerprinting is the one exploited by a

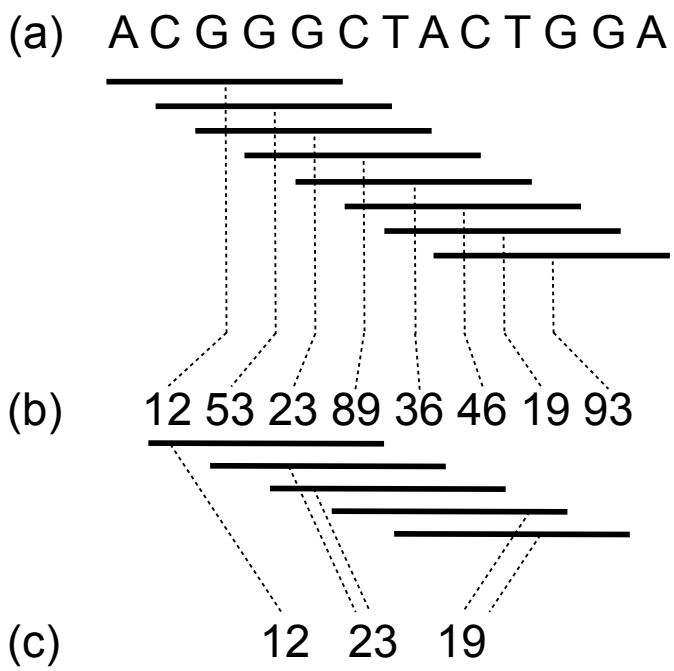

Figure 2.9: The Winnowing algorithm step by step: (a) the original sequence is split in overlapping 5-mer words and (b) hashed; the *Winnowing*, sub sample the *k-mer* with the minimum hash in a sliding window made of 4 elements

tool called *COSINE*, which, choosing a given *k-mer* size, translate a sequence into an array of integers counting the number of occurrences of each of the possible 4^k *k-mer* in a fixed sized sliding window. Even if such procedure is not spread like the previous *Winnowing* algorithm, it is worth to be reported as it represents a real break with any past seed based method: *Winnowing* based fingerprints lead to different approaches than seeding alignment tool, but still rely on a kind of seed exact match as long as, supposing that the function used for *k-mer* hashing does not yield collisions, it can be considered as a loose exact seed match approach which does not detect every seed in common between query and reference sequences, but is limited to detect one

seed match per window size. Instead, sequence content based methods rely on the hypothesis that *k-mer* distribution is a more robust way of determining sequence similarity.

MashMap[11] is a mapping-only tool for finding similarity regions between queries and reference without providing a detailed base-per-base alignment. The core idea of the algorithm is that similarity between two sequences can be computed considering the sequences fingerprints as set of hash values, whose similarity can be expressed by the *Jaccard distance*, a well-known metric for measuring similarity between sets, reported in equation 2.2.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{2.2}$$

Moreover, *MashMap* analyzes the problem within a statistical background, where the technological error is supposed to be Poisson distributed, such that the expected value for the *Jaccard distance* between a sequence and its copy perturbed by a given error rate ϵ is proved to be predicted by equation 2.3.

$$\mathbf{E}(J) = \frac{1}{2e^{\epsilon k} - 1} \tag{2.3}$$

In this context, mapping a query over a long reference means computing the fingerprints for the two sequences and finding the regions of the reference fingerprint exposing a *Jaccard similarity* greater or equal to the expected one, except for a security margin used for taking into account an error rate slightly bigger than the theoretical one. The best mapping positions look-up is made faster by a preliminary indexing step, where the reference fingerprint is stored in an hash table, using the hash value of the *k-mers* considered as key and the list of positions in the fingerprint where such *k-mers* appear as value.

MiniMap[9] tool was proposed in 2017 in the context of a project for the design of error correction-free de novo assembly tool, called *Miniasm*, as an efficient erroneous read mapper for *PacificBiosciences* and *Oxford Nanopore Sequencing* data. It relies on fingerprints calculated exploiting the *Winnowing* algorithm, and index the *k-mer* hashes approximately in the same way as *MashMap*, but instead of defining an inter-fingerprints distance function, applies an heuristic chaining procedure to hash matches between queries and reference, clustering exact matches which are approximately co-linear. Two matches m_1 and m_2 are said to be co-liner if, naming i_1, i'_1, i_2 and i'_2 the offset matches m_1 and m_2 belong to, on the query and on the target, the equation 2.4 holds, being ϵ a fixed constant. Being designed in the context of a de novo assembly, this tool does not provide any detailed alignment step, as it is not required, usually, in read overlapping phase of assembler tools.

$$|(i_1 - i'_1) - (i_2 - i'_2)| < \epsilon \tag{2.4}$$

Comparison

MashMap paper[11] presents a set of data obtained by running the four tools previously described on two real long-reads datasets, which are useful for better devising the impact of the different mapping approaches both on computational metrics, such as runtime and memory usage, and on biological mapping precision. The two dataset used are:

- dataset N1, containing 30000 reads sampled from a *Oxford Nanopore* machine sequencing run on a *escherichia coli* K12 sample.

- dataset P1, containing 18000 reads generated by a a *PacificBiosciences* cell sequencing the human genome.

		Index [s]	Map [s]	Memory [MB]
N1	MashMap	0.5	54	17
	Minimap	0.7	37	232
	BWA-MEM	2.6	20340	72
	BLASR	1.3	37020	697
P1	MashMap	352	84	37888
	Minimap	187	116	69632
	BWA-MEM	4740	24360	56320
	BLASR	2436	74400	180224

Table 2.4: Different state-of-the-art tools performances, as claimed by [11]

As reported in table 2.4, for the purpose of mapping, both the fingerprint-based approaches presented so far proved to be much more computational efficient than *BLASR* and *BWA-MEM*: the amount of time they need for building their own custom index and for mapping is at least two order of magnitude smaller than the one required for building traditional full-text indexes; on the other hand the peak amount of memory used seem to be more tool-dependent, not taking into account weather the mapping strategy relies on fingerprints or exploit seeding techniques. Interestingly, such speed-up is claimed to come at almost no cost with respect to the ability of *MashMap* and *Minimap* of mapping reads correctly; the recall statistics presented in table 2.5 measure the number of queries whose mapping predicted position error is comprised between $\pm 50\%$ of the query length with respected to the mapping position predicted by *BWA-MEM*: it shows that both fingerprint based methods are almost always able to discover the correct target position the queries come from. The main drawback exposed by fingerprints methods is their tendency in generating false positive mapping, especially in long repetitive genomes like the Human one. However, such inconvenience cannot shadows the impressive speed-ups granted by such approaches and that is

the main reason why different approaches for fingerprinting strategy worth be investigated.

		Recall [%]	Precision [%]
N1	Minimap	99.87	94.32
	MashMap	100.00	94.39
P1	Minimap	98.70	30.34
	MashMap	96.80	84.59

Table 2.5: Recall and precision statistics taken from [11]

Chapter 3

Methods

Even if Third Generation Sequencing is a relatively new and still in development technology, different kind of approaches have been proposed for aligning long noisy reads typically produced by novel machines. Such *fingerprint* based approaches promise to introduce a new effective way of dealing with biological sequences, but are relatively new and still not well-studied.

Here two different approaches are investigated: the former exploits hash tables for building *k-mer* based indexes and implements a chaining procedure already proven to work by other tools, trying to make it even faster exploiting an heuristic procedure able chain a set of exact hash matches in a single pass in the best case, assigning each chain a score emulating the way dynamic programming scores base matches. The latter approach exploits a radically different strategy: it aims at using *Bloom filters* for compressing sequence fingerprints, leading to a still more succinct representation of sequences, with respect to the one provided by ordinary *Winnowing* based methods.

3.1 Approximate *k*-chaining

Chaining algorithms are common procedures already exploited in alignment tools dealing with short reads: they relies on the assumption that if two exact seed matches coming from the same read are co-linear, then the region they span can be considered a similarity region between the reference and the read being aligned against it; as long as a sequence fingerprint computed through the *WInnowing* algorithm can be seen as a reduced set of the seed composing the original read, such approach can be easily extended from seed-like algorithms to fingerprint based approaches, by substituting the concept of exact seed match with the one of exact hash match, between features coming from the sequence and the reference fingerprint. The procedure proposed here consists in three different steps: reference fingerprint indexing, exact matches discovery and matches chaining.

3.1.1 Reference fingerprint indexing

The indexing algorithm used here is the same exploited by *MashMap* and *Minimap*, as it proved to be effective for the sake of long fingerprints indexing: the *Winnowing* algorithm can be slightly modified such that a sequence fingerprint not only keeps track of the *k-mer* hashes sequence, but also of the position in the sequence each hash comes from; is so, an easy way for fingerprint indexing consists of an hash table where the fingerprint hashes represent the keys of each entry, while the set of position in the reference where each hash appear is the value of the entry. For the purpose of approximate k-chain, such approach is followed, computing an index table for each sequence contained in the reference input file. The function used for hashing is called *ntHash*[19]: it is a well-known function designed to be faster

than general purpose hash function when dealing with biological sequences and allowing the rolling computation of successive *k-mer* hash values, for the sake of speed, even if such property is still not included in the current version of the tool used for performance assessing.

3.1.2 Exact matches discovery and *k*-chaining

Once the indexes are built, batches of reads are recovered from the disk and mapped. As long as queries and reference must have the same representation, first of all any read is fingerprinted with the same *Winnowing* parameters used for reference fingerprinting; then, every index is queried with each of the hashes composing the read fingerprint: querying a fingerprint index with an hash value results in the creation of a number of two-dimensional euclidean-like points equal to the number of times the hash value used for querying the index is present in the reference fingerprint. X-coordinate of such points represents the offset in the reference where the hash can be found, while y-coordinate represents the offset where the hash is found in the query: such situation is described in figure 3.1

If this procedure is repeated for each of the features composing the read fingerprint a number of match points are discovered, each of which represents a particular exact *k-mer* match between one of the query features and one of the reference ones; every point defined in this way belong to a region of the first quadrant of the Cartesian plane, delimited by the reference size on the x-coordinate and by the query size on the y-coordinate, that from now on will be called *query-reference-space* for convenience. This situation can be visualized in figure 3.2.

The key assumption behind heuristic chaining is that there exists a proper set of parameter for computing the sequences fingerprints such that high

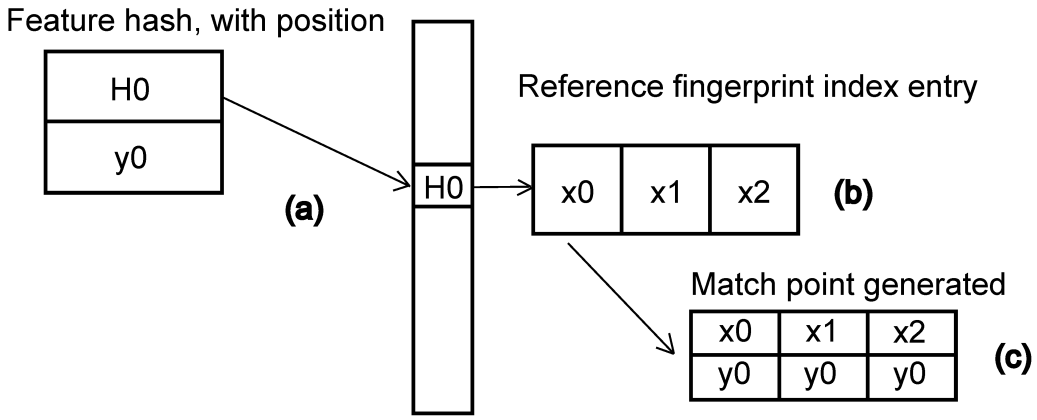

Figure 3.1: Schematic of the procedure for match points generation: each read feature hash is searched in the reference index (a), then the offset in the reference where such hash can be found are retrieved (b) and finally the match points are build, one for each x-coordinate retrieved from the index.

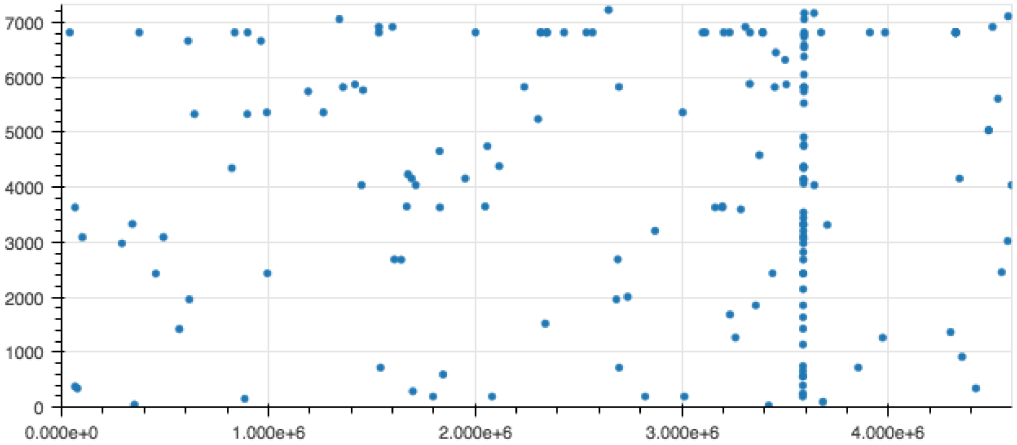

Figure 3.2: Real example of points disposition in the *query-reference-space*. The configuration results from a 7000 base-pairs long read fingerprint matched against an *escherichia coli* reference fingerprint.

similarity regions in common between queries and reference present a easily detectable pattern of co-linear set of points, like the one visible in figure 3.2, and that such pattern can be detected by simply scanning the set of matches once. For this purpose all the points found by querying the index are sorted first by their position on the reference and then by their position on the

query; then, the set of sorted matches is scanned linearly: for each point the linear chaining procedure decides whether it can be chained with point next to him by assigning the chain a score. Given two points P_0 and P_1, with coordinates $P_0(reference_0, read_0)$ and $P_1(reference_1, read_1)$ they are considered co-linear and joint together if the score computed with equation 3.1 is greater than zero.

$$\begin{cases} \Delta_{reference} = |reference_0 - reference_1| \\ \Delta_{read} = |read_0 - read_1| \\ Score = min(\Delta_{reference}, \Delta_{read}) - |\Delta_{reference} - \Delta_{read}| \end{cases} \tag{3.1}$$

The rationale behind the previous relationship is to assign scores similarly to dynamic programming algorithms do: the two match points are equivalent to a single base match and the algorithm guess that the maximum number of base matches are comprised in the interval between the two points; the score is then adjusted by applying a penalty to the score if the two matches are not exactly co-linear. When all the points are analyzed once, the original set of matches is transformed into a number of scored chains, and a certain amount of points the algorithm was not able to join together. For the purpose of mapping, only the chains, whatever their score is reported for further detailed alignment procedures.

This procedure reaches the best efficiency in term or run-time when each match is tried to be chained only against the match immediately successive, given the order the points are sorted according to. However, there exist certain points configuration which make this procedure never chain two points together, like the one shown in figure 3.4. In that case point A cannot be chained with point B, as long as the chain score would be negative; even if

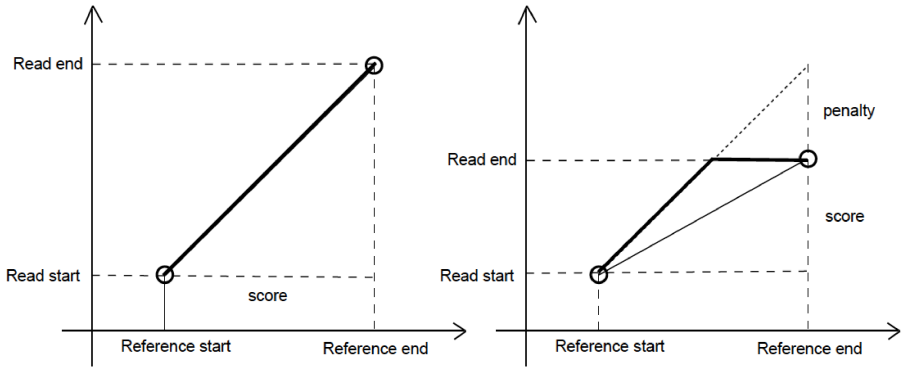

Figure 3.3: Two examples of score

point A is perfectly co-linear with point C, those two cannot be chained together, because A analyzed only the point immediately after itself according to the sorting criterion previously defined. Figure 3.5 remarks the differences between 1-chain, 2-chain and 3-chain procedure. Such events becomes particularly frequent when the number of points in the *query-reference-space* starts growing up, which happen if wrong parameters are chosen for the preliminary fingerprint procedure or for very long reads. However, solving such issue is easy in principle: it would be sufficient allowing the chain procedure to analyze the k points successive to the target match point analyzed instead of just the next one, transforming the chain procedure into a more general k-chain procedure. Figure 3.5 shows the differences between the results obtained by *1*-chain, *2*-chain and *3*-chain procedure for a particularly difficult to manage points configuration.

3.2 Bloom filters for fingerprint compression

All the tool discussed up to now proved that sequence fingerprinting is an effective way for aligning long noisy reads; the reason behind their run-time efficiency is that sequence fingerprinting acts as a sort for *lossy* compression

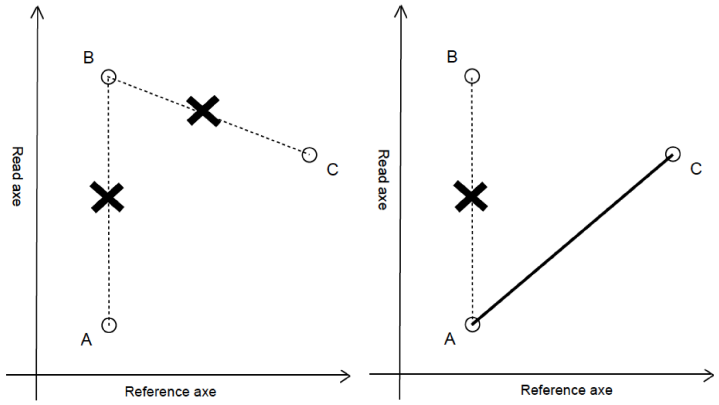

Figure 3.4: An example of point configuration where *1*-chain procedure fails.

step, allowing for building a concise representation of the features of a sequence of nucleotides, which still contains a sufficient amount of information for the purpose of similarity regions discovery. When dealing with very long data, the amount of fingerprint hashes to deal with can increase, demanding for solutions able to compress such set of hash features in a more concise representation.

3.2.1 Bloom filters

Bloom filters are data structure for space efficient representation of sets of objects. They allow to fast check element membership, granting that no false negative occurs but allowing a certain probability of false positive as a trade-off for space efficiency. *Bloom filters* can be implemented in different ways and many variants exist, but for the sake of this discussion we will focus on basic *Bloom filters*, which represent a set as a bit array and support two basic operations:

- Insertion: inserting an element in a basic *Bloom filter* means setting a certain fixed number of its bits. Given an object and a set of n hash

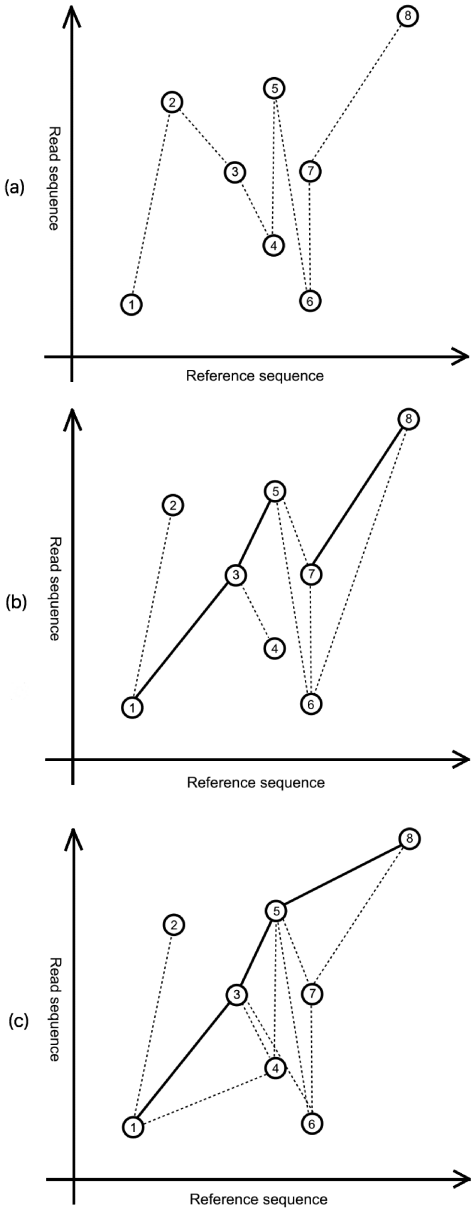

Figure 3.5: Difference between (a) *1*-chain, (b) *2*-chain and (c) *3*-chain.

functions, item insertion is performed by hashing the element to be inserted once for each of the n hash functions and setting the bits in the filter whose indexes is equal to the hash value computed by the functions.

- Query: given an element and a set of hash functions element insertion is performed by hashing the object to be inserted once for each hash function and checking if the bits whose indexes is equal to the hash value computed by the functions are all set in the filter.

Figure 3.6 represent the two previous operations: the filter consists in an array of 18 bits, and each item insertion sets three of the filter bits. Elements x, y and z are successfully hashed and inserted in the filter. The test for verifying the presence of element w in the filter fails, because one of the bits supposed to be set, the one in position 15, is not. Notice that as long as basic *Bloom filters* don't support item deletion, no false negative can happen, because once a certain bit cell is set, it will never be cleared.

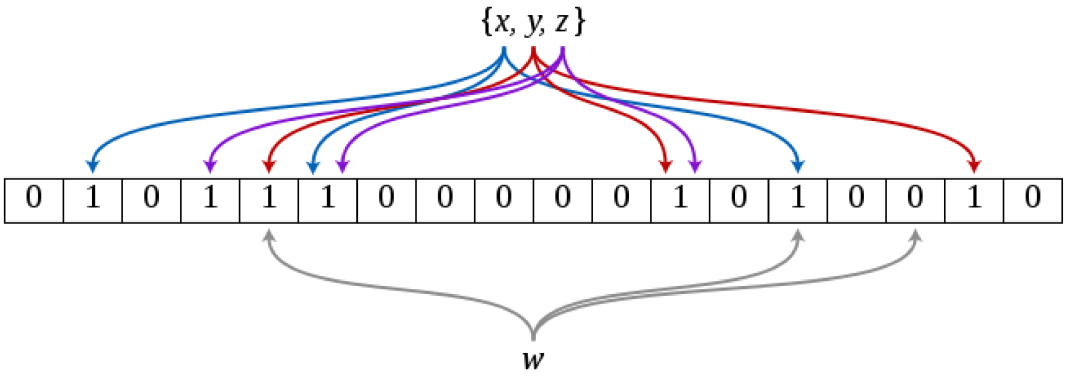

Figure 3.6: An example of items insertion and search in a *Bloom filter*.

What makes *Bloom filters* useful for read mapping is the capability of expressing how much two sets of objects are related using equation 3.2, already

proved to work in the context of digital forensic[23].

$$Score(f_1, f_2) = \frac{e_{12} - E_{min}}{E_{max} - E_{min}} \tag{3.2}$$

where e_{12} represents the number of bits in common between the two filters, E_{max} is the maximum number of set bits in common and E_{min} is the number of bits expected to be in common by chance, expressed by relationship 3.3.

$$\begin{cases} p = 1 - \frac{1}{m} \\ E_{min} = m(1 - 2p^{ks} + p^{2ks}) \end{cases} \tag{3.3}$$

where m is the filter size, k is the number of bits set for each insert and s is the number of elements each filter contains, that for our purpose is assumed to be fixed at run time.

3.2.2 Mapping procedure

The rationale behind Bloom filter fingerprint compression is to represent a reasonably large number of sequence fingerprint hash values as a Bloom filter: first of all a sequence is first fingerprinted exploiting the usual *Winnowing* algorithm already described, then each of the features composing the fingerprint is inserted in a Bloom filter of capacity c: whenever the capacity is reached, a new empty filter is created and filled. Assuming an input fingerprint made of n features, this procedure outputs an array of $\lfloor n/c \rfloor$ *Bloom filters*, representing the compressed version of the original fingerprint.

The mapping procedure is performed by computing the cross-correlation between the reference sequence of *Bloom filters* and the query one, substituting the multiplication operation with the *Bloom filters* similarity score expressed above. For formally, given the reference sequence of *Bloom filters*

R, of length $|R|$ and the query sequence Q made of $|Q|$ filters, the mapping score in position i is expressed by equation 3.4.

$$MapScore(j) = \sum_{i=0}^{|Q|} Score(R(i+j), Q(i)) \tag{3.4}$$

The previous situation is graphically represented in figure 3.7; once the *map score* for each reference fingerprint position is computed, the top *best* positions are retained and forwarded to detailed dynamic programming step for further refinements.

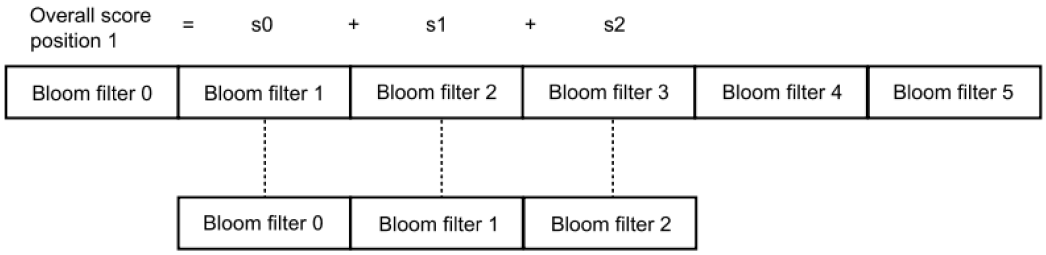

Figure 3.7: Example of map score computation of a query over a reference position

Figures 3.8 and 3.9 represents the map qualities of a query over the reference for all possible mapping positions: it can be seen that very different results are obtained for different sets of parameters, underlining that the fragility of such approach with respect to the input data has to be deeply investigated.

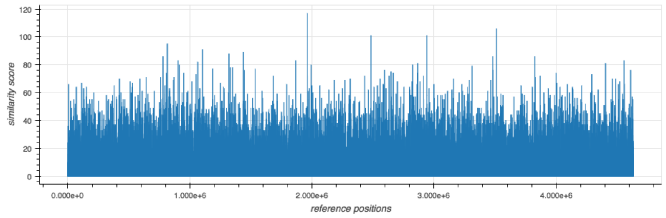

Figure 3.8: An example of map quality track.

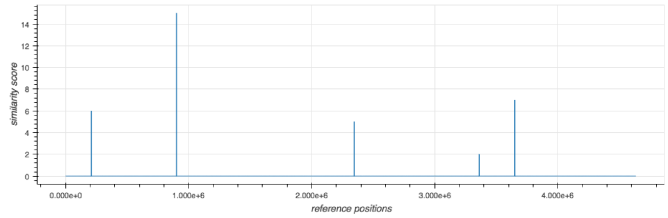

Figure 3.9: A further example of map quality track.

3.3 C++ implementation

Both the approaches previously described were implemented in C++ using *SeqAn*, a library implementing data structure and algorithms for easily approach bioinformatics problems. Both the tools exploits the `ArgumentParser` class for command line parsing, and reads the reference and target sequences using the readRecords method provided by the library itself. Instead, all the classes and methods for computing sequence fingerprints, exploiting the *Winnowing* algorithm, and implementing Bloom filters has been previously implemented specifically in the context of this work.

The Bloom filter based fingerprint compression tool first of all reads all the sequences composing the reference genome to be analyzed, exploiting the facilities provided by the SeqAn library, and computes the *Winnowing* sequence of hash value for each sequence, first, and then inserts group of n of such values in Bloom filters, where n is a parameter passed through the command line. Once the reference manipulation is finished, the program read batches of reads from the source fasta file exploiting the capability of the readRecords function to accept a parameter specifing the size of the group of reads to be read, compute the *Winnowing* sequence and compresses it in Bloom filters, as explained for the reference. Then, cross-correlate each read against the reference using a simple for-loop. This operation is made parallel exploiting openMP, an high level library for multi-thread programming in

C-like languages. As a last step, the top scoring position in the reference are selected and reported to a file formatted as PAF file, a known format for reporting read mapping, more concise and human-readable with respect to other formats such as SAM.

The heuristic k-chain procedure has been implemented in a similar way: again, the reference sequences are read and *Winnowing*-fingerprinted, bu this time each of the hash values composing the sequence is stored in an index implemented as an `std::unordered_map<uint64_t, std::vector<uint64_t>>` , a data structure provided by the C++ standard library. The key of the unordered_map is the hash value of a certain *k-mer* over a sequence, while the value is the list of positions where it occurs in the reference, implemented as a vector of unsigned, for making the implementation easier. Whenever a new query sequence is read it is fingerprinted, and the set of match points is retrieved by looking in the `unordered_map` for every hash composing the fingerprint and building match points up from the offset in the query the hash comes from and the positions in the reference it appear. The chaining procedure is implemented as a for-loop which scans all the match points, again stored in an `std::vector<MatchPoint>` , and implement the best-effort chaining procedure.

Chapter 4

Results

For better devising the main characteristics of state-of-the-art tools and both the approaches we here propose, they were run on a set of different datasets for better underlining their weakness and their strengths. We used two families of datasets:

- synthetic datasets, created with a tool called SimLoRD[27], sampling real genomes at random positions and perturbing the so obtained sequences with different error rates, emulating the two most common third generation sequencing technologies: *SMRT* and *nanopore sequencing*. Eighteen datasets were simulated, each dataset containing 10000 reads, a constant error rate, 10%, 15% or 20% and a constant read length, 1000 base-pairs, 5000 base-pairs or 10000 base-pairs.

- real datasets, freely available on the web, again from both technologies. In particular, three real datasets were exploited:

  - one reporting 10000 reads from the novel *PacificBiosciences* chemistry, called *P6C4*.

- one reporting 10000 reads from the novel *Oxford Nanopore* cell R9, containing 1D sequences only.
- one reporting 10000 reads from the previous *Oxford Nanopore* cell R7, containing 2D sequences only.

The mapping results are evaluated according to three different metrics: *accuracy*, *precision* and *run-time*. Accuracy can be 0 or 1 for the single query, depending if the mapping positions proposed from the tool contain the correct location. The accuracy reported is the average of the accuracy for all the reads, transformed in parts per one hundred queries; basically it measures how good an algorithm is in discovering the correct position in the reference the data come from, and can be expressed by equation 4.1, where *corrects* are the number for queries correctly mapped and —*dataset*— is the overall number of reads in the dataset.

$$accuracy = \frac{corrects}{|dataset|} \tag{4.1}$$

Precision measures how good a tool is in reporting the lowest possible amount of false positive; it computed by formula 4.2, where *reported* is the overall number of mappings reported in the output PAF file.

$$precision = \frac{1}{reported} \tag{4.2}$$

The run-time is measured by a python script, and express how many seconds a tool needs for working. Whenever the mapping needs the construction of an index, the times for building the data structure and for mapping are reported separately. For the purpose of computing *accuracy*, a sequence is said to be correctly mapped if the predicted position is within half of its length from

the actual offset in the reference sequence it was sampled from, as long as mapping tools are requested to only approximately find the correct mapping position for a given read. All the following tests were run on a server with 2x Intel Xeon E5-2630v3 running at 2.40GHz CPU equipped with 128GB DDR3 RAM.

4.1 State-of-the-art tools

Three state-of-the-art tools, *BLASR*, *MashMap* and *Minimap2* were tested on a server with the previously described configuration, each of them run on 16 threads. As expected, fingerprint based tools, such as *Minimap2* and *MashMap* are, in general, faster than tools exploiting old-fashioned tree based data structures for exact matches discovery between sequences. However, two elements worth being noticed: first, for a complex genome such the Human one, the run time difference between *BLASR* and *MashMap* is smaller than the one claimed by the *MashMap* paper [11], where the tools were run on a single thread, meaning that for tools evaluation, understanding the way their run-time scales with respect to the number of threads used is crucial. Second, most of the time *BLASR* needs for aligning reads over the Human genome, is used for reference indexing, an operation that, given a set of samples coming from organisms of the same species, can be performed only once; for smaller genomes, instead, is the mapping operation the critical one.

All the three tools tested are able to grant a great accuracy, almost always over 98%, no matter the properties of the reads being mapped, but the two *fingerprint* based tools here show their major drawback: *Minimap2* difficulty maps reads 1000 base-pairs long, with an error rate approaching 20%, leaving 21% of the queries not mapped for the *Escherichia coli* genome. Such per-

centage grows up to more than 27% when dealing with the Human genome. On the other hand, *MashMap* exposes a different behavior, as long as experiences an accuracy decrease on the Human genome for such short and inaccurate reads while increasing its unmapped rate on shorter genome.

On the average, *BLASR* results show a lower precision than the one obtained by the other two tools. Probably that is a consequence of the fact it uses dynamic programming for refine mappings, as long as such procedure makes him able to discover also sub-optimal regions of similarity between query and reference. Notably, *MashMap* reports only the position exposing the best similarity with the reference, making it has 100% precision by definition; however, notice that that's not necessarily a good feature: in real mapping problems, the real position a read comes from is not known, and a tool able to report only the best similarity region may not be flexible enough for all the context where solving a mapping problem is required.

The three tables 4.1, 4.2 and 4.3 show the details of the results obtaining by the mapping tests carried out on the three tools discussed above, with the exact run-times and accuracy for each simulated dataset used for testing.

4.2 Novel approaches proposed

In the following section the results of the tests carried out on the two novel approached proposed during this work are presented.

4.2.1 Bloom filters for fingerprint compression

The Bloom filters based fingerprint compression method, which computes the most suitable mapping regions resorting to cross-correlation is able to provide high accuracy in general, given that the proper number of positions

in the reference exposing the highest correlation score are recorded. Even if the result table 4.4 reported shows that the approach can work in principle, its performances are not completely satisfactory: the sequence of Bloom filter a set of hash values is compressed into is not indexed, making the C++ implementation extremely slow, actually not usable for the purpose of mapping, even for very short genomes like the *Escherichia coli* one. There are two reasons why Bloom filter are not easily organized in indexes: first of all, they are not numbers, but long sequences of bits, then, it is not possible to look for Bloom filter only resorting to exact matches, but some kind of nearest neighbor search is needed, as long as the previously described procedure behind such approach assigns a score to pairs of Bloom filters, meaning that it is not only interested in fully similar bits vectors but also in more approximate matches. A further limitation, more easy to solve is that this method, as described here, compute the cross-correlation track between two sequences, and blindly reports the top k position with highest score, meaning that the precision of such tool is always exactly the reciprocal of the number of position reported, leading to a non-solvable trade-off between accuracy and precision. As a result, fingerprint compression through Bloom filters seem to be a promising but still raw approach, which need to be better studied for making it viable, first of all by introducing some kind of indexing strategy within the tool.

4.2.2 Heuristic *k*-chaining

The heuristic chaining procedure has been tested with a plethora of different parameter configuration, but the most promising three only have been reported:

- *K-mer* size = 12, *window* size = 12, $k = 50$, retaining only the chains

with score bigger than 1.

- *K-mer* size = 16, *window* size = 12, $k = 10$, retaining only the chains with score bigger than 1.

- *K-mer* size = 20, *window* size = 12, $k = 1$, retaining only the chains with score bigger than 1.

This approach seems to work very well for all the configuration proposed; in particular, table 4.5 shows that when sequence fingerprinting is accomplished with both *k-mer* size and *window* size equal to 12, the difficulties experienced by other tools in mapping relatively short and erroneous reads are overcame. Notice that such configuration is not extendable to all the datasets, because it yields a very low mapping precision on sequences longer than few kilo base-pairs, meaning that a suitable approach for mapping any kind of read length data in real datasets may be to define a set of configuration parameters and dynamically choose the most suitable one given the length of the read being mapped. The run-time performances of this tools can be increased by redesigning a proper data structure for sequence indexing.
In this moment, the C++ implementation of this approach exploits a data structure called *unordered map* from the C++ standard library, but this may not be the best solution, especially given the fact that tools such as *Minimap* uses a custom data structure, designed specifically for the operation the tool has to perform.

4.3 Tools performances on real datasets

As a last step, we compare the set of mapping positions discovered by the three state-of-the-art tools and the heuristic k-chaining approach on three

real datasets, for proving that our method is able to discover the same mapping positions of already proven to work algorithms. The Bloom filter fingerprint compressor implementation is not tested here, because it took too long to manage real datasets, especially for what concern *Oxford Nanopore* one, containing extremely long reads.

The following plots should be interpreted in the following way: each point is correspond to a read that the heuristic k-chaining approach maps on the reference at the position where the point is placed on the x-axis of the plot, while the other tool, maps at an offset equal to the y-coordinate of the point on the plot. Obviously, the most interesting points are the one over the diagonal, because they represents reads mapping position common to the two tools.

The results, shown in figures from 4.6 to 4.14, are extremely positive: they show that the proposed approach is able to recognize almost all the mapping position discovered by other tools on unknown real datasets. However, such result could have been devised by looking at the very high accuracy all the approaches showed during the testing phase of synthetic datasets.

BLASR

10k reads simulated			*escherichia coli*					
Dataset properties			Mapping performances			Runtime		
Technology	Length [bp]	Error rate [%]	Accuracy [%]	Precision [%]	Unmapped [#]	Indexing [s]	Mapping [s]	Total [s]
PacBio	1000	10	100,00	94,40	-	2,19	5,55	7,74
PacBio	1000	15	100,00	95,49	-	2,19	5,48	7,67
PacBio	1000	20	99,93	96,43	12	2,19	5,60	7,79
PacBio	5000	10	100,00	88,81	-	2,19	29,95	32,14
PacBio	5000	15	100,00	90,91	-	2,19	30,03	32,22
PacBio	5000	20	100,00	92,56	-	2,19	29,60	31,79
PacBio	10000	10	100,00	82,74	-	2,19	89,39	91,58
PacBio	10000	15	100,00	86,64	-	2,19	87,86	90,05
PacBio	10000	20	100,00	89,48	-	2,19	88,78	90,97
ONT	1000	10	100,00	94,59	-	2,19	5,48	7,67
ONT	1000	15	100,00	95,45	-	2,19	5,53	7,72
ONT	1000	20	99,94	96,44	27	2,19	5,61	7,80
ONT	5000	10	100,00	88,27	-	2,19	29,83	32,02
ONT	5000	15	100,00	90,56	-	2,19	30,39	32,58
ONT	5000	20	100,00	92,25	-	2,19	30,40	32,59
ONT	10000	10	100,00	82,51	-	2,19	88,08	90,27
ONT	10000	15	100,00	85,49	-	2,19	88,05	90,24
ONT	10000	20	100,00	88,45	-	2,19	88,15	90,34
10k reads simulated			*homo sapiens*					
Dataset properties			Mapping performances			Runtime		
Technology	Length [bp]	Error rate [%]	Accuracy [%]	Precision [%]	Unmapped [#]	Indexing [s]	Mapping [s]	Total [s]
PacBio	1000	10	98,75	52,90	-	3428,00	136,49	3564,49
PacBio	1000	15	98,30	63,38	1	3428,00	136,04	3564,04
PacBio	1000	20	97,04	75,11	45	3428,00	136,05	3564,05
PacBio	5000	10	99,19	30,07	-	3428,00	248,40	3676,40
PacBio	5000	15	99,25	41,07	-	3428,00	241,05	3669,05
PacBio	5000	20	98,97	57,47	5	3428,00	239,44	3667,44
PacBio	10000	10	99,43	21,20	-	3428,00	401,38	3829,38
PacBio	10000	15	99,23	31,14	2	3428,00	380,75	3808,75
PacBio	10000	20	99,21	47,42	11	3428,00	363,97	3791,97
ONT	1000	10	99,02	51,54	-	3428,00	137,67	3565,67
ONT	1000	15	98,03	61,73	-	3428,00	135,58	3563,58
ONT	1000	20	96,87	73,78	47	3428,00	134,10	3562,10
ONT	5000	10	99,28	27,97	-	3428,00	246,31	3674,31
ONT	5000	15	99,17	39,84	-	3428,00	239,95	3667,95
ONT	5000	20	98,69	56,69	5	3428,00	237,04	3665,04
ONT	10000	10	99,23	20,39	-	3428,00	403,51	3831,51
ONT	10000	15	99,11	30,42	1	3428,00	382,48	3810,48
ONT	10000	20	99,16	47,25	8	3428,00	366,98	3794,98

Figure 4.1: *BLASR* mapping results

Minimap2

10k reads simulated			*escherichia coli*					
Dataset properties			Mapping performances			Runtime		
Technology	Length [bp]	Error rate [%]	Accuracy [%]	Precision [%]	Unmapped [#]	Indexing [s]	Mapping [s]	Total [s]
PacBio	1000	10	99,98	98,98	-	0,37	0,18	0,56
PacBio	1000	15	99,65	98,93	29	0,37	0,17	0,55
PacBio	1000	20	97,28	98,48	2.123	0,37	0,17	0,54
PacBio	5000	10	100,00	99,71	-	0,37	0,43	0,81
PacBio	5000	15	100,00	99,68	-	0,37	0,42	0,79
PacBio	5000	20	99,99	99,63	-	0,37	0,37	0,75
PacBio	10000	10	100,00	100,00	-	0,37	0,74	1,11
PacBio	10000	15	100,00	100,00	-	0,37	0,72	1,09
PacBio	10000	20	100,00	99,99	-	0,37	0,66	1,03
ONT	1000	10	99,98	99,02	-	0,37	0,17	0,55
ONT	1000	15	99,35	98,92	22	0,37	0,17	0,55
ONT	1000	20	94,59	98,74	2.188	0,37	0,17	0,54
ONT	5000	10	100,00	99,87	-	0,37	0,44	0,82
ONT	5000	15	100,00	99,76	-	0,37	0,41	0,78
ONT	5000	20	100,00	99,82	-	0,37	0,38	0,75
ONT	10000	10	100,00	100,00	-	0,37	0,74	1,12
ONT	10000	15	100,00	100,00	-	0,37	0,70	1,08
ONT	10000	20	100,00	99,99	-	0,37	0,66	1,03
10k reads simulated			*homo sapiens*					
Dataset properties			Mapping performances			Runtime		
Technology	Length [bp]	Error rate [%]	Accuracy [%]	Precision [%]	Unmapped [#]	Indexing [s]	Mapping [s]	Total [s]
PacBio	1000	10	99,20	95,62	60	187,30	10,21	197,51
PacBio	1000	15	98,59	95,56	317	187,30	9,93	197,23
PacBio	1000	20	96,17	95,22	2.760	187,30	9,82	197,12
PacBio	5000	10	99,92	95,56	-	187,30	11,36	198,66
PacBio	5000	15	99,71	96,16	24	187,30	10,81	198,11
PacBio	5000	20	99,31	96,58	118	187,30	10,59	197,89
PacBio	10000	10	99,99	96,44	-	187,30	13,01	200,31
PacBio	10000	15	99,83	96,24	1	187,30	12,23	199,53
PacBio	10000	20	99,47	96,76	66	187,30	12,07	199,37
ONT	1000	10	99,18	95,09	41	187,30	10,09	197,39
ONT	1000	15	97,62	95,34	263	187,30	9,80	197,10
ONT	1000	20	92,25	95,22	2.874	187,30	9,77	197,07
ONT	5000	10	99,97	95,33	-	187,30	11,39	198,69
ONT	5000	15	99,63	95,59	9	187,30	10,81	198,11
ONT	5000	20	98,99	96,07	93	187,30	10,64	197,94
ONT	10000	10	100,00	95,97	-	187,30	13,06	200,36
ONT	10000	15	99,87	95,82	1	187,30	12,33	199,63
ONT	10000	20	99,36	95,98	39	187,30	11,87	199,17

Figure 4.2: *MashMap* mapping results

MashMap

10k reads simulated			*escherichia coli*					
Dataset properties			Mapping performances			Runtime		
Technology	Length [bp]	Error rate [%]	Accuracy [%]	Precision [%]	Unmapped [#]	Indexing [s]	Mapping [s]	Total [s]
PacBio	1000	10	99,55	100,00	-	1,18	1,94	3,12
PacBio	1000	15	99,27	100,00	2	1,12	1,70	2,82
PacBio	1000	20	98,20	100,00	347	1,38	0,80	2,18
PacBio	5000	10	100,00	100,00	-	1,12	7,06	8,18
PacBio	5000	15	99,99	100,00	-	1,21	6,50	7,71
PacBio	5000	20	99,93	100,00	14	1,39	3,57	4,96
PacBio	10000	10	100,00	100,00	-	1,10	16,91	18,01
PacBio	10000	15	100,00	100,00	-	1,36	13,45	14,81
PacBio	10000	20	100,00	100,00	-	1,35	7,79	9,14
ONT	1000	10	99,40	100,00	-	1,36	1,08	2,44
ONT	1000	15	99,25	100,00	1	1,30	0,93	2,23
ONT	1000	20	98,15	100,00	478	1,33	0,92	2,25
ONT	5000	10	100,00	100,00	-	1,20	5,01	6,21
ONT	5000	15	99,97	100,00	-	1,32	4,47	5,79
ONT	5000	20	99,96	100,00	35	1,29	4,13	5,42
ONT	10000	10	100,00	100,00	-	1,11	12,16	13,27
ONT	10000	15	100,00	100,00	-	1,29	9,91	11,20
ONT	10000	20	100,00	100,00	1	1,35	7,73	9,08
10k reads simulated			*homo sapiens*					
Dataset properties			Mapping performances			Runtime		
Technology	Length [bp]	Error rate [%]	Accuracy [%]	Precision [%]	Unmapped [#]	Indexing [s]	Mapping [s]	Total [s]
PacBio	1000	10	97,03	100,00	-	929,20	1559,82	2489,02
PacBio	1000	15	95,10	100,00	-	1023,34	878,69	1902,03
PacBio	1000	20	78,89	100,00	-	999,34	654,01	1653,35
PacBio	5000	10	98,61	100,00	-	1134,01	1012,98	2146,99
PacBio	5000	15	98,01	100,00	-	1002,32	687,74	1690,06
PacBio	5000	20	96,04	100,00	8	1174,09	286,35	1460,44
PacBio	10000	10	98,97	100,00	-	1232,01	1414,78	2646,79
PacBio	10000	15	98,57	100,00	-	909,38	987,93	1897,31
PacBio	10000	20	97,47	100,00	-	1000,89	466,66	1467,55
ONT	1000	10	97,00	100,00	-	987,23	1592,49	2579,72
ONT	1000	15	94,38	100,00	-	1258,23	816,36	2074,59
ONT	1000	20	74,25	100,00	-	1078,75	616,01	1694,76
ONT	5000	10	98,38	100,00	-	1174,64	1059,85	2234,49
ONT	5000	15	97,85	100,00	-	988,60	762,46	1751,06
ONT	5000	20	96,28	100,00	26	1008,45	460,72	1469,17
ONT	10000	10	98,87	100,00	-	1307,11	1418,84	2725,95
ONT	10000	15	98,18	100,00	-	1108,39	741,13	1849,52
ONT	10000	20	97,29	100,00	1	999,23	606,49	1605,72

Figure 4.3: *Minimap2* mapping results

Bloom filter cross-correlation

1k reads simulated								*escherichia coli*								
Dataset properties			Tool parameters					Mapping performances								
Technology	Length [bp]	Error rate [%]	k-mer size	window size	filter size	filter hashes	filter capacity	1-Accuracy [%]	2-Accuracy [%]	3-Accuracy [%]	10-Accuracy [%]	1-Precision [%]	2-Precision [%]	3-Precision [%]	10-Precision [%]	Mapping [s]
PacBio	1000	10	8	4	128	16	4	98,70	99,20	99,40	100,00	100,00	50	33,33	10,00	217,54
			8	4	512	16	16	98,60	99,30	99,60	99,90	100,00	50	33,33	10,00	40,53
			8	6	512	16	16	96,30	97,70	98,50	99,90	100,00	50	33,33	10,00	20,92
PacBio	1000	15	8	4	128	16	4	97,50	98,20	99,10	99,80	100,00	50	33,33	10,00	219,34
			8	4	512	16	16	89,90	92,90	94,90	97,00	100,00	50	33,33	10,00	42,00
			8	6	512	16	16	70,40	74,60	76,10	78,10	100,00	50	33,33	10,00	21,22
PacBio	1000	20	8	4	128	16	4	85,60	89,40	90,60	95,20	100,00	50	33,33	10,00	218,21
			8	4	512	16	16	58,20	65,20	67,40	75,30	100,00	50	33,33	10,00	41,01
			8	6	512	16	16	33,50	37,00	38,40	40,60	100,00	50	33,33	10,00	20,63
PacBio	5000	10	8	4	128	16	4	98,90	99,30	99,50	100,00	100,00	50	33,33	10,00	1083,83
			8	4	512	16	16	99,10	99,80	99,80	99,90	100,00	50	33,33	10,00	202,65
			8	6	512	16	16	99,10	99,40	99,80	100,00	100,00	50	33,33	10,00	104,26
PacBio	5000	15	8	4	128	16	4	97,70	98,90	99,10	99,80	100,00	50	33,33	10,00	1180,51
			8	4	512	16	16	97,90	98,90	99,20	99,80	100,00	50	33,33	10,00	261,34
			8	6	512	16	16	97,80	98,80	99,50	99,80	100,00	50	33,33	10,00	110,72
PacBio	5000	20	8	4	128	16	4	93,00	95,40	96,80	99,00	100,00	50	33,33	10,00	1087,55
			8	4	512	16	16	95,00	96,80	97,90	99,10	100,00	50	33,33	10,00	206,46
			8	6	512	16	16	77,50	84,20	86,70	91,80	100,00	50	33,33	10,00	106,08
PacBio	10000	10	8	4	128	16	4	99,00	99,60	99,70	100,00	100,00	50	33,33	10,00	2556,97
			8	4	512	16	16	99,30	99,80	99,90	100,00	100,00	50	33,33	10,00	406,26
			8	6	512	16	16	99,10	99,50	99,60	100,00	100,00	50	33,33	10,00	214,33
PacBio	10000	15	8	4	128	16	4	98,20	99,30	99,80	100,00	100,00	50	33,33	10,00	2182,90
			8	4	512	16	16	98,30	99,00	99,30	99,80	100,00	50	33,33	10,00	409,11
			8	6	512	16	16	97,60	98,10	99,00	99,90	100,00	50	33,33	10,00	245,17
PacBio	10000	20	8	4	128	16	4	89,80	94,10	95,40	99,00	100,00	50	33,33	10,00	2173,63
			8	4	512	16	16	96,30	97,90	98,80	99,70	100,00	50	33,33	10,00	408,69
			8	6	512	16	16	91,40	95,50	96,90	99,20	100,00	50	33,33	10,00	209,41

Figure 4.4: Bloom filter fingerprint compression results

Approximate k-Chaining

10k reads simulated			*escherichia coli*										
Dataset properties			Tool parameters					Mapping performances			Runtime		
Technology	Length [bp]	Error rate [%]	k-mer size	window size	flow	k-chain	erase	Accuracy [%]	Precision [%]	Unmapped [#]	Indexing [s]	Indexing [s]	Total [s]
PacBio	1000	10	12	12	ce	50	1	99,30	46,09	-	0,92	2,98	3,90
			16	12	ce	10	1	99,47	96,56	-	0,95	3,10	4,05
			20	12	ce	1	1	99,79	97,87	79	1,16	3,30	4,46
PacBio	1000	15	12	12	ce	50	1	99,21	48,52	-	0,92	2,72	3,64
			16	12	ce	10	1	99,31	97,94	18	0,95	2,95	3,90
			20	12	ce	1	1	99,77	99,39	765	1,16	3,30	4,46
PacBio	1000	20	12	12	ce	50	1	98,51	49,48	-	0,92	3,49	4,41
			16	12	ce	10	1	98,63	98,69	1.070	0,95	3,55	4,50
			20	12	ce	1	1	99,85	99,83	5.220	1,16	3,67	4,83
PacBio	5000	10	12	12	ce	50	1	99,92	11,65	-	0,92	12,43	13,35
			16	12	ce	10	1	99,97	87,43	-	0,95	16,30	17,25
			20	12	ce	1	1	99,95	88,35	-	1,16	17,10	18,26
PacBio	5000	15	12	12	ce	50	1	99,94	12,91	-	0,92	13,55	14,47
			16	12	ce	10	1	99,98	91,79	-	0,95	16,34	17,29
			20	12	ce	1	1	99,93	93,66	5	1,16	17,15	18,31
PacBio	5000	20	12	12	ce	50	1	99,87	13,84	-	0,92	13,97	14,89
			16	12	ce	10	1	99,84	94,11	-	0,95	15,67	16,62
			20	12	ce	1	1	99,83	97,61	79	1,16	16,13	17,29
PacBio	10000	10	12	12	ce	50	1	100,00	6,60	-	0,92	33,56	34,48
			16	12	ce	10	1	100,00	77,29	-	0,95	32,50	33,45
			20	12	ce	1	1	100,00	78,00	-	1,16	35,41	36,57
PacBio	10000	15	12	12	ce	50	1	100,00	6,60	-	0,92	33,21	34,13
			16	12	ce	10	1	100,00	85,05	-	0,95	33,34	34,29
			20	12	ce	1	1	100,00	86,90	-	1,16	35,88	37,04
PacBio	10000	20	12	12	ce	50	1	100,00	7,15	-	0,92	31,31	32,23
			16	12	ce	10	1	100,00	89,76	-	0,95	32,90	33,85
			20	12	ce	1	1	99,99	93,77	-	1,16	34,91	36,07

10k reads simulated			*escherichia coli*										
Dataset properties			Tool parameters					Mapping performances			Runtime		
Technology	Length [bp]	Error rate [%]	k-mer size	window size	flow	k-chain	erase	Accuracy [%]	Precision [%]	Unmapped [#]	Indexing [s]	Mapping [s]	Total [s]
ONT	1000	10	12	12	ce	50	1	99,36	45,86	-	0,92	2,77	3,69
			16	12	ce	10	1	99,43	86,43	-	0,95	3,32	4,27
			20	12	ce	1	1	99,76	98,14	69	1,16	3,30	4,46
ONT	1000	15	12	12	ce	50	1	99,25	48,23	-	0,92	2,70	3,62
			16	12	ce	10	1	99,28	97,89	17	0,95	2,88	3,83
			20	12	ce	1	1	99,85	99,40	881	1,16	3,78	4,94
ONT	1000	20	12	12	ce	50	1	98,41	49,00	1	0,92	4,02	4,94
			16	12	ce	10	1	98,99	98,61	1.238	0,95	3,70	4,65
			20	12	ce	1	1	99,88	99,79	5.531	1,16	3,03	4,19
ONT	5000	10	12	12	ce	50	1	99,99	11,51	-	0,92	12,20	13,12
			16	12	ce	10	1	99,99	86,79	-	0,95	17,02	17,97
			20	12	ce	1	1	99,97	87,74	-	1,16	16,98	18,14
ONT	5000	15	12	12	ce	50	1	99,96	12,68	-	0,92	13,00	13,92
			16	12	ce	10	1	99,94	91,35	-	0,95	16,30	17,25
			20	12	ce	1	1	99,96	93,32	5	1,16	17,65	18,81
ONT	5000	20	12	12	ce	50	1	99,93	13,69	-	0,92	14,42	15,34
			16	12	ce	10	1	99,91	94,32	-	0,95	15,67	16,62
			20	12	ce	1	1	99,97	97,72	102	1,16	16,44	17,60
ONT	10000	10	12	12	ce	50	1	100,00	5,78	-	0,92	32,08	33,00
			16	12	ce	10	1	100,00	76,03	-	0,95	32,51	33,46
			20	12	ce	1	1	100,00	76,76	-	1,16	35,33	36,49
ONT	10000	15	12	12	ce	50	1	100,00	6,52	-	0,92	33,21	34,13
			16	12	ce	10	1	100,00	83,71	-	0,95	33,34	34,29
			20	12	ce	1	1	100,00	85,98	-	1,16	36,10	37,26
ONT	10000	20	12	12	ce	50	1	100,00	7,06	-	0,92	32,00	32,92
			16	12	ce	10	1	100,00	89,77	-	0,95	30,90	31,85
			20	12	ce	1	1	100,00	93,42	1	1,16	34,95	36,11

Figure 4.5: Approximate k-chaining results

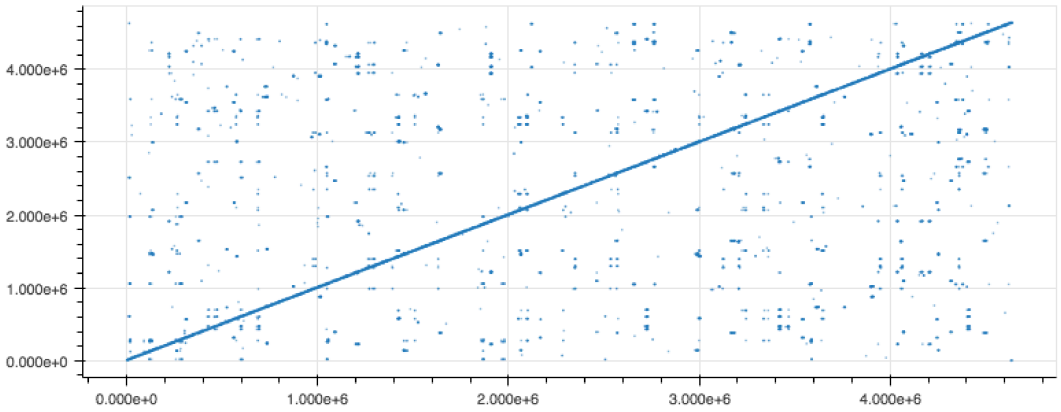

Figure 4.6: Dataset *Oxford Nanopore* R7, *BLASR*

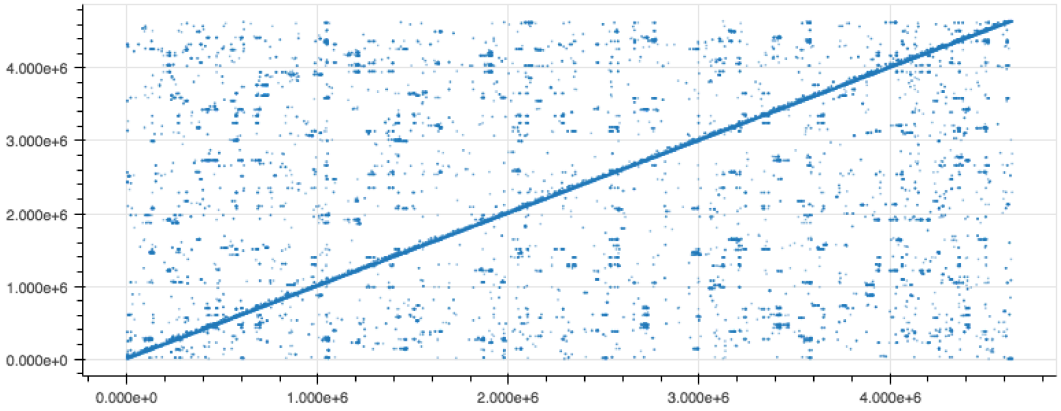

Figure 4.7: Dataset *Oxford Nanopore* R9, *BLASR*

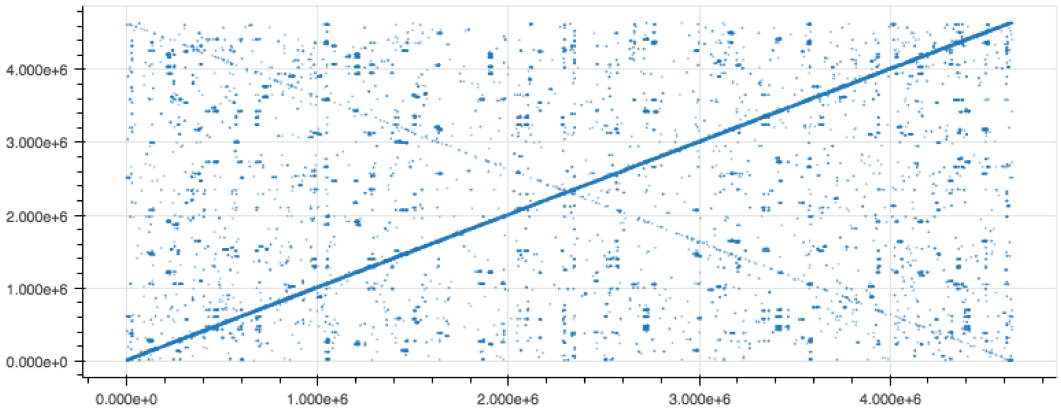

Figure 4.8: Dataset *PacificBioscience* P6C4, *BLASR*

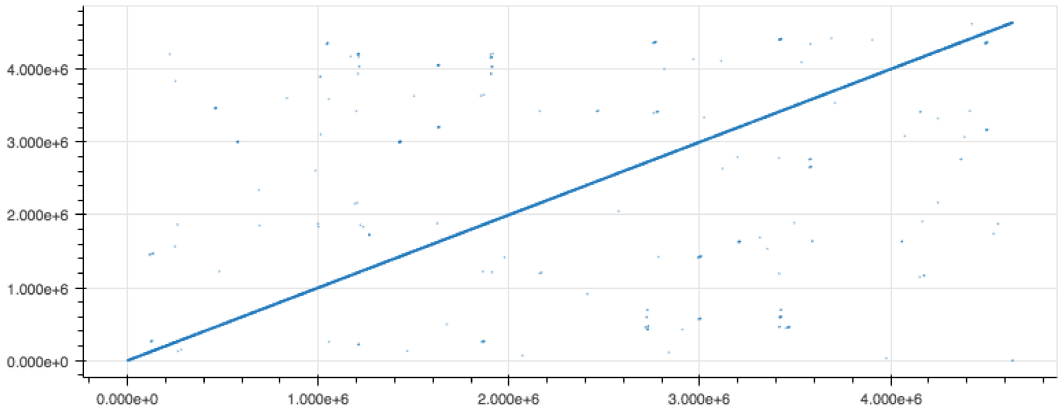

Figure 4.9: Dataset *Oxford Nanopore* R7, *MashMap*

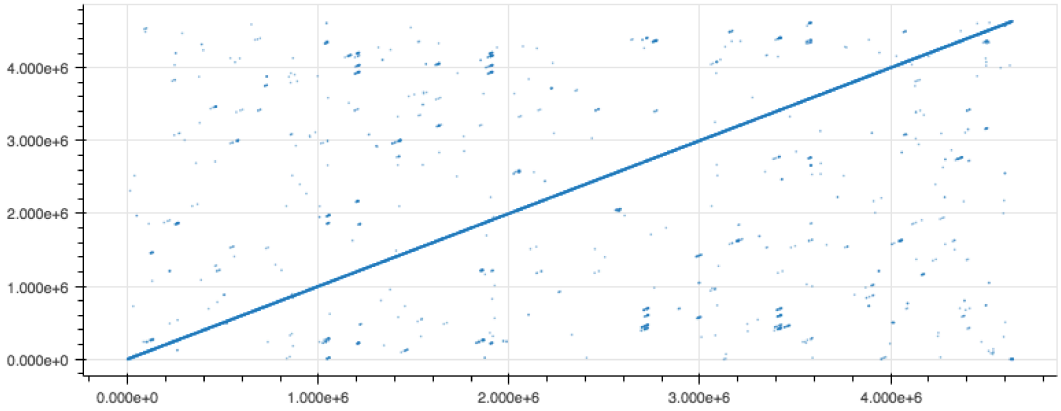

Figure 4.10: Dataset *Oxford Nanopore* R9, *MashMap*

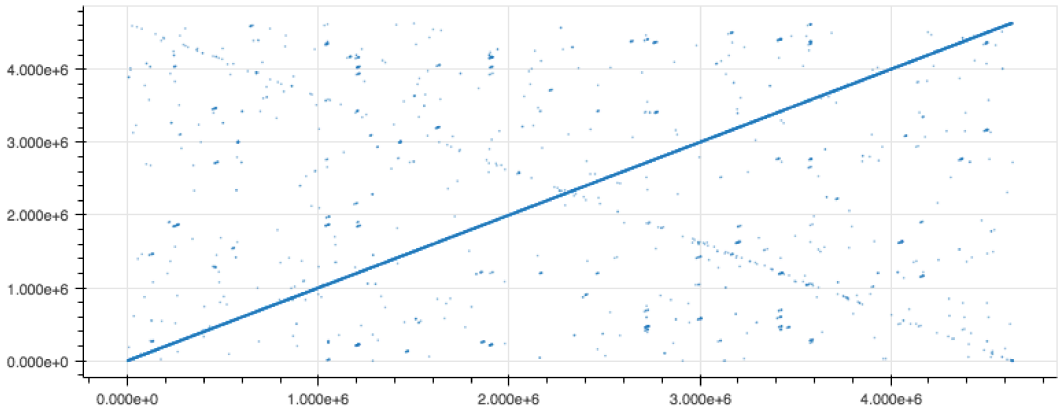

Figure 4.11: Dataset *PacificBioscience* P6C4, *MashMap*

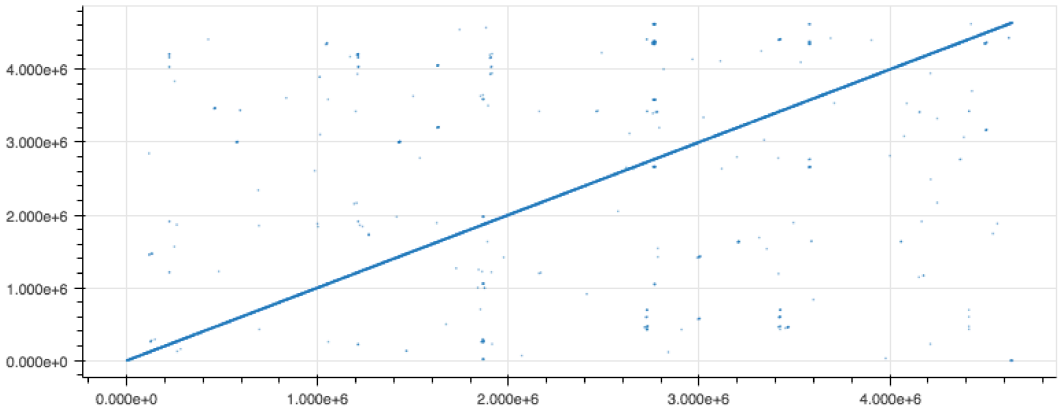

Figure 4.12: Dataset *Oxford Nanopore* R7, *Minimap2*

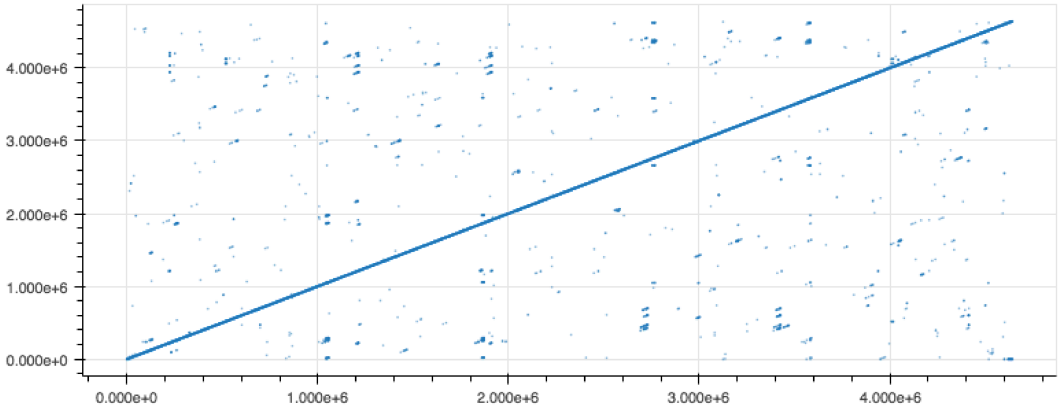

Figure 4.13: Dataset *Oxford Nanopore* R9, *Minimap2*

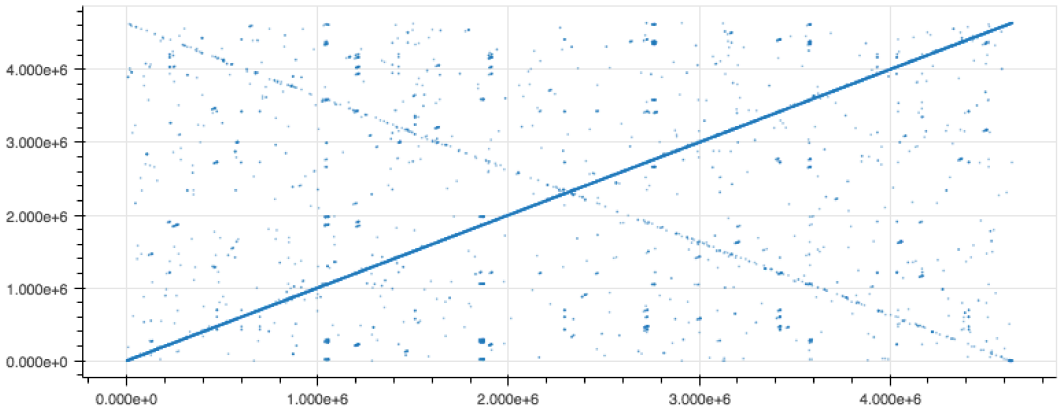

Figure 4.14: Dataset *PacificBioscience* P6C4, *Minimap2*

Chapter 5

Conclusions

This thesis work contributes to the analysis and development of novel approaches and algorithms for the alignment of long-reads sequenced using novel third-generation sequencing technologies, with a more in-depth focus on the problem of sequence mapping. In particular, we had two main objectives:

- defining the main characteristics a third-generation sequencing alignment tool should have, in terms of data representation, data structures exploited for indexing and mapping algorithms
- trying to extend the approaches already present in literature, proposing new algorithms for the management of read fingerprints

About the former point, mapping tools based on sequence hashing and fingerprinting, almost all of them exploiting the *Winnowing* algorithm, proved to be effective in discovering the positions over a reference characterized by a high similarity with a given query read, both in terms of accuracy and in terms of run-time, suggesting that read fingerprinting is an effective way for long-reads mapping. However, such methods should take into account

the fact that real datasets read length distribution is not constant, meaning that a modern tool should be able to manage both more than tens of thousand long reads and shorter than hundreds of base-pairs ones. For the sake of defining read similarity, how the error rate distribute among insertions, deletions and substitutions proved not to be relevant.

Bloom filters based fingerprint compression proved to work in principle, as there exists parameters able to reach near perfect accuracy for every kind of read length and accuracy tested; however, it suffers from two major drawbacks: it depends on four different parameters, two regarding the Winnowing procedure and two for the designing of Bloom filters, making difficult to analytically describe the behavior of the tool. Moreover, Bloom filter sequences proved to be difficult to index, making this tool relies on a brute-force approach for finding the best mapping positions, making it impossible to be used as it is, due to high run-time. However, the simplicity of its approach worth further investigations before completely abandon it.

On the other hand, emulating traditional seed-and-chain algorithms, already used for aligning second-generation short-reads, substituting the concept of seed with the one of exact hash matches, proved to be an effective strategy for long-reads mapping. It proved to be fast and easily controllable through its inner parameters, even if its potential is not still fully exploited in the current tool implementation realized for the sake of this work.

Third-generation sequencing technologies are only at the beginning of their development, and the future approaches that will be adopted for read alignment strongly depends on technologies characteristics. For what presented in this work the major efforts for future developments regards building data structure able to better indexing arrays of multi-dimensional features representing biological sequences, the modeling of algorithms for better

exploits hardware acceleration through vectorization and, maybe the most interesting, the search for machine learning models able to appropriately and efficiently extract feature from raw base-pairs, for improving the task of read fingerprinting over the *Winnowing* algorithm.99,97

Appendix A

First and Second Generation Sequencing

A.1 First Generation Sequencing

After Watson and Creek solved the structure of the DNA in 1953, the scientists involved in molecular biology researches lack of instruments for "reading" sequences of nucleic acids: at that time some techniques were known for inferring the sequence of protein chains, but such methods were not effective in determine DNA sequences because of its length and structure, made of few different units, pretty similar one to the others. Even techniques borrowed by analytic chemistry were ineffective as long as they were only able to determine the concentration of each nucleotide within a solution, but not the order in which they appear in the strand itself[7].

A.1.1 The Sanger method

Even though it is not the first sequencing technique appeared over the years, when referring to the first generation of sequencing protocol, usually the

main focus is on the "chain termination" sequencing method, also known as "Sanger sequencing", published by Sanger in a paper dated December 1977[24]. The main principle in Sanger sequencing is using DNA polymerase, an enzyme involved in DNA replication, in a solution containing the strand to be sequenced chained with a synthetic primer for making the polymerase bind to the template, regular deoxy-nucleotide-triphosphate (dNTP) and a small amount of di-deoxy-nucleotide-triphosphate (ddNTP). ddNTPs are molecules with the same structure of regular dNTP, but missing the 3' hydroxyl group, making the polymerase reaction stops whenever such a molecule is incorporated in the strand by the enzyme. Such molecule is also marked by a fluorescent die, so that a luminescence is released whenever it is hit by a laser: a different colour is used for each of the four kinds of ddNTPs available, for making possible to distinguish which of them is incorporated in the growing strand. The rationale behind the Sanger sequencing protocol is that if it is possible to determine the base present at a random position in the template by making the polymerase enzyme stops the reaction whenever a ddNTP is included, then, triggering multiple reactions in parallel over a sufficiently high number of fragments makes possible to determine the whole nucleotides in the template sequence. The actual base call process is performed through electrophoresis: the multi-sized sequenced fragments are detached from the template and flow through a capillary gel after an electrical difference of potential is applied. A laser beam hit the ddNTPs attached as last nucleotide in the fragment at the end of the path and a light detector register what of the four fluorescence is released. Shorter fragments flow faster through the capillary, making the machine call their last base first[7]. The steps involved in Sanger sequencing can be summarized as follows:

- Library preparation: a well-known primer sequence is attached to each

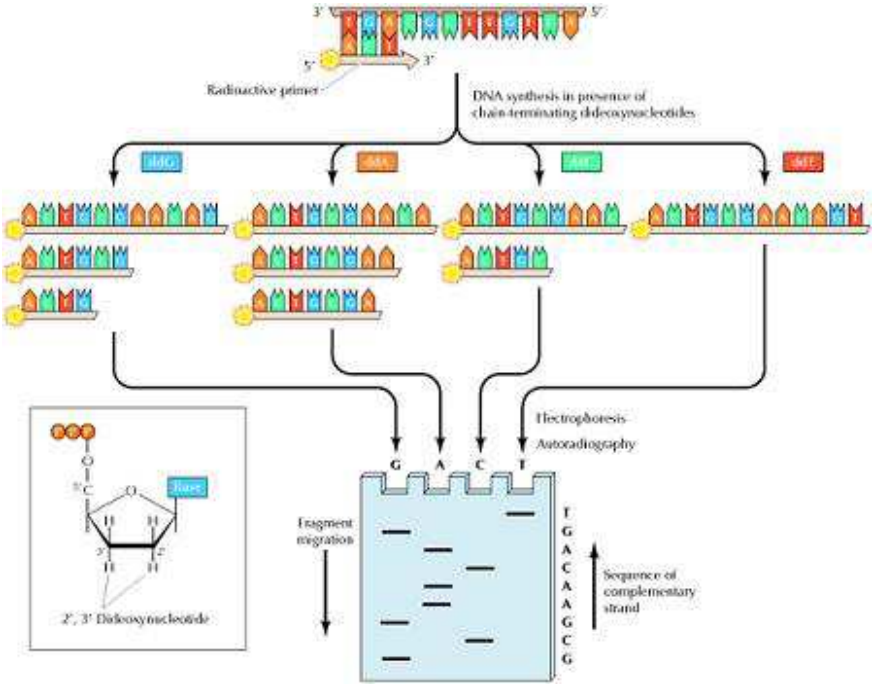

Figure A.1: Sanger sequencing pipeline

template to be sequenced for allowing the DNA polymerase to attach to the template.

- Reaction mixture preparation: four different samples are prepared, each containing the DNA template to be sequenced, DNA polymerase, normal dNTPs and a minor concentration of a kind of ddNTP for each sample: ddATP, ddCTP, ddGTP and ddTTP.

- Primer elongation: DNA polymerase incorporates dNTPs until the process is stopped by the inclusion of a ddNTP.

- Capillary electrophoresis: fragments are discriminated by their length and the measures of the fluorescence emitted by the ddNTP on their tail when hit by a laser beam, is performed.

- Fluorescence analysis: a sequence of light signals are translated into

sequence of bases reported in a formatted file.

A.1.2 The Human Genome Project

Sanger method was very important from an historical point of view: it was the main sequencing protocol used for carrying on the so called Human Genome Project, a US Congress founded plan aiming at sequencing the whole human genome for the first time which took place between 1990 and 2003. The project was an important milestone in the history of genetics: even if it had its ideologically origins in 1980, virtually it represented the continuation of the experiments designed by Morgan and his collaborators at the Columbia University, which, during the first decade of the twentieth century, demonstrated that genes are located on chromosomes, posing the basis for modern genetics. HGP researchers deciphered the Human genome not only by sequencing long nucleic acid molecules with the Sanger method, but also exploiting two more techniques:

- Mapping technologies, showing gene locations for major sections of all out chromosome.
- Linkage-maps, for tracking inherited traits over generations of individuals.

The first draft of the human genome was published in Nature in 2001, with almost 90% of the whole genome sequenced and was already significant in proving the number of genes in the our chromosomes, whose number was estimated to be more than fifty thousand, being actually a bit more than twenty thousand genes, much lower that the expected. The project was declared accomplished in 2003 after 13 years of efforts and an overall budget of 2.7 billion dollars, showing that the whole genome sequencing of any living

being, even as complex and long as the Human one, was theoretically possible, with a number of limitations regarding the cost and the process speed.

A.2 Second generation sequencing

Second generation sequencing machines were powered by a technological paradigm shift: the amount of DNA the machines were able to sequence at once was greatly increased exploiting massively parallel protocols during sequencing reaction. All the protocols considered here share the first two steps of the sequencing process:

- Library preparation: the double-stranded DNA is fragmented, denatured and ligated to proper adapter molecules for making the sequencing protocol being triggered.

- Clonal amplification: the previously denatured fragments are replicated using a certain number of PCR cycles, depending on the purpose why the sequencing is being carried out.

The actual sequencing process is often referred to as *wash-and-scan* because consists in sequentially flooding reagents in, incorporating nucleotides into the growing DNA strands, stopping the incorporation reaction, washing out the excess reagent, scanning to identify the incorporated bases and repeat until possible[25]. Some of the most interesting SGS protocols discussed here includes, the pyrosequencing reaction implemented in the Roche 454 machine, the first mature technology exploited, the Illumina protocol, which has the largest market share and less common technologies such as IonTorrent and SOLiD, interesting for the different approaches they use for the sequencing purpose.

A.2.1 Pyrosequencing and the Roche 454 machine

During the same years when the "chain termination" sequencing technologies were improved more and more, a new technique for sequencing DNA appeared, which was able to determine whether at least one, and if so how many, nucleotides are included by the polymerase enzyme in the growing DNA strand by measuring the light intensity emanated by a two-enzymes reaction called pyrophosphate synthesis. This process is called pyrosequencing and needs a solution of DNA polymerase, adenosine phosphosulfate (APS) and two enzymes: ATP sulfurylase and luciferase, along with the DNA template to be sequenced; it consists of the following steps:

- Whenever a nucleotide is incorporated in the growing DNA strand by the polymerase enzyme, a pyrophosphate (PPi) molecule made of two phosphate groups is released.
- ATP sulfurylase acts as a catalyst for the reaction forming ATP from PPi and APS.
- Luciferase catalyzes the conversion of luciferin to oxyluciferin with the participation of the ATP molecule previously synthetized, ending in the liberation of light, whose intensity depends on how many molecules of PPi were produced: the more nucleotides are incorporated, more PPi is produced, greater is the intensity of light being produced.

Repeating such a procedure four times per cycle, releasing in the solution one dNTP at a time and cleaning the solution by the unneeded molecules, actually allows determining the sequence of bases composing a DNA template. Such procedure has a number of advantages with respect to Sanger sequencing: it exploits natural nucleotides, with respect to heavily modified ddNTPs and can be observed in real time, instead of requiring lengthy

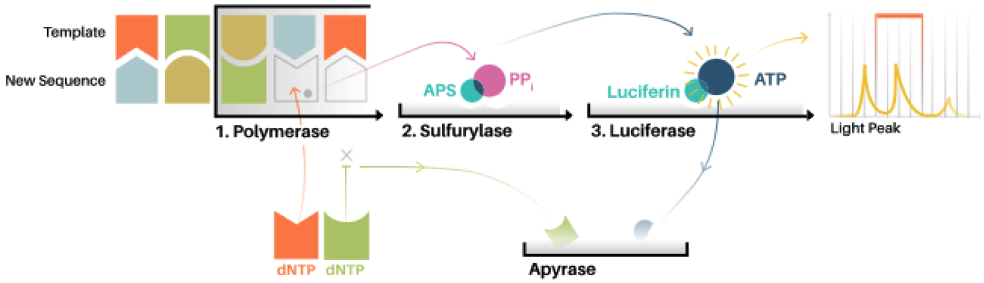

Figure A.2: Pyrosequencing 2-enzymes reaction

electrophoresis processes. The main drawback stands in the length of the sequences it is capable of producing being at most 400-500 base pairs long, shorter than Sanger protocol[7]. Pyrosequencing is at the base of the very first second generation sequencing machine, called GS20, produced by Roche. It allowed a huge parallelism in the sequencing reactions, by attaching the single stranded DNA fragments resulting from library preparation to streptavidin beads via adapter sequences and by undergo an emulsion PCR process for filling the whole bead with clones of the first fragment attached. The beads are then washed over a reaction plate made of wells, fitting one bead each. As a last step, Pyrosequencing occurs and a CCD sensor is used for registering the amount of light released in each well. The output of such a process is a diagram showing the intensity of the light peaks detected by the sensor over time, in correspondence of annotations reporting what dNTP was released in the reaction at any time. Such graphic, reporting the evolution of pyrophosphate synthesis, is called pyrogram and allows retrieving the correct sequence of bases in the template strand. The success of this machine paved the way for a number of similar technologies exploiting a massive clonal amplification, followed by successive wash-and-scan cycles the Roche machine proved to work.

A.2.2 The Illumina machines

The most well-settled sequencing technology on the market is the one used by Illumina machines, which, taking inspiration from Pyrosequencing, completely redesigned the chemistry involved in the library preparation step and the actual sequencing procedure. The clonal amplification is no more realized by streptavidin beads but by attaching fragments in a flowcell at random positions by one end, performing some cycles of PCR called "bridge amplification"; this name comes from the way fragments fold over for binding to the complementary flow cell adapter sequence before the PCR reaction can occur, resembling the shape of a bridge. Once cluster of identical strands are created the actual sequencing reaction can take place, consisting in a number of cycles where the cell is filled with all four colour-labelled nucleotides at the same time, competing for being incorporated in the single strand chain built step-by-step by DNA polymerase. Once the correct nucleotide is chained light is released and the sequencing process can proceed synchronously, as long as the polymerization is stopped until the coloured die is cleaved away. Here the reduced read length depends on the signal-to-noise ratio diminishing as the reaction proceed due to defects in the wash-and-scan technique, being not able to cleave all the procedure junk products away. The typical read length for such a technology was of only few tenths of base pairs initially, increasing to more than one hundred nucleotides as the process was optimized[7].

A.2.3 Other technologies

The two most common sequencing machines described so far both relying DNA polymerase, for synthetizing a DNA strand complementary to the tem-

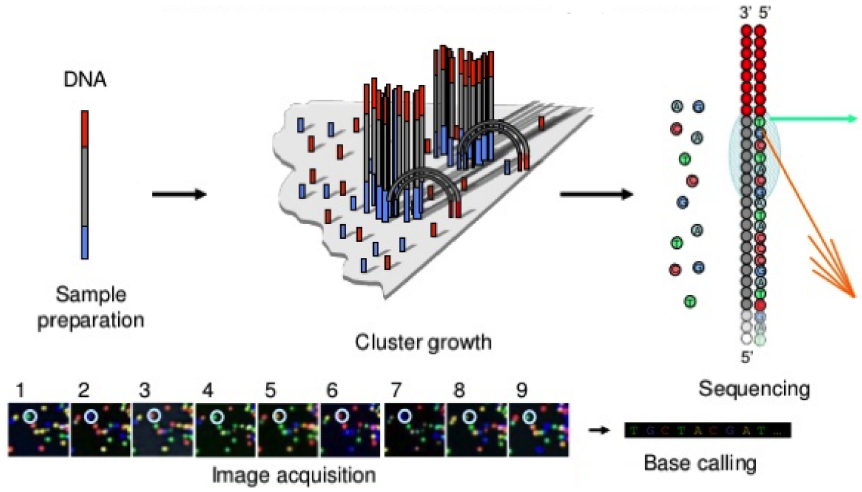

Figure A.3: Illumina sequencing pipeline

plate, and some kind of light capturing device for detecting the nucleotide being sequenced. However other companies appeared during the years of development of second generation sequencing, proposing protocols relying on different principles.

SOLiD sequencing machine relies on a clonal amplification chemistry similar to the one proposed by Roche but exploits the action of DNA ligase instead of DNA polymerase for binding a proprietary chemical probe with an attached coloured die to the template sequence for detecting what kind of base is being sequenced, making this technology being classified as *sequencing-by-ligation*, instead of *sequencing-by-synthesis*.

Another interesting technology was developed by IonTorrent which again uses the same amplification chemistry as Roche, along with a very similar base incorporation mechanism, but the base-call is performed by measuring pH difference caused by the hydrogen ions released during polymerization exploiting the CMOS technology, actually making IonTorrent the first machine taking advantage of a post-light sequencing technology[7].

A.2.4 The genomic revolution

Figure A.4 reports how the cost of sequencing decreased over the years: analyzing the last years during which the Human Genome Project was active already shows how the sequencing technology was becoming cheaper and cheaper at a rate comparable to the one predicted by the Moore's law, concerning the costs of computing. In 2006 the landscape of sequencing technologies was changing due to the introduction of Pyrosequencing and other second generation sequencing technologies, but those were not completely implemented at that time, so that researchers involved in genomics continued improving the same methods at the basis of the Human Genome Project, lowering the cost for genome sequencing at a Moore-like rate. After 2008 the framework for whole genome sequencing completely changed: as "Next Generation Sequencing" technologies approached the market, they lead to a more-than-exponential decrease in sequencing cost resulting in the ability of sequencing the whole human genome for less than two thousand dollars in 2016. This huge cost reduction, along with the increasingly deep understanding of DNA role in most genetic diseases, lead to incredible efforts for sequencing billion of bases from organisms of the same species, studying how different kind of mutations in different genes are related to people health; concerning the human genome, huge projects stem from the ambitions of turning a deep but generic DNA knowledge into some kind of personalized medicine[2]. In 2012 the UK government launched the "100000 genome project", an attempt of whole genome sequencing of more than 70000 National Health Service patients and their relatives affected by rare diseases, looking for the possibility of developing a more in-depth knowledge of these kind of patologies and ensuring a more effective medical treatments. In the same years a large-scale genomic project in Iceland succeeded in sequencing

more than two thousand Icelanders, correlating millions of Single Nucleotide Polymorphism and insertion/deletion phenomena discovered to the different phenotypes they provoked[6][28], proving the capabilities of state-of-the-art sequencing technologies and data analysis pipelines.

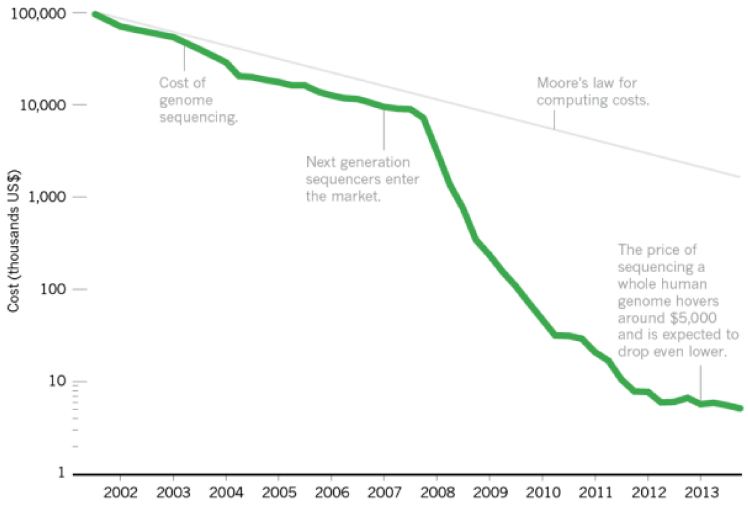

Figure A.4: Cost per Human genome sequencing

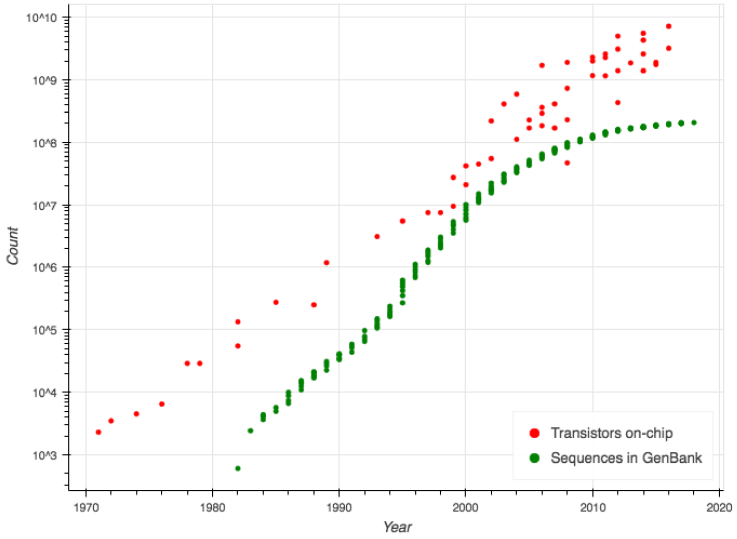

Figure A.5: Transistors on-chip vs sequences in GenBank

Bibliography

[1] M.J. Chaisson and G. Tesler. "Mapping single molecule sequencing reads using basic local alignment with successive refinement (BLASR): application and theory". In: *BMC Bioinformatics* (2012).

[2] F.S. Collins et al. "A vision for the future of genomics research". In: *Nature* 442 (2003).

[3] Francis Creek. "Central dogma of molecular biology". In: *Nature* 227 (1970), pp. 561–563. DOI: 10.1038/227561a0.

[4] Francis Creek. "What mad pursuit". In: 1988.

[5] J. Eid et al. "Real-Time DNA Sequencing from Single Polymerase Molecules". In: *Science* 323 (2009). DOI: 10.1126/science.1162986.

[6] D.F. Gudbjartsson et al. "Large-scale whole-genome sequencing of the Icelandic population". In: *Nature Genetics* (2015). DOI: 10.1038/ng.3247.

[7] J.M. Heater and B. Chain. "The sequence of sequencer: The history of sequencing DNA". In: *Genomics* 107 (2016). DOI: 10.1016/j.ygeno.2015.11.003.

[8] Li Heng. "Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM". In: *arXiv* (2013). DOI: arXiv:1303.3997.

[9] Li Heng. "Minimap and Miniasm: fast mapping and de novo assembly for noisy long sequences". In: *arXiv* (2017). DOI: arXiv:1512.01801v2.

[10] Li Heng. "Minimap2: versatile pairwise alignment for nucleotide sequences". In: *arXiv* (2018). DOI: arXiv:1708.01492v4.

[11] C. Jain et al. "A Fast Approximate Algorithm for Mapping Long Reads to Large Reference Databases". In: *bioRxiv* (2017). DOI: 10.1101/103812.

[12] Watson James D. "DNA: the story of genetic revolution". In: Knopf, 2017. Chap. 1.

[13] X. Jiao et al. "A Benchmark Study on Error Assessment and Quality Control of CCS Reads Derived from the PacBio RS". In: *NIH-PA* 4 (2013). DOI: 10.4172/2153-0602.1000136.

[14] Hayan Lee et al. "Third-generation sequencing and the future of genomics". In: *bioRxiv* (2016). DOI: 10.1101/048603.

[15] H. Lodish et al. "Molecular cell biology". In: Freeman, 2008, pp. 8–13.

[16] H. Lu, F. Giordano, and Z. Ning. "Oxford Nanopore MinION Sequencing and Genome Assembly". In: *Genomics Proteomics Bioinformatics* 14 (2016). DOI: 10.1016/j.gpb.2016.05.004.

[17] R.C. McCoy et al. "Illumina TruSeq Synthetic Long-Reads Empower De Novo Assembly and Resolve Complex, Highly-Repetitive Transposable Elements". In: *PLoS One* 9 (2014). DOI: 10.1371/journal.pone.0106689.

[18] S. McGinn and I.G. Gut. "DNA sequencing - spanning the generations". In: *New Biotechnology* 00 (2012). DOI: 10.1016/j.nbt.2012.11.012.

[19] H. Mohamadi et al. "ntHash: recursive nucleotide hashing". In: *Bioinformatics* (2016). DOI: 10.1093/bioinformatics/btw397.

[20] *Oxford Nanopore Technology R9 1D E.coli library*. http://lab.loman.net/2016/07/30/nanopore-r9-data-release/.

[21] *PacBio SMRT P6C4 E.coli library*. https://github.com/PacificBiosciences/DevNet/wiki/E.-coli-Bacterial-Assembly.

[22] A. Rhoads and K.F. Au. "PacBio Sequencing and Its Applications". In: *Genomics Proteomics Bioinformatics* 13 (2015). DOI: 10.1016/j.gpb.2015.08.002.

[23] V. Roussev. "Advances in digital forensic IV". In: 2010. Chap. 8.

[24] F. Sanger, S. Nicklen, and A.R. Coulson. "DNA sequencing with chain-terminating inhibitors". In: *Proc. Natl. Acad. Sci. USA* 74 (1977).

[25] E.E. Schadt, S. Turner, and A. Kasarkis. "A window into third-generation sequencing". In: *Human Molecular Genetics* 19 (2010). DOI: 10.1093/hmg/ddq416.

[26] S. Schleimer, D.S. Wilkerson, and A. Aiken. "Winnowing: Local Algorithms for Document Fingerprinting". In: *ACM* (2003).

[27] B.K. Stöcker, J. Köster, and S. Rahmann. "SimLoRD: Simulation of Long Read Data." In: *Bioinformatics* (2016). DOI: 10.1093/bioinformatics/btw286.

[28] *The 100,000 Genomes Project*. https://www.genomicsengland.co.uk/the-100000-genomes-project/.

[29] *The SMRTBell template*. https://www.pacb.com/smrt-science/smrt-sequencing/single-molecule-resolution/.