IonGAP – an Integrated Genome Assembly Platform for Ion Torrent Data

Author: Adrián Báez Ortega

Supervisors: Marcos Colebrook Santamaría, José Luis Roda García

Date: 17/07/2014

D. Marcos Colebrook Santamaría, Profesor de la Escuela Técnica Superior de Ingeniería Informática, adscrito al Departamento de Matemáticas, Estadística e Investigación Operativa de la Universidad de La Laguna, y

D. José Luis Roda García, Profesor de la Escuela Técnica Superior de Ingeniería Informática, adscrito al Departamento de Ingeniería Informática y Sistemas de la Universidad de La Laguna.

CERTIFICAN: Que la presente memoria titulada ***IonGAP – an Integrated Genome Assembly Platform for Ion Torrent Data***, ha sido realizada bajo nuestra dirección por el estudiante **Adrián Báez Ortega**, y constituye su Proyecto Fin de Carrera de Ingeniería Informática por la Universidad de La Laguna.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos que haya lugar, firmamos el presente certificado en La Laguna, a **17 de julio de 2014**.

D. Marcos Colebrook Santamaría

D. José Luis Roda García

Resumen

Ante la falta, por parte del Instituto Universitario de Enfermedades Tropicales y Salud Pública de Canarias, de una plataforma adecuadamente configurada para el ensamblado de genomas a partir de datos obtenidos mediante secuenciación masiva de ADN usando la tecnología específica de Ion Torrent, se presenta la solución IonGAP, una plataforma web pública capaz de llevar a cabo tanto este proceso de ensamblado como posteriores etapas de análisis preliminar de los datos, de forma automatizada y cómoda para el usuario. Existen muchos ejemplos de herramientas públicas accesibles vía web y diseñadas para ayudar a los investigadores en algunas fases del análisis de datos genómicos; no obstante, la originalidad de IonGAP reside en la capacidad de proporcionar un uso simple y automatizado de toda una colección de aplicaciones de ensamblado y análisis genómico, configuradas específicamente para trabajar con datos de genomas microbianos procedentes de secuenciación por Ion Torrent.

La determinación de las herramientas y configuraciones más adecuadas para cada una de las etapas del proceso ha conllevado una serie de estudios comparativos que servirán como referencia para medir la adaptación de los diferentes programas evaluados a la tecnología de secuenciación por Ion Torrent. Por otra parte, la apertura al público de la plataforma como un servicio web posibilita su uso por parte de investigadores de todo el mundo, con el único requisito de disponer de un simple navegador web; además de servir de guía para futuros desarrollos de soluciones especializadas que utilicen datos de secuenciación obtenidos mediante esta tecnología.

Abstract

Because of the lack, faced by the *Instituto Universitario de Enfermedades Tropicales y Salud Pública de Canarias* (University Institute of Tropical Diseases and Public Health of the Canary Islands), of a platform properly configured for genome assembly of data obtained by massive DNA sequencing using the Ion Torrent specific technology, the solution IonGAP is presented. IonGAP is a public Web platform capable of performing both the assembly process and subsequent preliminary data analyses, in an automated, user-friendly way. There are many examples of public Web tools designed to help researchers in particular stages of genomic data analysis; however, the originality of IonGAP lies in the ability to provide a simple, automated use of a whole collection of genome assembly and analysis applications, specifically configured to handle microbial genomic data generated by using the Ion Torrent chemistry.

The determination of the most suitable tools and configurations for each stage of the process has entailed a series of comparative studies that will serve as a reference to measure the adaptation of the different evaluated programs to the Ion Torrent sequencing technology. On the other hand, the availability of this platform to the public as a Web service makes it accessible to researchers from all over the world, with the only requirement of having an ordinary Web browser. Besides, it can be used as a guide for future development of specialized solutions that employ genome sequence data obtained by means of this technology.

Agradecimientos

Este trabajo no habría sido posible sin la ayuda de muchas personas que me han brindado su colaboración, sus conocimientos y su apoyo. Quedo especialmente agradecido con mis directores y colaboradores: el Dr. Marcos Colebrook y el Dr. José Luis Roda, de la Escuela Técnica Superior de Ingeniería Informática (ETSII) de la Universidad de La Laguna; y el Dr. Luis Fabián Lorenzo, el Dr. Carlos Flores y el Dr. Mariano Hernández, del Grupo de Genómica Aplicada del Instituto Universitario de Enfermedades Tropicales y Salud Pública de Canarias (IUETSPC), no sólo por guiar y hacer posible el desarrollo de todo el proyecto, sino también por su inagotable y contagiosa ilusión por el mundo de la bioinformática y sus perspectivas, así como por su exhaustiva revisión de este manuscrito. Sus ansias de descubrimiento y su voluntad de trabajo son la base sobre la que está construido este proyecto, y aún más, son la motivación que me ha conducido a ampliar mi formación en bioinformática con el fin de dedicarme profesionalmente a un campo tan asombrosamente fascinante como éste.

Los gastos implicados en la secuenciación de los datos empleados en el estudio han corrido a cargo de diferentes proyectos de investigación del IUETSPC, a cuyos miembros, especialmente al Dr. Basilio Valladares, trasmito también mi más sincero agradecimiento. De igual modo doy las gracias al Dr. Manuel Espinosa, del Centro de Investigaciones Biológicas del Consejo Superior de Investigaciones Científicas (CIB-CSIC), y a la Dra. Teresa Coque, del Hospital Universitario Ramón y Cajal, por poner a nuestra disposición la cepa de *Streptococcus agalactiae* que ha sido fuente principal de los datos empleados en este proyecto. En deuda quedo también con Pedro González y el personal del Centro de Cálculo de la ETSII, por haber proporcionado los valiosos recursos que han sido imprescindibles para el desarrollo y funcionamiento del servicio web, con una diligencia y profesionalidad extraordinarias.

Acknowledgements

This work would not have been possible without the help of many people who have provided me their collaboration, knowledge and support. I am especially grateful to my directors and collaborators: Dr. Marcos Colebrook and Dr. José Luis Roda, at the *Escuela Técnica Superior de Ingeniería Informática* (ETSII), *Universidad de La Laguna*; and Dr. Luis Fabián Lorenzo, Dr. Carlos Flores and Dr. Mariano Hernández, at the Applied Genomics Group of the *Instituto Universitario de Enfermedades Tropicales y Salud Pública de Canarias* (IUETSPC), not only for guiding and making the whole project development possible, but also for their endless and infectious passion for the world of bioinformatics and its prospects, as well as for the exhaustive revision of this manuscript. Their thirst for discovery and their willingness to work are the basis on which this project is built, and yet more, the motivation that has led me to broaden my professional training in order to keep working in such an astoundingly fascinating field.

The expenses involved by the sequencing of the data employed in the study have been assumed by different research projects of the IUETSPC, to whose members, especially to Dr. Basilio Valladares, I express my most sincere gratitude. In the same way, I thank Dr. Manuel Espinosa, at the *Centro de Investigaciones Biológicas* of the *Consejo Superior de Investigaciones Científicas* (CIB-CSIC), and Dr. Teresa Coque, at the *Hospital Universitario Ramón y Cajal*, for providing the *Streptococcus agalactiae* strain that has been the main source of the data employed in this project. I also owe Pedro González and all the ETSII's Computing Center personnel for supplying the indispensable resources for the development and functioning of the Web service, with extraordinary diligence and professionalism.

It is in our genes to understand the universe if we can, to keep trying even if we cannot, and to be enchanted by the act of learning all the way.

Lewis Thomas

Table of contents

List of Figures and Tables

Chapter 1. Introduction

Abstract

- The genome is composed of DNA, a long sequence of nucleotides that are represented by their bases (A, C, G, T) and code all the genetic information of an organism.
- Genomics studies the genome of living beings in order to discover their structure and function.
- In order to obtain a manageable representation of the sequence of bases of a genome, two processes need to be performed: genome sequencing and genome assembly.
- The IonGAP project comes up from the need of the *Instituto Universitario de Enfermedades Tropicales y Salud Pública de Canarias* for an integrated, easy-to-use platform that automates the stages of assembly and preliminary analysis of genomes under study.

1.1 Genomes, genes and DNA

The hereditary nature of every living organism comes defined by its **genome**. It consists of a long sequence of nucleic acid which provides the necessary information to constitute and maintain an organism in functioning. By means of a complex series of interactions, this sequence is used to produce all the proteins of the organism in the right moment and place. Proteins are part of the body structure or have the ability to build structures or perform the metabolic reactions that are necessary for life.

Functionally, the genome is composed by **genes**. Genomes of living organisms can contain from less than 500 genes, like in some bacteria, up to more than 25000, in human beings. On the other hand, from a physical point of view, the genome is composed by molecules of **deoxyribonucleic acid**, or **DNA**. DNA is a polynucleotide, a polymer which consists of the union of between hundreds and billions of **nucleotides**. Each nucleotide is made up of a sugar (deoxyribose), a **nitrogenous base**, which can be adenine (A), thymine (T), cytosine (C), or guanine (G), and a phosphate group, which acts as a link between each nucleotide and the next one. What distinguishes one nucleotide from other is its nitrogenous base, and because of this, a DNA sequence is specified by naming only the sequence of its bases.

In living beings, DNA turns up as a pair of strands which are wound round each other, forming a spiral-staircase-shaped structure called double helix. The double helix maintains itself stable by the formation of hydrogen bonds between the bases in one strand and those in the other one. Each **base pair** (**bp**) constitutes a "step" of this staircase, and each type of base in one strand is paired only with one type of base in the other one, so A pairs only with T, and C only with G.

The sequence of nucleotides of most genes is translated into a protein that an organism is able to synthesize or *express* in one or several moments of its life, using the information in this sequence. The connection between the sequence of nucleotides and the sequence of amino acids of the protein is determined by the genetic code, which is used during the process of **translation** or protein synthesis. The coding unit of the genetic code is a group of three

nucleotides (**triplet** or **codon**), represented by the three initials of the nitrogenous bases (for instance, ACT, CAG, TTT) [1].

1.2 Studying the genome: Genomics

Genomics is the discipline in genetics devoted to the comprehensive study of the functioning, structure, content, evolution and origin of genomes. To this end, it employs knowledge from other sciences, such as molecular biology, biochemistry, computer science, mathematics or physics. One of the aims of genomics is to predict the function of genes from their sequence or their interaction with other genes. For this reason, genomics is of great importance in the research of diseases which are related to genome defects, such as cancer, congenital immunodeficiency, cystic fibrosis or dementia; as well as in the study of the variability in the response of different individuals to certain drugs. Currently, it is one of the areas of biology in greatest development. However, in order to be able to study the genome of an organism, first it is necessary to read the sequence of nucleotides that makes up the genes in it. This is possible thanks to a set of methods and techniques that allow DNA sequencing.

1.3 Genome sequencing

Genome sequencing is the set of biochemical methods and techniques whose purpose is the determination of the order of the nucleotides in a DNA sequence. Although sequencing was initially performed by chemical and radioactive methods, nowadays there are enzymatic sequencing technologies which offer lower costs and higher performance, gathered in the so-called **next-generation sequencing** (NGS). These technologies permit to massively parallelize several sequencing reactions, producing thousands or millions at the same time. The first step to sequence a genome by NGS is generating several copies of the target sequence, and then splitting them into short-length fragments (currently between 30 and 800 bp). The sequence of each one of these fragments is subsequently interpreted thanks to a device called **genome sequencer** or genome sequencing platform, which may employ any of the existing sequencing technologies. This process is known as **whole-genome sequencing** or **shotgun sequencing**.

Each sequence of nucleotides is read, by convention, from one of its ends, called 5' (five prime) end, and towards the other one, named 3' end. However, the sequencer is not capable of controlling the direction in which the DNA fragments are sequenced. Moreover, a fragment can be sequenced either in one way (from one end to the other), generating a **single-end read** (often called unpaired read), or from both ends in opposite directions, producing two **paired-end reads**. Each interpreted base of a read has an associated **quality score value**, which refers to the reliability of the sequencing result for this base. Therefore, the result of sequencing is an enormous amount of short-length reads, which come from the sequencing of each DNA fragment and, in turn, belong to several replicas of an original, much longer DNA sequence. In order to obtain the original whole sequence, it is required a way to bind back all these reads together to create longer ones, as though a colossal puzzle had to be resolved. Here, bioinformatics play an essential role, making genome assembly possible.

1.4 Genome assembly

Genome assembly refers to the alignment and merging of multiple fragments of a much longer DNA sequence (a genome or part of it) with the purpose of reconstructing the original sequence. In broad outline, it can be defined as a process of all-against-all comparison between the reads in order to find regions shared by two or more reads, due to the fact that they belong to different copies of the original sequence. The reads that present regions in common (in other words, that *overlap*) are progressively merged together to form longer contiguous sequences called **contigs** (from "contiguous sequence"). This kind of genome assembly is also known as ***de novo* assembly** because the contigs are generated using only the input reads, unlike **mapping assembly**, in which reads are only aligned (mapped) to the genomic sequence of a reference organism, often called reference sequence. Subsequently, it is possible to perform a process known as **scaffolding**, in which **mate-pair** reads (pair-end reads with a long gap between them) are used to determine the relative position of contigs and to estimate the length of the gaps between them. In consequence, contigs are gathered into non-contiguous DNA segments called **scaffolds**. Figure 1.1 outlines the whole process of genome sequencing and assembly.

The genome assembly problem is not so simple as a common string-matching problem, due to the presence of sequencing errors, redundant sequences that can be almost impossible to solve, and variations in sequencing coverage (the number of reads that cover a same base position in the genome), which is usually higher in some regions than in others. Assembly is also complicated due to the complexity of processing huge data volumes (normally, gigabytes or terabytes). In practice, the different existing assembly algorithms and their implementation are considerably complex, and they may require high-performance computing systems in order to deal with large size genomes [2].

In the near future, bacterial genome sequencing and assembly at low cost is expected to be applied with major repercussions in problems such as clinical diagnosis of infectious diseases, the deciphering of evolutionary history of certain microorganisms, and even the analysis of forensic samples in the context of criminal investigations or in the defense against biological warfare [3].

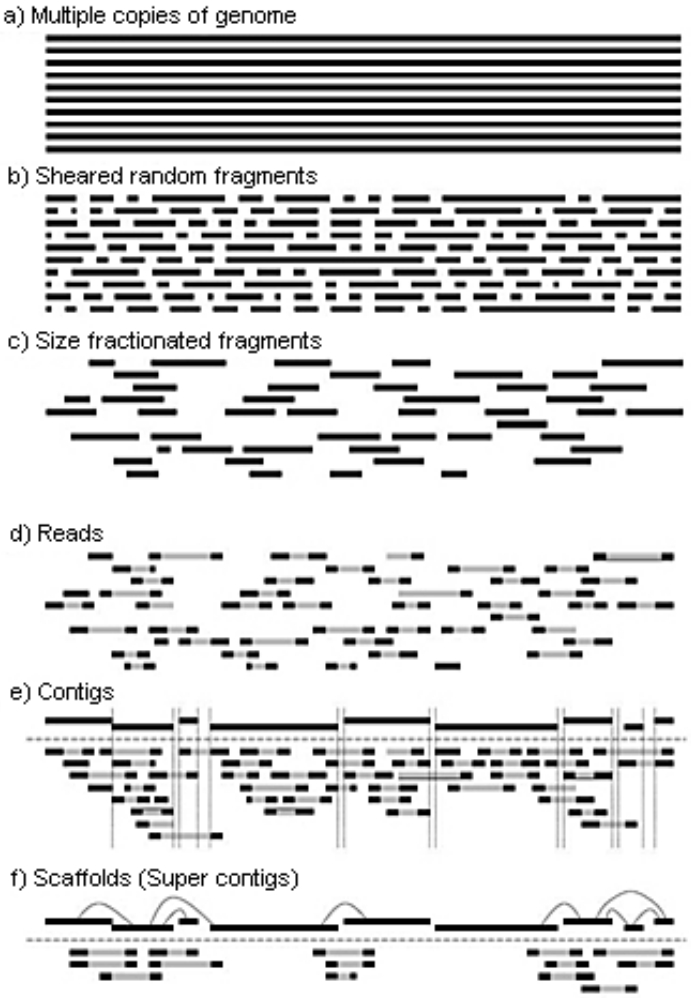

Figure 1.1. Outline of the DNA sequencing and assembly processes [4].

1.5 Approach to the project

One of the principal microbiological and genome research centers in the Canary Islands is the ***Instituto Universitario de Enfermedades Tropicales y Salud Pública de Canarias*** (**IUETSPC**, University Institute of Tropical Diseases and Public Health of the Canary Islands), located in San Cristóbal de La Laguna (Tenerife) and belonging to the *Universidad de La Laguna* [5]. This center has advanced equipment that allows microbial genome sequencing at low cost, using the **Ion Torrent PGM** (Personal Genome Machine) platform, commercialized by Life Technologies [6].

Ion Torrent sequencing is a method based on the detection of protons that are released during the DNA polymerization. This type of genome sequencing differs from the other ones; it does not use modified nucleotides and the detection is carried out not optically, but by measurement of changes in pH. This offers advantages as for sequencing speed and low operation costs, as well as certain limitations in base repeat detection and in the length of the DNA fragments employed. Nevertheless, the investment involved by this sequencing equipment has to be accompanied by the tune-up of proper tools which can help in the processes of assembly and subsequent analysis of genomes under study. Genome researchers frequently find problems to obtain and use specific computer applications for handling the information

resulting from sequencing. This is due to the fact that the great majority of open-use programs require complex installation and setting processes and lack a graphical interface, using shell commands and configuration files which are an obstacle to the progress of research.

Because of this, with the objective of making a better use of their equipment possible, the Applied Genomics Group of the IUETSPC has proposed to professors at the ***Escuela Técnica Superior de Ingeniería Informática*** (**ETSII**, Superior Technical School of Computer Engineering) of the *Universidad de La Laguna* [7] the development of an easy-to-use **integrated software platform** that offers an optimally configured processing and *de novo* assembly of genomic data obtained by Ion Torrent sequencing, also complemented with several result analysis stages. The determination to alleviate this need has given rise to the formation of a **multidisciplinary group**, integrated by professionals of both genomics and computer science. The initial study of the problem by this group led to the idea of turning this specific solution into a public Web service, in order to make it accessible to researchers not only of the IUETSPC, but from all over the world, independently of their geographic location. These objectives constitute the seed of the **IonGAP** project.

Chapter 2. State of the art

Abstract

- Currently, genome sequencing is performed by means of next-generation sequencing methods. The most commonly employed ones are pyrosequencing, ligation and two-base coding, sequencing by synthesis, and semiconductor sequencing.
- There exist more powerful techniques, belonging to the so-called third-generation sequencing, such as single-molecule real-time and nanopore sequencing.
- Genome assembly applications are based on graphs, and are classified into three groups: greedy graph-based assemblers, overlap-layout-consensus assemblers and de Bruijn graph assemblers.
- Before assembly, sequence reads may be previously treated by preprocessors or trimmers, which filter and trim them in order to improve the data set overall quality.
- It is possible to apply postprocessing (or genome finishing) tools on the assembly results in order to make them more accurate and reliable. Finishing techniques include scaffolding, correction of assembly errors, and detection or elimination of redundant sequences.

2.1 Genome sequencing systems

As mentioned in the previous chapter, the aim of genome sequencing is to determine the sequence of nucleotides of the DNA that forms a genome or part of it. Nowadays, **next-generation sequencing** (NGS) techniques allow to parallelize millions of sequencing reactions by breaking down DNA into short-length fragments.

The ideal sequencing technology should be fast, accurate, cheap and easy to apply. In recent years, the different existing sequencing technologies have experienced a rapid development, tirelessly competing to prevail in aspects such as performance, accuracy, cost or read size. An outline of the features and advantages of the main sequencing methods is presented below [8].

2.1.1 Pyrosequencing: Roche 454 system

Roche 454 was the first commercially successful next-generation system. This sequencer, released in 2005, uses **pyrosequencing** technology, which relies on the detection of pyrophosphate released during nucleotide incorporation to the chain.

The most outstanding advantage of Roche is its speed: it takes only 10 hours from sequencing start until completion. The read length (up to 700 bp) is also a distinguished character compared with other NGS systems. However, the high cost of reagents remains a challenge for this technology, which also has a moderate throughput, outputting only 0.7 Gb (gigabases) data per run. Another notable disadvantage of pyrosequencing is that homopolymer repeats (one same type of base repeated a number of times) are not properly detected.

2.1.2 Ligation and two-base coding: AB SOLiD system

SOLiD (Sequencing by Oligo Ligation Detection) was purchased by **Applied Biosystems** (now owned by Life Technologies) in 2006. This sequencer applies the technology of **two-base sequencing** based on **ligation sequencing**. Here, a pool of all possible oligonucleotides (very short sequences of nucleotides) of a fixed length are labeled according to the sequenced position. Oligonucleotides are annealed and ligated; the preferential ligation by DNA ligase for matching sequences results in a signal informative of the nucleotide at that position.

The SOLiD system offers a very high accuracy, but like other NGS systems, its computational infrastructure is expensive and not trivial to use; it requires an air-conditioned data center, computing cluster, skilled personnel in computing, distributed memory cluster, fast networks, and batch queue system. Each sequencer run takes 7 days and generates around 4 TB of raw data, with 85 or 100 bp reads.

2.1.3 Sequencing by synthesis: Illumina GA/HiSeq/MiSeq systems

In 2006, Solexa released the **Genome Analyzer** (GA), and in 2007 the company was purchased by **Illumina** [9]. Today, after outflanking and outlasting competitors, Illumina is on top of the genome sequencing business.

The GA adopts the technology of **sequencing by synthesis** (SBS). To determine the sequence in this method, four types of reversible terminator bases (RT-bases) are added and non-incorporated nucleotides are washed away. A camera takes images of the fluorescently labeled nucleotides; then this dye, along with the terminal 3' blocker, is chemically removed from the DNA, allowing for the next cycle to begin.

In early 2010, Illumina launched **HiSeq 2000**, which adopts the same sequencing strategy than GA. Compared with 454 and SOLiD, HiSeq 2000 is the cheapest in sequencing. On the other hand, **MiSeq**, a benchtop sequencer launched in 2011 which shared most technologies with HiSeq, is flexible to perform single 36 bp reads (120 MB output) up to 2 × 150 paired-end reads (1–1.5 GB output). In 2014, the company announced a multi-million dollar product, **HiSeqX Ten**, that could provide large-scale whole-genome sequencing for only $1000 per genome. The $1000 genome has been a catchphrase of the sequencing industry for years; the company claimed that forty such machines would be able to sequence more genomes in one year than had been produced by all other sequencers to date.

2.1.4 Semiconductor sequencing: Ion Torrent PGM system

Ion PGM (Personal Genome Machine) was released by **Ion Torrent** (now owned by Life Technologies) at the end of 2010. PGM is a benchtop sequencer that uses **semiconductor sequencing** technology. When a nucleotide is incorporated into the DNA molecules by the polymerase, a proton is released. By detecting the change in pH, PGM recognizes whether the nucleotide is added or not, as depicted in Figure 2.1.

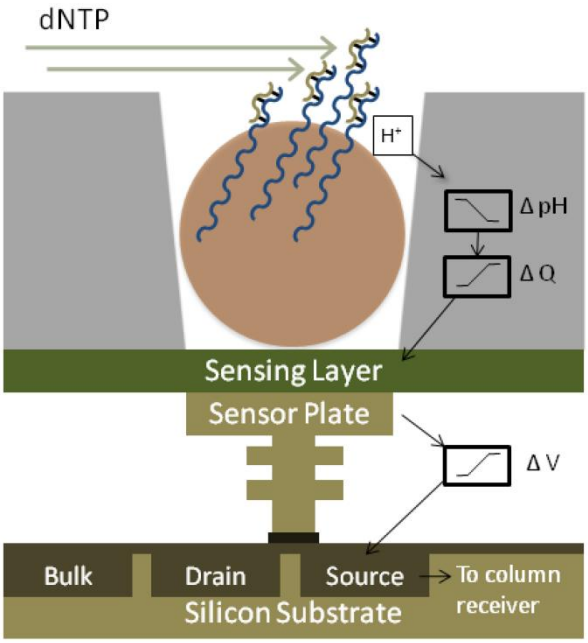

Figure 2.1. Schematic cross-section of a single well of an Ion Torrent sequencing chip. When a nucleotide incorporates, a proton releases and the pH of the well changes. A sensing layer detects the change in pH and translates the chemical signal to a digital signal [10].

PGM is the first commercial sequencing machine that does not require fluorescence and camera scanning, resulting in higher speed, lower cost, and smaller instrument size. Currently, it enables up to 400 bp reads in 8.5 hours, with a throughput of up to 2 Gb per run. In addition, the quality of Ion Torrent is more stable, while the quality of HiSeq 2000 decreases noticeably after 50 cycles, which may be caused by the decay of fluorescent signal as the read length increases. On the other hand, this technology suffers from the same homopolymer errors as Roche 454 does.

An exemplary application of the Ion Torrent PGM sequencer is the **identification of microbial pathogens**. This reason, among others like its low cost and high speed, has led the IUETSPC to include one Ion PGM in its equipment. **All the data sets employed in this project have been produced using this sequencing machine.**

2.1.5 Third-generation sequencing: single-molecule real-time and nanopore sequencing

While the increasing usage and new modification in next-generation sequencing, the **third-generation sequencing** is coming out with new insight in sequencing. An outstanding advantage of third-generation sequencing is that there is no need to create several copies of the DNA sequence before sequencing, which shortens DNA preparation time for sequencing.

Single-molecule real-time (SMRT) is the third-generation sequencing method developed by **Pacific Biosciences** [11], which makes use of modified enzyme and direct observation of the enzymatic reaction in real time using a camera. Comparing to second generation, **PacBio RS** (the first sequencer launched by this company) has several advantages. First, the sample preparation is very fast; it takes 4 to 6 hours instead of days. Second, the turnover rate is quite fast; runs are finished within a day. Third, the average read length is

1300 bp, which is longer than that of any second-generation sequencing technology. Although the throughput of the PacBio RS is lower than that of second-generation sequencers, this technology is quite useful for clinical laboratories, especially for microbiology research.

Nanopore sequencing is another third-generation sequencing method. A nanopore is a tiny biopore with diameter in nanoscale, which can be found in protein channel embedded on lipid bilayer which facilitates ion exchange. The core concept of nanopore sequencing involves putting a thread of single-stranded DNA across one of these pores, while an ionic flow is applied continuously. Nanopore sequencing possesses a number of advantages over existing commercialized next-generation sequencing technologies. For instance, it potentially reaches long read length (>5 Kbp), and detection of bases is fluorescent tag-free.

2.2 Genome assembly methods

An assembly is a hierarchical data structure that maps the sequence data to a presumed reconstruction of the target. It groups reads into contigs, which provide a multiple sequence alignment of reads plus the consensus sequence. The most widely accepted data file format for an assembly is FASTA, wherein contig consensus sequence can be represented by strings of the characters A, C, G, T, plus possibly other characters with special meaning.

Genome assemblers are computer applications designed to perform genome assembly; all of them are based on different kinds of graphs. A **graph** is an abstraction used widely in computer science, consisting of a set of nodes plus a set of edges between the nodes. If the edges may only be traversed in one direction, the graph is known as a directed graph. Nodes and edges may also be called vertices and arcs, respectively, and they may be assigned a variety of attributes and semantics.

The main characteristics of the three current categories of genome assemblers are described below [2].

2.2.1 Greedy graph-based assemblers

The first NGS assembly packages used **greedy algorithms**. These algorithms apply one basic operation: given any read or contig, add one more read or contig. The basic operation is repeated until no more operations are possible. Each operation uses the next highest-scoring overlap to make the next join. The scoring function measures, for instance, the number of matching bases in the overlap. Thus the contigs grow by greedy extension, always taking on the read that is found by following the highest-scoring overlap. The greedy algorithms can get stuck at local maxima if the contig at hand takes on reads that would have helped other contigs grow even larger. These algorithms are implicit graph algorithms; they drastically simplify the graph by considering only the high-scoring edges.

SSAKE [12] was the first short-read assembler. It was designed for unpaired short reads of uniform length, and it does not use a graph explicitly, but a lookup table of reads indexed by their prefixes. **SHARCGS** [13] and **VCAKE** [14] are other iterative extension algorithms.

2.2.2 Overlap-layout-consensus assemblers

An **overlap graph** represents the sequencing reads and their overlaps. The overlaps must be pre-computed by a series of (computationally expensive) pair-wise sequence alignments. Conceptually, the graph has nodes to represent the reads and edges to represent overlaps. In practice, the graph might have distinct elements or attributes to distinguish the 5′ and 3′ ends of reads, the forward and reverse complement sequences of reads, the lengths of reads, the lengths of overlaps, and the type of overlap (suffix-to-prefix or containment). Paths through the graph are the potential contigs, and can be converted to sequence. Figure 2.2 presents an example of an OLC assembly graph.

The **overlap-layout-consensus** (OLC) approach performs genome assembly using an overlap graph. It was typical of the Sanger-data assemblers, and it was optimized for large genomes in software including **Celera Assembler** [15] and **Arachne** [16].

Their operation has three phases. First, an overlap discovery involves all-against-all, pair-wise read comparison, splitting the reads in short fragments called K-mers. The software pre-computes K-mer content across all reads, selects overlap candidates that share K-mers, and computes alignments using the K-mers as alignment seeds. Overlap discovery is sensitive to settings of K-mer size, minimum overlap length, and minimum percent identity required for an overlap. Larger values of these three parameters lead to more accurate but shorter contigs.

The second stage is the construction and manipulation of an overlap graph, which leads to an approximate read layout. Then the final step, multiple sequence alignment (MSA), determines the precise layout and then the consensus sequence. There is no known efficient method to compute the optimal MSA. Therefore, the consensus phase uses progressive pair-wise alignments guided by, for instance, the approximate read layout.

Some distinguished modern OLC assemblers are **Newbler** [17], **Edena** [18] and **MIRA** [19].

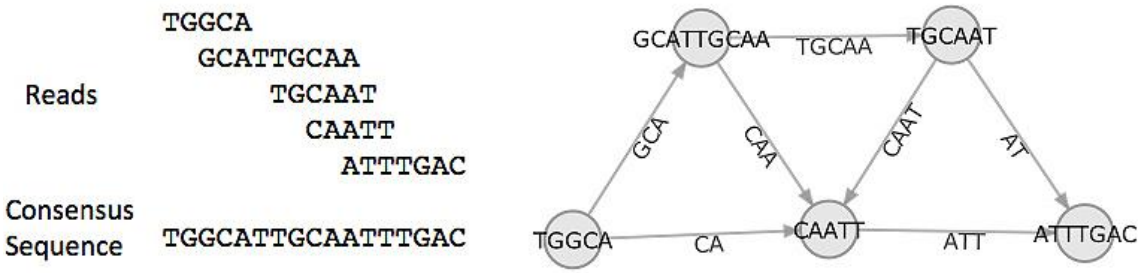

Figure 2.2. An OLC assembly graph. Nodes are complete reads and edges connect reads that overlap (in an actual OLC assembly graph, reads and overlaps would be much larger) [20].

2.2.3 De Bruijn graph assemblers

The **de Bruijn graph** (DBG) was developed outside the realm of DNA sequencing to represent strings from a finite alphabet. The nodes represent all possible fixed-length strings. The edges represent suffix-to-prefix perfect overlaps.

A **K-mer graph** is a form of de Bruijn graph; its nodes represent all the fixed-length subsequences drawn from a larger sequence, and its edges represent all the fixed-length overlaps between subsequences that were consecutive in the larger sequence. In one

formulation, there is one edge for the K-mer that starts at each base (excluding the last K–1 bases), and the nodes represent overlaps of K–1 bases. Alternately, there is one node representing the K-mer that starts at each base, so the edges represent overlaps of K–1 bases. By construction, the graph contains a path corresponding to the original sequence. Figure 2.3 shows these two formulations of a de Bruijn graph.

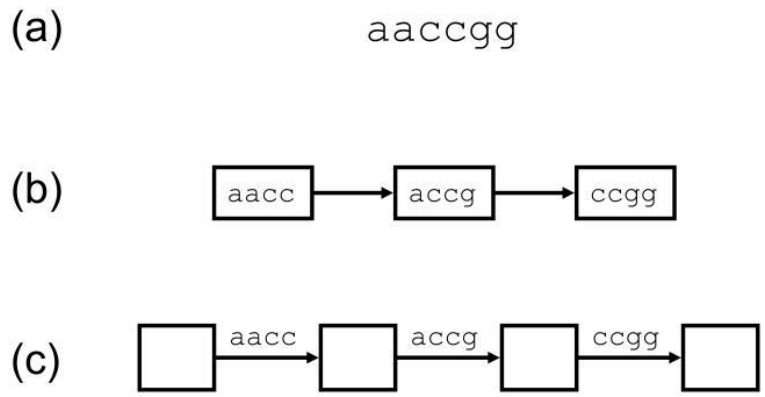

Figure 2.3. A read represented by K-mer graphs, with K=4 (larger values of K are used for real data). (b) The graph has a node for every K-mer in the read plus a directed edge for every pair of K-mers that overlap by K–1 bases in the read. (c) An equivalent graph has an edge for every K-mer in the read and the nodes implicitly represent overlaps of K–1 bases [2].

The **DBG** approach to assembly is most widely applied to the short reads from the Solexa/Illumina and SOLiD platforms. It relies on K-mer graphs, which do not require all-against-all overlap discovery and compress redundant sequence. Conversely, the K-mer graph does contain actual sequence and the graph can exhaust available memory on large genomes. Distributed memory approaches may alleviate this constraint.

Given perfect data (error-free K-mers providing full coverage and spanning every repeat) the K-mer graph would be a de Bruijn graph and it would contain an Eulerian path, that is, a path that traverses each edge exactly once. The path would be trivial to find making the assembly problem trivial by extension. Of course, K-mer graphs built from real sequencing data are more complicated.

To the extent that the data is ideal, assembly is a by-product of the graph construction. The graph construction phase proceeds quickly using a constant-time hash table lookup for the existence of each K-mer in the data stream. Although the hash table consumes extra memory, the K-mer graph itself stores each K-mer at most once, no matter how many times the K-mer occurs in the reads. In terms of computer memory, the graph is smaller than the input reads to the extent that the reads share K-mers.

Many of the most modern and effective genome assemblers, such as **EULER** [21], **Velvet** [22], **ABySS** [23], **ALLPATHS** [24] and **SOAPdenovo** [25], apply the DBG approach.

2.3 Data preprocessing techniques

Raw data generated by genome sequencers has defects, such as the presence of incorrect bases due to sequencing errors, regions with low quality score, and sequences corresponding to sequencing adapters (see below). Some genome assemblers are capable of detecting these

problems in the read set and acting according to them. However, most of them need the reads to be previously processed by specialized tools, which are known as **preprocessors** or **trimmers**.

There are different methods for improving the quality of reads before the assembly. The first one is to eliminate **sequencing adapters**, very short DNA sequences used by the sequencing platform in order to amplify and sequence the genome fragments, thereby generating the read set, and often remain attached to one or both ends of the reads. The easier way to eliminate them is performing a trimming of a fixed number of bases (equal or close to the adapter length) at the read ends. Another option is using an adapter library which contains the sequence of the employed adapters, so that they can be found and selectively suppressed from the affected reads.

Incorrect bases due to sequencing errors are usually not detectable in this phase. The genome assembler has to correct them by detecting inconsistencies between the different reads that cover a same portion of the genome. On the other hand, given that quality scores decrease as the DNA strand of the fragment is sequenced, low-quality regions usually appear near to the 3' end of the reads. These regions can be suppressed either by trimming the reads to a specified maximum length, deduced from a quality distribution analysis, or employing a **sliding window** trimming method. This technique is based on a progressive quality analysis along the read, using a window of fixed (usually short) length in base pairs. The window is initially placed at the beginning of the read (5' end). A quality function of the bases inside the window, which is often the maximum, minimum or average quality, is evaluated. If its value is less (or equal) than an established threshold, the read is trimmed at that point, and otherwise, the window is shifted one base towards the 3' end. In this way, in the event of a trimming, the new 3' end would be the base immediately previous to the beginning of the window.

Some well-known preprocessing and quality control tools are **PRINSEQ** [26], **Quake** [27], **Trimmomatic** [28], **ConDeTri** [29] and **FastQC** [30].

Another outstanding preprocessing method for raw data is called **digital normalization** [31]. This technique progressively removes high-coverage reads from shotgun data sets, normalizing average coverage to a specified value. By doing so, it reduces sampling variation while removing reads, and also removes the many errors contained within those reads. This data and error reduction results in dramatically decreased computational requirements for *de novo* assembly. Currently, the only software tool that performs the digital normalization technique is **khmer** [32].

2.4 Genome finishing techniques

Most genome assembly applications generate a set of contigs in one or more file formats. In many occasions, it is advisable to treat this result in order to increase its accuracy and reliability. Earlier assembly algorithms followed an overlap-layout-consensus paradigm, where *consensus* refers to fixing errors in the contigs. Since this paradigm faces difficulties in short-read assembly, most NGS assemblers employ a de Bruijn graph approach that effectively deals with large amounts of data. However, most NGS assemblers neglect the consensus step, that is, there exists no post-processing of the contigs in many popular assemblers [33]. Owing to this, nowadays it is usually necessary an external **postprocessing** or **genome finishing** phase.

The first assembly postprocessing method to be considered is **scaffolding**, whose aim is to determine the relative position of the contigs and to estimate the length of the gaps between them, generating discontinuous groups of contigs named **scaffolds** (or supercontigs). Scaffolding technique is widely utilized in order to increase the accuracy of assemblies; nevertheless, it requires a set of mate-pair reads (paired-end reads with a long gap between them), so one of the reads in a pair is located in one contig, and the other in another contig, making it possible the determination of the relative position of contigs, as well as the size of the gap that separates them. An increasing number of assemblers include their own scaffolding modules in order to complement genome assembly. Some well-known scaffolding tools are **Bambus** [34], **ABySS** [23], **Opera** [35] and **SSPACE** [36].

If mate-pair reads are not available, an alternative to scaffolding lies in **aligning the contigs to the sequence of a reference genome** (if there exists any), thereby allowing to ascertain the relative position of contigs and also the size of gaps, using as a reference the spaces between the alignment of contigs to the reference genome (assuming the assembled genome does not differ from the reference one in the areas corresponding to these gaps). Some tools which are capable of performing this process are **ABACAS** [37] and **Mauve** [38], [39].

Another kind of genome finishing is the so-called **positional reassembly**. This process is used to refine an initial assembly of short-read data by using approximate positions of reads in contigs. It takes as input an assembled contig, the paired-end reads that align to that contig and the approximate positions where they aligned, and returns a refined contig. A specialized tool for performing positional reassembly is **SEQuel** [33]. The algorithm behind SEQuel makes use of a graph structure called the positional de Bruijn graph, which models K-mers within reads while incorporating their approximate positions into the model.

Finally, another way to refine the assembly is by means of tools designed to detect and correct repetitive sequences that can be caused by errors of the assembler, as well as redundant contigs, whose sequence is already present in another contig of the same assembly. An example of tool designed to selectively eliminate redundant sequences from the collection of contigs generated by *de novo* assembly is **Simplifier** [40].

Chapter 3. The genome assembler

Abstract

- The project development is divided into one first stage of comparative study of genomic data processing applications, followed by the integration of analysis tools and the development of the final Web application.
- The comparative study of some well-known genome assemblers has determined that MIRA is the assembler which works the best with Ion Torrent data.
- The study of MIRA parameters demonstrates that the assembler is set by default on its optimal configuration.

3.1 Work development scheme

The development of this project can be structured in two fundamental parts. The first one is carrying out a **comparative study** aimed at finding the best combination of tools for performing the processing and assembly of reads, as well as the optimal configuration of the former. The second phase corresponds to the integration of the **analysis tools** that are required in order to complete the functionality of the platform, together with the design of the user interface and the **deployment of the service** on a Web server.

The comparative study of tools is directed to obtain a specific **pipeline** for genome assembly and analysis of Ion Torrent data. The phases of a genomic data workflow normally are: data preprocessing, genome assembly, assembly results postprocessing and genome analysis. Figure 3.1 shows an example of this kind of workflow.

During the process of determination of the proper combination of tools for treating data from genome sequencing, different types of computer application were used. Both the comparative tests and the platform development have been carried out on a conventional laptop equipped with a 2.20 GHz Intel Core i7 processor and 8 GB RAM, using the **Ubuntu 12.04** 64 bits operating system.

The data set utilized for all the tests in this study corresponds to a FASTQ file containing 686800 reads, which were sequenced using the Ion Torrent PGM platform owned by the IUETSPC, from the genome of a ***Streptococcus agalactiae*** strain isolated at the *Hospital Universitario Ramón y Cajal* (Madrid). *S. agalactiae* is a Gram-positive bacterium that can be found in the digestive, urinary and genital systems of healthy individuals, although it can cause the death of newborn babies (see Figure 3.2).

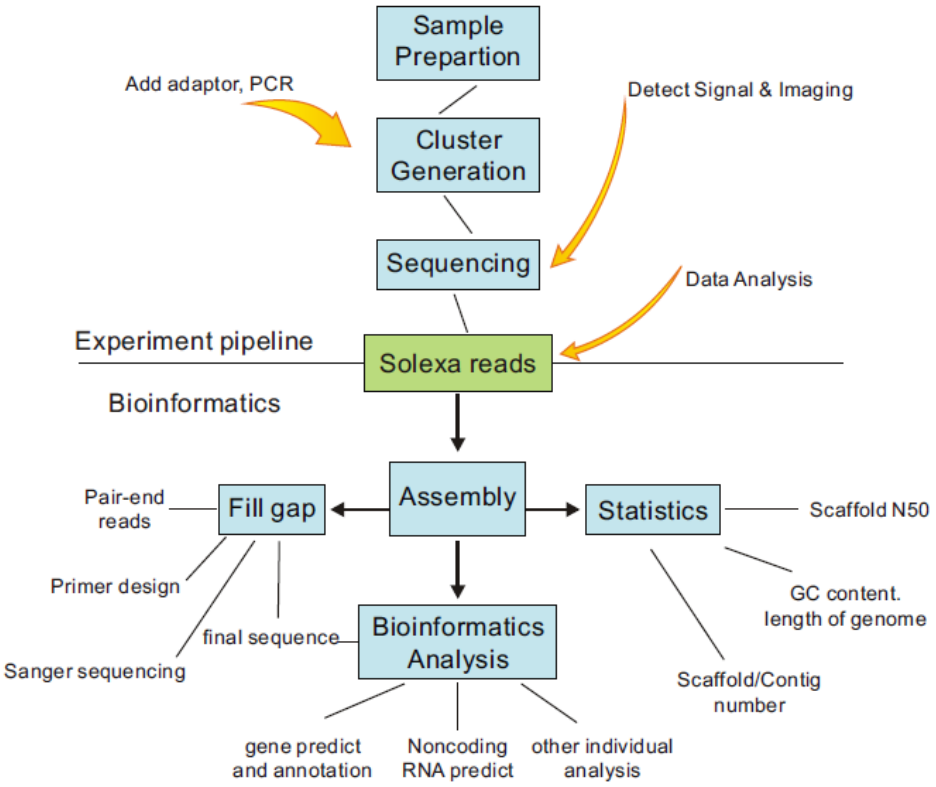

Figure 3.1. Example of genomic data workflow (Solexa/Illumina technology), including genome sequencing, assembly and analysis [41].

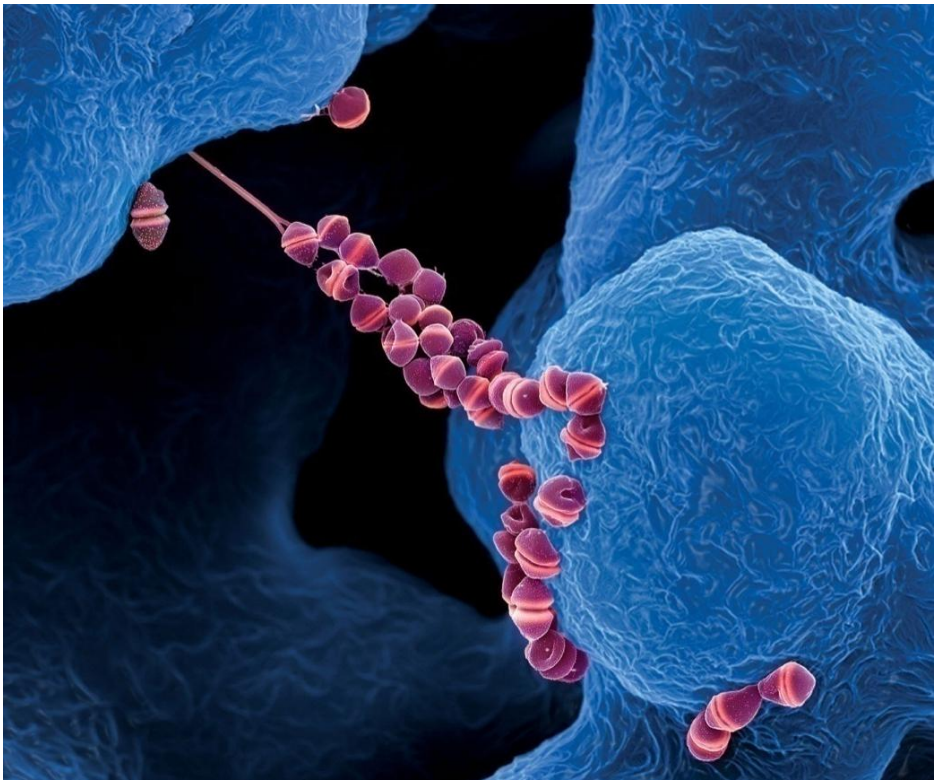

Figure 3.2. A colorized electron microscope image captures chains of streptococcus in a laboratory sample [42].

3.2 Comparative study of genome assemblers

3.2.1 Materials and methods

Genome assembly constitutes the fundamental process in a workflow for genomic data, as well as the core of the integrated Web service which is the aim of this project. Due to this, the first step in order to obtain the desired pipeline of tools is the selection of a set of genome assemblers, whose effectiveness on Ion Torrent data has been evaluated by means of an optimization process of each tool and a comparison between their results.

With the purpose of discovering the most suitable assembler for the employed type of reads (Ion Torrent single-end long reads), some well-known genome assemblers were selected. According to the sequencing technology and read type requirements, the assemblers considered for the study were: **MIRA** [19], **Celera (WGS-Assembler)** [15], **SGA** [43], **ABySS** [23], **Ray** [44], **Velvet** [22], **SparseAssembler** [45] and **Minia** [46]. The first three of them are based on an overlap graph, and depend mainly on the imposed minimum length of the overlap between reads; whereas the rest of them follow an approach based on a de Bruijn graph, and their main parameter is the K-mer length (see section 2.2, *Genome assembly methods*). On the other hand, the last two assemblers stand out because they are focused to reduce memory consumption, which is currently a major bottleneck of the genome assembly process. The metrics used to evaluate the result of all the comparison studies in this project have been obtained by means of the **abyss-fac** script, that can be found together with the ABySS v1.3.6 software, and calculates a series of assembly statistics taking only the set of contigs of length equal to or higher than 500 bp.

3.2.2 Results and discussion

The results from the comparative study of assemblers prove without doubt that **MIRA** is the most effective assembler when it comes to work with Ion Torrent data. Figures 3.3 and 3.4 outline the effectiveness results of the evaluated assemblers, according to the number of generated contigs of length equal to or higher than 500 bp, and the **N50 length**, that is defined as the length for which 50% of the entire assembly is contained in contigs of length equal to or higher than this value. An assembly is considered superior in quality than other one if it covers an equal or larger part of the genome with less contigs, so a low number of contigs and a high N50 (and other statistics such as N20, N80, N95) value are pursued.

Celera Assembler is not included in the study results because it was not able to perform the assembly with the available memory (8 GB). On the other hand, in both OLC and DBG assemblers, the **overlap** or the **K-mer length**, respectively, has been adjusted to its optimal value depending on the case, so the values shown in the figures represent the results from the best case scenario of each assembler. Appendix A1 contains all the result tables relating to the optimization process of each assembler. These tables show a greater variety of quality metrics than the ones exposed here.

It should be noted that the results from MIRA (the first assembler evaluated) correspond to the assembly performed directly on the *S. agalactiae* raw data set, whereas for the rest of programs the reads have been **preprocessed** using PRINSEQ, therefore the assembled reads have a minimum length of 40 bp and a maximum length of 250 bp, and in addition, the first 20

bp, corresponding to the sequencing adapter, have been suppressed. The justification for this measure lies in the quality improvement obtained in the results by using this preprocessing, which is based on a previous quality distribution analysis of the reads. If this step were suppressed, the assembly results would be slightly or significantly worse (read trimming effects on MIRA will be analyzed later). This is because most assemblers that are based on a de Bruijn graph neglect some or all data preprocessing steps (unlike MIRA), what makes then more sensitive to reads that contain sequencing errors or low quality regions.

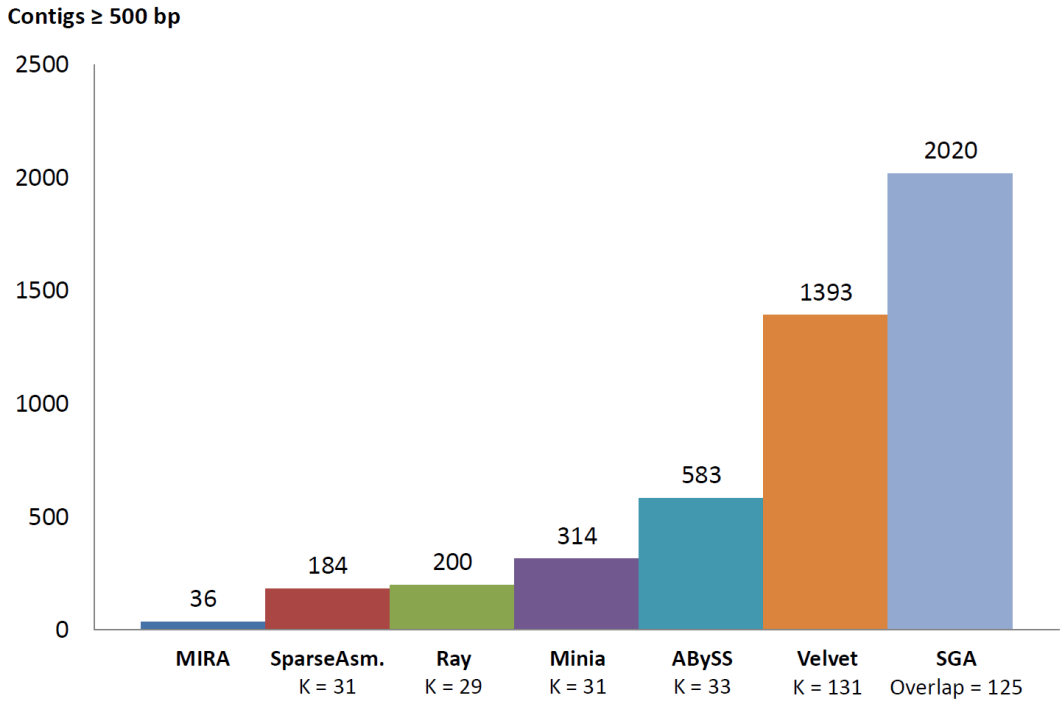

Figure 3.3. Results from the comparative study of assemblers, according to the number of contigs of length equal to or higher than 500 bp.

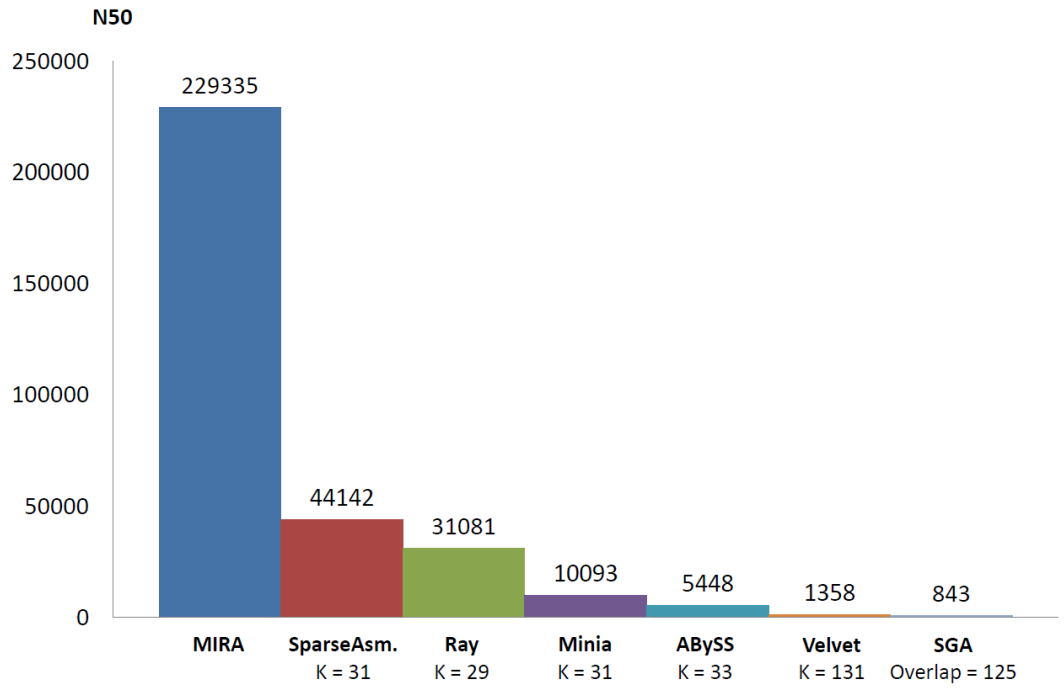

Figure 3.4. Results from the comparative study of assemblers, according to the N50 contig length (in bp).

The fundamental conclusion from this study is that **only one of the selected assemblers has been capable of offering acceptable results** for reads obtained using the Ion Torrent technology, while the majority of the other programs produced only a high number of contigs that are too short to be very useful. The main reasons for this problem are, presumably, the sequencing technology and the type of reads employed. Currently, most sequencing platforms make use of sets of paired-end short reads, often generated using the Illumina technology; however, in this case the data consists of Ion Torrent single-end long reads (see section 2.1, *Genome sequencing systems*). Ion Torrent technology allows considerably longer reads which have an error distribution very different from that of other technologies. Therefore, although Ion Torrent offers the advantages of low cost and high sequencing speed, its use may be hindered by the fact that many genome processing and assembly tools are designed to work with Illumina data (considering that it is the most common technology in the market). Besides, paired-end reads are nowadays more widely used than single-end reads.

Some of the evaluated assemblers are capable of working with Ion Torrent or 454/Roche reads, whose error distribution is the most similar one to that of the former. However, **MIRA** has proven itself to be the only assembler that is really adapted to this kind of reads (and to many others as well). Another conclusion from the study is that assemblers based on a de Bruijn graph (the most recent and widespread tendency) are not able to work with Ion Torrent single-end long reads. It has not been possible to find a K-mer length for which this kind of data can be used to generate quality results. However, several studies have demonstrated the high effectiveness of all the evaluated assemblers, when short reads are utilized [47], [48], [49], [50]. This result supports the idea that the OLC approach to assembly, used by MIRA, is more suitable for long reads, whereas the DBG approach, followed by many modern assemblers, is intended for short-read assembly [33] (see section 2.2, *Genome assembly methods*).

3.3 Genome assembler optimization

3.3.1 Structure and functioning of the MIRA assembler

The MIRA assembler applies the **overlap-layout-consensus** (OLC) paradigm, which performs genome assembly using an overlap graph. In order to build and review the graph, MIRA employs (among other techniques) an adapted version of the **Smith-Waterman alignment algorithm** [51]. Figure 3.5 depicts an approximate overview of the developed strategy as well as the different phases involved while assembling a project [52]:

1. The reads constituting an assembly project are preprocessed by external programs to perform different refinement steps to the original read data, for instance, sequencing vector clipping, standard repeat tagging, quality clipping, etc. A multitude of programs is available, each of them being very specialized for a certain task.
2. The high confidence region (HCR) of each read is compared with a quick heuristic algorithm to the HCR of every other read to see if it could match and have overlapping parts. All the possible overlaps form one or several initial building graphs.
3. The reads in the initial building graphs which could have overlaps are being reviewed with an adapted Smith-Waterman alignment algorithm (banded version). Obvious mismatches are

rejected and removed from the initial building graph, the accepted read-pairs are inserted into one or several alignment graphs. These alignment graphs define all the assemblies that are possible with the given reads.

4. Optional pre-assembly read extension step: the assembler can try to extend HCRs of reads by analyzing the overlap pairs from the previous alignments. This can help to elongate reads which were cut back too much by conservative quality clipping mechanisms during pre-assembly preprocessing. The confirmation of a base sequence by two similar reads combines the advantage of single read quality clips and coverage security.
5. A contig is assembled by building a preliminary partial path through the alignment graph and then adding the most probable overlap candidate to a given contig. Contigs can reject reads if these introduce to many unexpected and high profile errors in the existing consensus. Errors in regions known as dangerous (for example tagged standard repeats like Alu elements and other common repeats of the genome) get additional attention by performing simple signal analysis when alignment discrepancies occur.
6. Contigs can be optionally analyzed and corrected by an incorporated version of an automatic editor (EdIt). This editor analyzes fault regions in contigs and corrects base call errors (and alignment errors resulting from these) by analyzing the underlying trace signals of the reads and calculating probabilities with respect to the coverage of the contig.
7. Also optionally, long term repeats that were misassembled can be searched for. The assembler looks for typical misassembly patterns provoked by repeats (like mismatch columns). Bases that are dissimilar in the different repeats get tagged as possible repeat marker base (PRMB) to help the assembly algorithm in subsequent iterations and then the wrongly assembled contigs get dismantled for further reassembly.
8. The resulting project is written out to standard file formats for further post-assembly processing.

The overall structure of the assembly strategy (that is, the iterative nature of its design and the steps concentrating on high quality parts first) reflects the needs imposed to it by the aims of identifying repeats to avoid misassemblies and decreasing the error rate of the final consensus sequence. Each subpart of the assembly strategy is embedded into the overall scheme, which in turn means that, sometimes, algorithms show unexpected (and mostly undesired) side effects when used within the whole system.

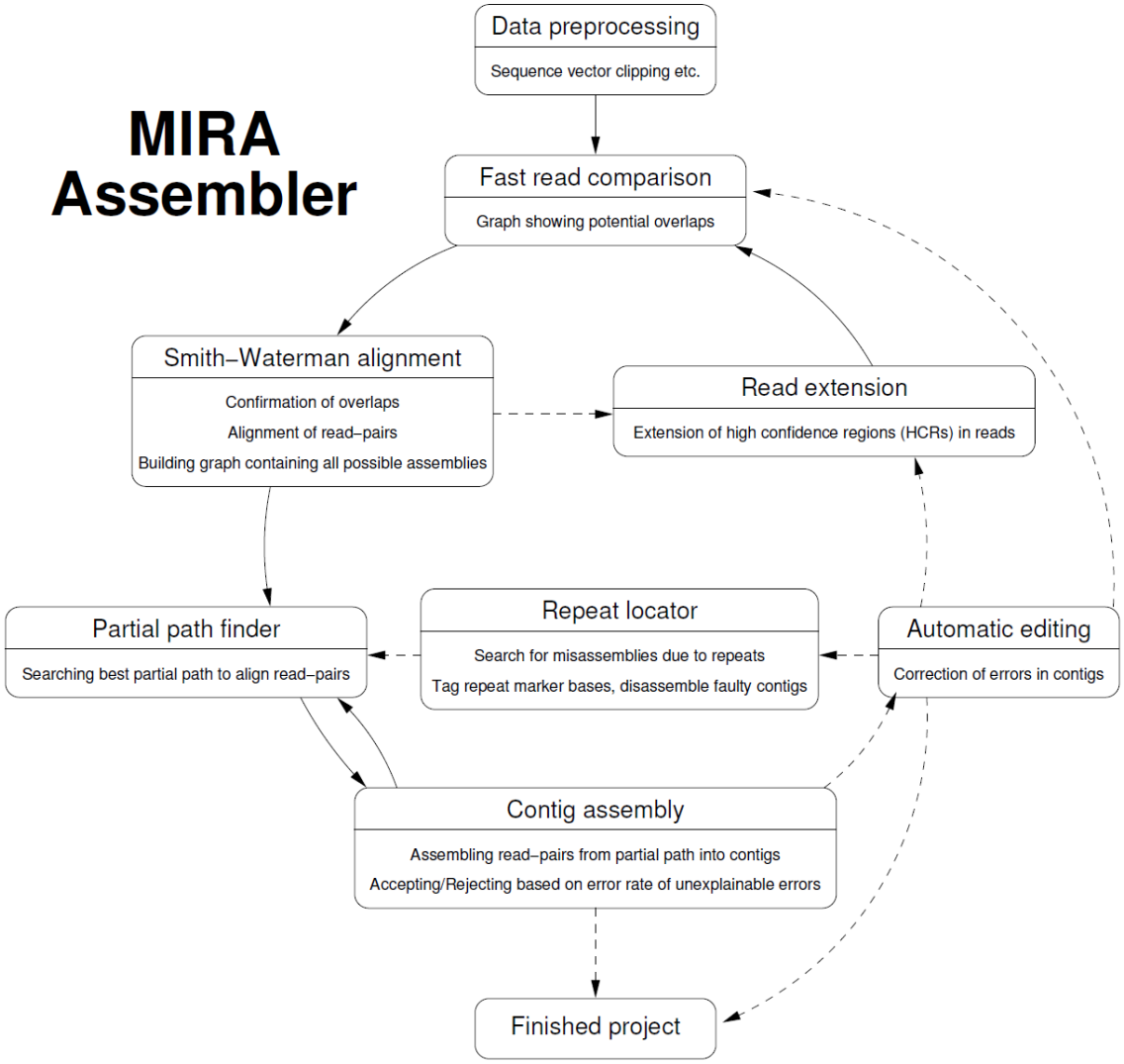

Figure 3.5. Phases of a MIRA assembly cycle. Plain arrows show imperative pathways, dashed arrows denote optional pathways [52].

3.3.2 Assembly parameters

Once determined that MIRA is the most suitable assembler for being included in the platform, it is possible to proceed to find an **optimal adjustment of its parameters**. The studied parameters were those ones that most critically affect the assembly, and are in turn the closest to the criteria of genomics researchers. These parameters are: number of assembly iterations; uniform read distribution; separation of long repeats in different contigs; maximum number of times a contig can be rebuilt during an iteration; minimum repeat length; minimum size of a contig for being considered as "large"; minimum number of reads per contig; minimum read length; minimum overlap length and minimum overlap score.

3.3.3 Results and discussion

The study of MIRA parameters demonstrates that the assembler is set by default in its optimal configuration; the variation of any of the mentioned parameters causes a minor or a notable

decline in the assembly quality, according to the variation range. Appendix A2 contains all the result tables relating to the assembly parameter optimization process.

In some cases, metrics seem to show an assembly which is superior in quality than that obtained using the default configuration; nevertheless, when this happened, a detailed analysis of the contigs revealed an incorrect assembly, in which there existed an issue that has became informally known as ***unacceptable leaps***. This term refers to the case in which a contig is made up of two regions whose location in the reference genome is significantly distant, and which are often joined together in the contig in an opposite direction to the expectable one. It should be noted that this situation does not take place if the default assembly configuration is used (at least for the test data set).

The only parameter whose readjustment might be interesting is the **minimum size of "large" contigs** (by default, 500 bp), because these contigs are gathered in a separate file and constitute the main output of MIRA. The value of this parameter does not affect the assembly process, but the output set of contigs, so it may be convenient to increase it in some cases in order to reduce the number of contigs, if the shortest ones are not of interest.

The assembly obtained by using this MIRA configuration is significantly more coherent and satisfactory than the one generated by the built-in assembly module of the IUETSPC's Ion Torrent platform; although it is necessary to make it clear that genome assembly is a non-deterministic process, so two different executions of the assembler using the same data and configuration can generate different contigs.

In order to compare the results from the optimal MIRA assembly with those from a theoretical ideal assembly, the Lander-Waterman model has been used. This model estimates some ideal assembly statistics, such as the genome percentage sequenced, the number of contigs (or islands) and gaps, or their total length. Lander and Waterman suggested in their 1988 article [53] that the number of times a base is sequenced follows a Poisson distribution. There are two key assumptions they made:

1. Reads will be randomly distributed in the genome.
2. The ability to detect an overlap between two truly overlapping reads does not vary from clone to clone.

The calculations detailed in this model have been used to help planning shotgun sequencing projects. On this occasion, these have been employed to study the quality of the obtained assembly with respect to an ideal one, as Table 3.1 shows [54], [55].

Description [units]	Variable / Parameter	Theoretical assembly results	Actual assembly results
Genome size [bp]	G	2127839	2127839
# of reads	N	686800	686800
# of bases in all reads [bp]	BP	121736955	121736955
Average read length [bp]	$L = BP / N$	177	**177**
Coverage [##x]	$c = L \cdot N / G$	57	**49.86**
P(*"Any base not sequenced"*)	$P(X=0) = e^{-c} = 1 / e^{c}$	0.00	0.00
P(*"All bases sequenced"*)	$1 - P(X=0) = 1 - 1 / e^{c}$	1.00	1.00
Expected genome % sequenced	$(1 - 1 / e^{c}) \cdot 100$	100.00%	**92.76%**
# of bases in contigs = Expected size of sequenced region [bp]	$G \cdot (1 - 1 / e^{c})$	2127839	**2189381**
# of bases in gaps [bp]	G / e^{c}	0	**63376**
Expected # of contigs	N / e^{c}	1	**36**
Expected # of reads per contig [bp]	e^{c}	686800	**16910**
Average contig size [bp]	$(e^{c} - 1) \cdot L / c$	2127839	**60816 (N50 = 229335)**

Table 3.1. Theoretical statistics from the Lander-Waterman model beside empirical values from the *S. agalactiae* assembly. The empirically obtained (not calculated) values are highlighted in bold type, while the disregarded formulas (which do not provide valid values) are colored in gray.

It can be observed that, although the theoretical expected result for a uniform 57x coverage consists of a complete genome sequencing in a single contig (totally contiguous genome), the reality is that reads are not uniformly distributed, even though their distribution is considerably regular. Due to this, the real average coverage is 49.86x, having a maximum value of 377x. The expected number of contigs also differs from the real one (36), although their mean size surpasses that of many assembly projects of recent years. It is also significant that 7.24% of the reference genome has not been sequenced, but this is in part because the genome of the isolated bacterium slightly differs from that of the reference bacterial strain (*S. agalactiae* A909). In fact, the assembled contigs in turn contain 8.76% of bases which are not present in the reference genome, and because of this, the assembled sequence (consensus) is larger than the reference sequence. Considering this, when it comes to calculate the number of bases in gaps, only the gaps *between* contigs have been taken into account, not those ones present inside the contigs once they have been aligned to the reference sequence (since they are not actual assembly gaps, but discrepancies between the aligned sequences). On the other hand, the theoretical values corresponding to the last three parameters have been directly deduced from the previous ones, because the formulas to obtain them seem not to be valid for such a high coverage value as that of the employed data set.

Chapter 4. A genome assembly and analysis pipeline

Abstract

- The study of read preprocessing options, both external and internal to the assembler, proves that the application of this kind of techniques harms the overall assembly quality.
- After trying an assembly postprocessing phase based on the elimination of redundant sequences, the result from this process has been considered unreliable and hard to assess.
- The designed workflow has been completed with the inclusion of different genome analysis tools, which provide an added value to the assembly results, facilitating their interpretation.

4.1 From assembly to assembly pipeline

As mentioned in the previous chapter, the genome assembler is the fundamental component of any genomic data workflow. However, it is not enough for providing the process with a relevant scientific usefulness by itself. Genome assembly generates a set of contigs, which are no more than long sequences of nucleotides; almost endless strings of characters A, C, G and T that are, on their own, indecipherable even for genomics experts. In order to reveal the information contained inside the sequence of the contigs, they have to be refined, corrected, ordered, compared to the sequence of a reference genome (if there exists any) and complemented with the information of coding regions (or CDS, *Coding DNA Sequence*), specific genes coded by these regions and, ultimately, proteins synthesized from each one of these genes. On the other hand, it is fundamental to ensure an acceptable precision level in contigs, by discarding bad quality regions from the input reads, which will be probably harmful for the overall quality and accuracy of the assembly. All these complementary processes turn into a series of **computational tools** that are set out around the workflow core (the genome assembler), defining a specific **genome assembly, processing and analysis pipeline**. This pipeline means the difference between an indecipherable sequence of letters and an useful data set that allows researchers to extract conclusions and make discoveries which can contribute to advances in the knowledge and treatment of genetic or infectious diseases.

4.2 Preprocessing: Comparative study of trimmers

Once the study of genome assemblers is complete and the best assembly configuration has been found out, it is possible to proceed to evaluate the effects of the use of different **data preprocessing** tools and criteria on the assembly, with the purpose of improving its overall quality and discarding erroneous information.

4.2.1 Materials and methods

In addition to the quality control parameters present in the MIRA assembler, a set of data preprocessors (or trimmers) were incorporated into the study, in order to check their effect on

the assembly results. The selected trimmers were **PRINSEQ** [26], **ERNE-filter** [56] and **Trimmomatic** [28]. It was not necessary to consider more options because the operation functionalities do not really change between one tool and another.

According to the logical criteria provided by the IUETSPC's researchers, as well as to quality distribution analysis of the reads, preprocessing was applied in three different ways: establishing a minimum read length threshold (normally, 40 bp), in order to eliminate those reads too short to be useful; performing a trimming on the 5' read end (normally, 20 bp) to suppress the bases corresponding to the sequencing adapters; and establishing a maximum read length threshold (normally, 250 bp), in order to eliminate those bases with low quality scores, which tend to gather near the 3' read end.

4.2.2 Results and discussion

Table 4.1 shows that the result of applying the selected trimmers on the input data are always inferior to those obtained without preprocessing, independently of the program employed and of whether the assembler default preprocessing options are disabled or not. The results in the table correspond to the configurations detailed below.

A) **MIRA**: default configuration, no external preprocessing.
B) **PRINSEQ**: minimum length of 40 bp, 5' trimming of 20 bp, maximum length of 250 bp.
C) **PRINSEQ**: minimum length of 40 bp, 5' trimming of 20 bp, maximum length of 250 bp (disabling all MIRA preprocessing options).
D) **PRINSEQ**: minimum length of 40 bp, 5' trimming of 20 bp, maximum length of 350 bp.
E) **PRINSEQ**: 5' trimming of 20 bp.
F) **PRINSEQ**: 5' trimming of 40 bp.
G) **ERNE-filter** (not configurable).
H) **Trimmomatic**: minimum length of 40 bp, 5' trimming of 20 bp, maximum length of 250 bp.
I) **Trimmomatic**: minimum length of 40 bp, 5' trimming of 20 bp, maximum length of 250 bp (disabling all MIRA preprocessing options).
J) **Trimmomatic**: minimum length of 40 bp, 5' trimming of 20 bp, 3' trimming by sliding window 4 bp long and with quality threshold of 15 (Phred+33 scale).

	Contigs ≥ 500 bp	Contigs ≥ N50	Min	N80	N50	N20	Max	Sum
A	36	4	559	71366	229335	526123	526123	2189373
B	48	6	533	46300	121640	240705	247411	2188023
C	43	5	553	50180	130887	307339	315773	2187611
D	43	5	577	51631	177491	284219	313805	2191998
E	41	4	511	51631	181278	342667	399514	2188983
F	52	7	506	39892	98867	213233	244055	2192987
G	34	4	509	66392	155176	526055	526055	2188731
H	44	5	502	52239	160778	309464	313803	2189019
I	45	4	526	50290	181282	309462	342667	2191097
J	55	10	583	35112	71380	149774	159655	2185317

Table 4.1. Results from the comparative study of trimmers. The first two columns represent the number of contigs, and the other ones, total lengths expressed in bp.

Facing the possibility that some preprocessed assemblies could be more reliable than the metrics evaluated here seem to show, some of these assemblies were compared between them with respect to the reference sequence, using the **Mauve Assembly Metrics** [57] tool. In Figures 4.1, 4.2 and 4.3, some of the graphs resulting from these comparison can be observed, thereby proving that the quality of those preprocessing cases which could arouse any doubt is definitively inferior than that of the assembly without external preprocessing. The evaluated assemblies correspond to the configurations listed below.

A) **MIRA**: default configuration, no external preprocessing.
B) **MIRA**: no external preprocessing, minimum of 35 reads per contig imposed (this is an assembly from the MIRA optimization stage whose quality also generated doubts).
C) **PRINSEQ**: minimum length of 40 bp, 5' trimming of 20 bp, maximum length of 300 bp.
D) **PRINSEQ**: 5' trimming of 40 bp.

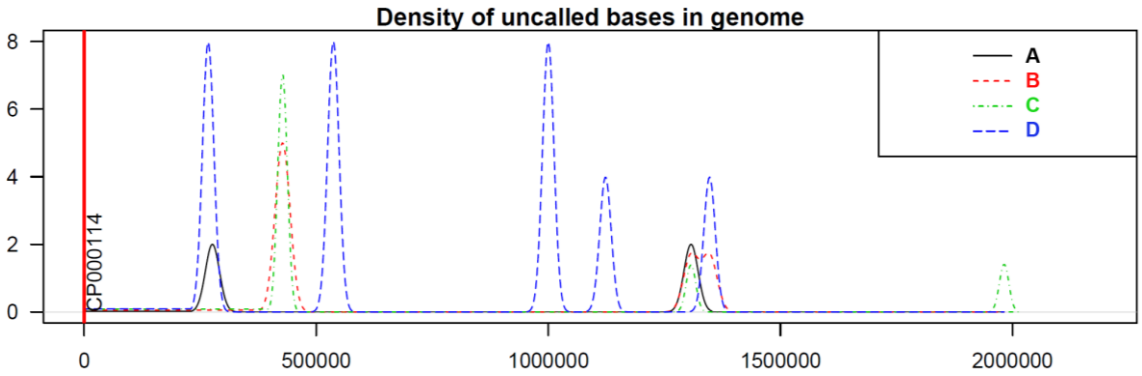

Figure 4.1. Density of uncalled bases (bases whose value is unknown) in the different evaluated assemblies with respect to the reference sequence. The x-axis represents the position in genome, and the y-axis, the number of uncalled bases present at that position.

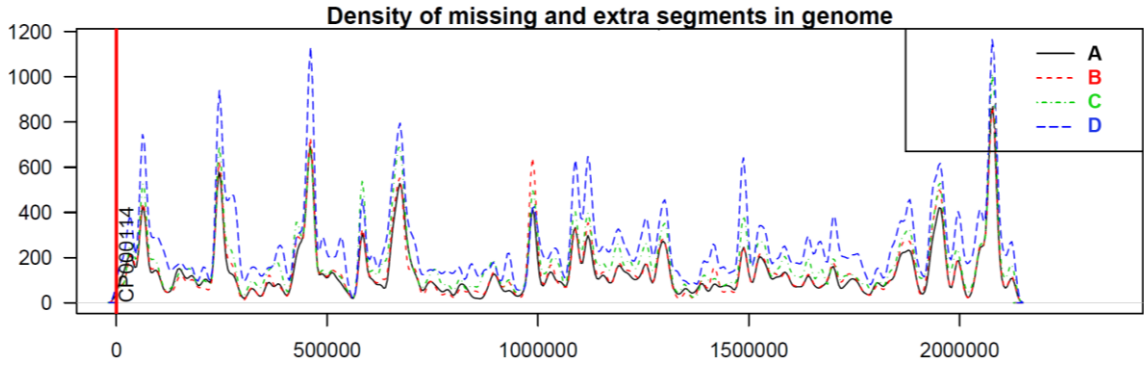

Figure 4.2. Density of missing and extra segments (of any length) in the different evaluated assemblies with respect to the reference sequence. The x-axis represents the position in genome, and the y-axis, the number of missing and extra segments present at that position.

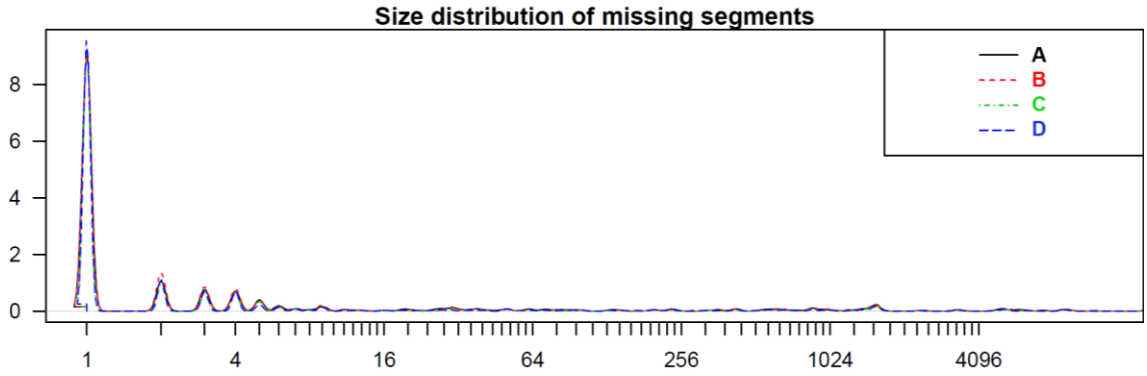

Figure 4.3. Size distribution of missing segments in the different evaluated assemblies with respect to the reference sequence. The x-axis represents the segment size, and the y-axis, the number of missing segments of that length.

Results from the use of Mauve Assembly Metrics (which extend beyond the ones displayed here) show that none of the assemblies surpasses the default assembly of MIRA in quality.

From this study, it has been discovered that **read preprocessing has negative effects on the assembly** with MIRA (whereas in DBG assemblers it can slightly increase results quality). It should be noted that most preprocessing applications are, just as most assemblers, designed to work with Illumina reads, so the sliding window trimming results are uncertain. For this reason, a fixed length trimming, determined from a previous quality analysis, was preferred.

Nevertheless, this situation is to be expected; there exist evidences that input data preprocessing harms the metrics of *de novo* assembly results, because it considerably constrains the options for constructing the assembly graph [58]. Even so, this technique remains useful for other types of assembly (such as EST assembly), or in order to decrease the time and memory consumption of the process. On the other hand, MIRA user manual [59] warns not to externally trim reads unless non-standard adapters are used, because the assembler itself is far more effective in this task, clipping the read ends around what is called the high confidence region (HCR).

4.3 Preprocessing: Internal quality control of MIRA

4.3.1 Quality control parameters

After verifying the ineffectiveness of preprocessing by external tools, the next step is to evaluate the **internal preprocessing and quality control parameters** of the MIRA assembler. In addition to the read preprocessing options enabled by default (which were considered necessary to maintain), MIRA offers the option of performing a quality control (QC) by means of a **sliding window**, which shifts across the read from the 5' to the 3' end, trimming it at the moment when the quality of the bases contained in the window falls below an specified threshold. If this option is enabled, the parameters that set both the window length and minimum quality threshold can be adjusted.

4.3.2 Results and discussion

In the same way as observed in the study of trimming tools, enabling the sliding window quality control option of MIRA always produces results of lower quality than if it is disabled, independently of the window length and the quality threshold adjustment. Table 4.2 presents the results corresponding to the configurations detailed below.

A) Default configuration (no QC).
B) QC: window length of 10 bp, minimum quality of 20 (Phred+33 scale).
C) QC: window length of 20 bp, minimum quality of 20.
D) QC: window length of 30 bp, minimum quality of 20 (default values).
E) QC: window length of 50 bp, minimum quality of 20.
F) QC: window length of 70 bp, minimum quality of 20.
G) QC: window length of 30 bp, minimum quality of 15.
H) QC: window length of 30 bp, minimum quality of 25.
I) QC: window length of 30 bp, minimum quality of 30.
J) QC: window length of 20 bp, minimum quality of 15 (best observed combination).

	Contigs ≥ 500 bp	Contigs ≥ N50	Min	N80	N50	N20	Max	Sum
A	36	4	559	71366	229335	526123	526123	2189373
B	38	5	532	54446	150964	277031	345587	2187243
C	42	4	543	54359	163912	568588	568588	2191231
D	43	5	555	51615	92435	243952	419009	2180693
E	57	8	509	37260	73790	179454	272712	2183943
F	76	13	521	29678	52633	85382	197354	2174318
G	40	4	595	65072	173837	559782	559782	2191906
H	58	7	519	35445	84655	195517	344171	2177004
I	161	23	508	12126	28382	61363	78919	2170248
J	47	6	508	50180	104733	230151	240372	2184883

Table 4.2. Results from the use of different MIRA quality control option values. The first two columns represent the number of contigs, and the other ones, total lengths expressed in bp.

Results demonstrate that **the application of quality control on reads has negative effects on the assembly**, just as the application of external preprocessing tools does. This is probably caused by the restriction that this quality control implies on the possibilities of the assembler during the overlap graph construction.

4.4 Result postprocessing

After analyzing the internal and external preprocessing options, the addition of **assembly postprocessing** (or finishing) tools remains to be studied. These tools may be classified into three types: scaffolding programs that try to generate groups of contigs (scaffolds) in order to increase the accuracy of the assembly; correction programs that repair errors in the contigs by comparisons with a reference sequence or with a set of paired-end reads; and programs focused on the detection and correction of redundant sequences that can be an effect of assembler errors. It should be pointed out that MIRA already implements its own phase directed to detect repetitive sequence in contigs.

According to the type of data the platform has to deal with (Ion Torrent single-end long reads, mainly from microbial genomes), it is not possible to carry out a scaffolding process because of the absence of mate-pair reads (see section 2.4, *Genome finishing techniques*). The correction of discrepancies between contigs and the reference genome has been indicated as undesirable by IUETSPC's researchers, because the regions where the assembly differs from the reference genome are the most relevant ones in terms of genome analysis. Therefore, the option of trying to **eliminate redundant sequences** from the assembly has been more deeply explored, choosing **Simplifier** [40] as a first tool for its study.

This correction tool has only one parameter (denoted as *c*), which sets how many bases at the ends can be disregarded for the identification of redundant sequences. The limitation of Simplifier is that it is only capable of detecting entire contigs as redundant and suppressing them, but it does not detect repetitive sequences inside the contigs themselves. This would suggest that it is directed to assemblies with shorter and more numerous contigs, where repeated contigs may be more frequent. As a result of applying this postprocessing method, some of the shortest contigs in the assembly are eliminated as the value of *c* is increased, so the different assembly statistics are barely affected.

Results from the application of Simplifier are unsatisfactory on this kind of assembly, especially if they are compared with its large execution time. On the other hand, this experiment led to a discussion about the possibility of properly evaluating the supposed improvement obtained by the application of a repeat correction method. In general, **repeat correction of the results has been considered unreliable**, given that it has not been possible to find a way to determine if the corrections made actually improve the assembly. Even if other options to perform this process had been tried out, it should be noted that repetitive sequences can be caused by errors of the assembler, but they are also very common in microbial genomes. Furthermore, MIRA already includes its own repeat detection stage (although its effectiveness is not determinable). Due to all this, assembly postprocessing has been left in the user's hands, dismissing its inclusion in favour of the integration of analysis tools, which are more necessary.

4.5 Integration of genome analysis tools

With the purpose of completing the platform functionality, a series of analysis tools has been integrated into the workflow, in order to obtain information relating to the classification and annotation of the assembly, the correct ordering and orientation of contigs with respect to the reference genome, the quality score distribution of reads and contigs, as well as different types of genome coverage and alignment graphs. Some of these tools produce results that are only valid for prokaryotic genomes, whereas other ones are applied only in cases in which a reference genome has been supplied. All these programs are reviewed below.

4.5.1 Quality analysis of reads and contigs

First of all, a **quality analysis** of both the supplied set of reads and the generated set of contigs is offered to the user. This data is merely informative and does not take part in any decision relating to the overall process.

The software **FastQC** [30] is used to generate these reports. The aim of FastQC is to provide a simple way to do some quality control checks on raw sequence data coming from high throughput sequencing pipelines. It provides a modular set of analyses that the user can use to give a quick impression of whether the data has any quality problems. The reports are offered to the user as HTML documents, which contain different graphs and tables relating to the quality analyses, such as those depicted by Figures 4.4 and 4.5.

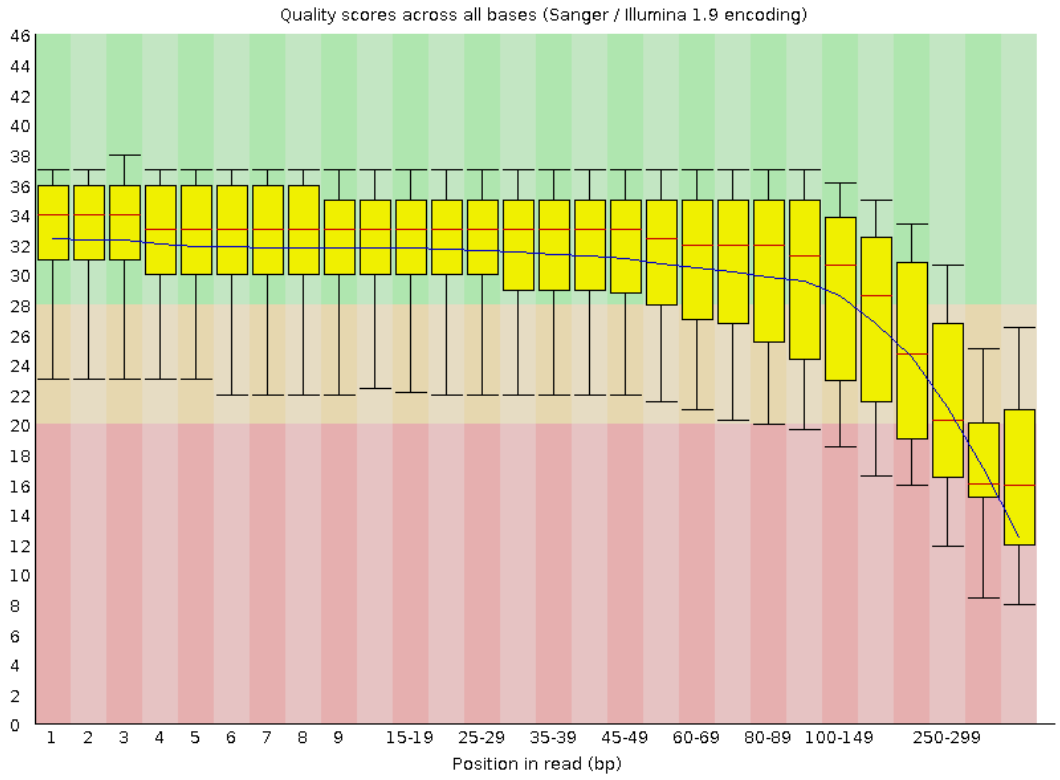

Figure 4.4. Diagram generated by FastQC, which shows an overview of the range of quality score values (Phred+33 scale) across all bases at each position in the *S. agalactiae* reads file. The central red line is the median value, the yellow box represents the inter-quartile range (25–75%), the upper and lower whiskers are the 10% and 90% points, and the blue line represents the mean quality. The y-axis shows the quality scores; the higher the score, the better the base call. The quality of calls on most platforms will degrade as the run progresses, so it is common to see base calls falling into the low-quality area towards the end of a read.

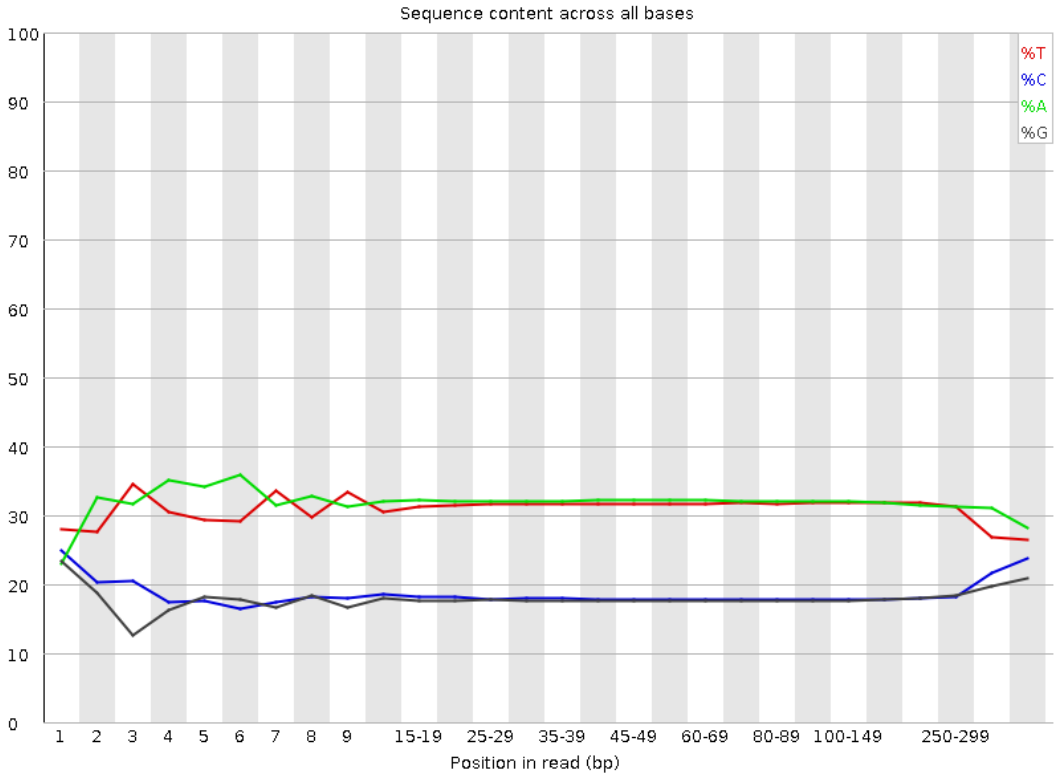

Figure 4.5. Diagram generated by FastQC, which plots out the proportion of each base position in the *S. agalactiae* reads file for which each of the four DNA bases has been called. In a random data set, it would be expected to be little to no difference between the different bases of a sequence run, so the lines in this plot should run parallel with each other. However, in this case there are irregularities derived from the presence of sequencing adapters (left end), as well as regions where quality scores decrease, so they are less accurate (right end).

4.5.2 Taxonomic classification

Another implemented functionality is the **taxonomic classification** of the assembly, that is carried out by a search of the sequence of the **16S ribosomal RNA gene** inside the contigs. This gene allows to rapidly identify a high number of microorganism species, which makes it useful for giving the user an idea of the content of the assembled contigs, as well as for detecting contamination by the presence of DNA from unwanted organisms.

In order to do this, the **BLAST** [60] tool, which finds regions of local similarity between genome sequences, is employed. BLAST allows to look for sequences in the contigs that match any of the 16S rRNA gene sequences of numerous microorganisms that are stored in its own database. The results are offered to the user as a tabular file, which includes the most significant three matches in each contig, using a matching threshold of 97% identity between the sequence from the contig and the one from the database (given that a lower value does not imply a match in terms of species).

4.5.3 Genome annotation

Genome annotation is defined as the addition of genetic information to the assembled sequence, which requires an identification of the genes contained in it. Annotation is a fundamental process for conferring usefulness and sense to the data resulting from genome assembly.

In the developed pipeline, genome annotation is carried out using **Prokka** [61], a software tool for the rapid annotation of prokaryotic genomes. It produces annotated genome files in GFF3, GBK and SQN formats, which can be edited and ultimately submitted to genome databases such as Genbank, DDJB or ENA. It also provides files containing the sequence of both genes and proteins associated with the supplied genome.

4.5.4 Genome alignment and coverage analysis

If the sequence of a reference genome is available, another indispensable step is performing a **genome alignment** between the assembled contigs and this reference sequence. This process allows to place the contigs over the regions of the genome to which they correspond, to check if their orientation is the correct one, to know the percentage of the reference sequence covered by the contigs, to study regions wherein both sequences differ, etc.

There exist a multitude of tools for carrying out the alignment between nucleotide sequences and generating text and graphic files relating to this process, which are very useful for the study of assembled genomes. The following sequence alignment utilities take part in the designed workflow: **MUMmer** [62], a software tool for rapidly aligning entire genomes, whether in complete or draft form, which also provides graphical output; **Mauve** [38], a system for efficiently constructing multiple genome alignments in the presence of large-scale evolutionary events such as rearrangement and inversion; **Circos** [63], a software package for visualizing data and information in a circular layout, which makes it ideal for exploring relationships between objects or positions; **Circoletto** [64], an online or stand-alone visualization tool based on Circos, which provides a fast, aesthetically pleasing and informative overview of sequence similarity search results; and **genoPlotR** [65], a R [66] package to produce reproducible, publication-grade graphics of gene and genome maps.

Alignment results are provided as tabular files, like the SNPs file generated by Mauve, and also in the form of graphs, which are contained in PNG of PDF files. Some of them, produced by MUMmer and Circoletto, are generated only if the alignment (and therefore, the genomes involved) is small enough to permit manageable graphic files. Figures 4.6–4.10 present some of the most representative diagrams generated by these genome analysis systems.

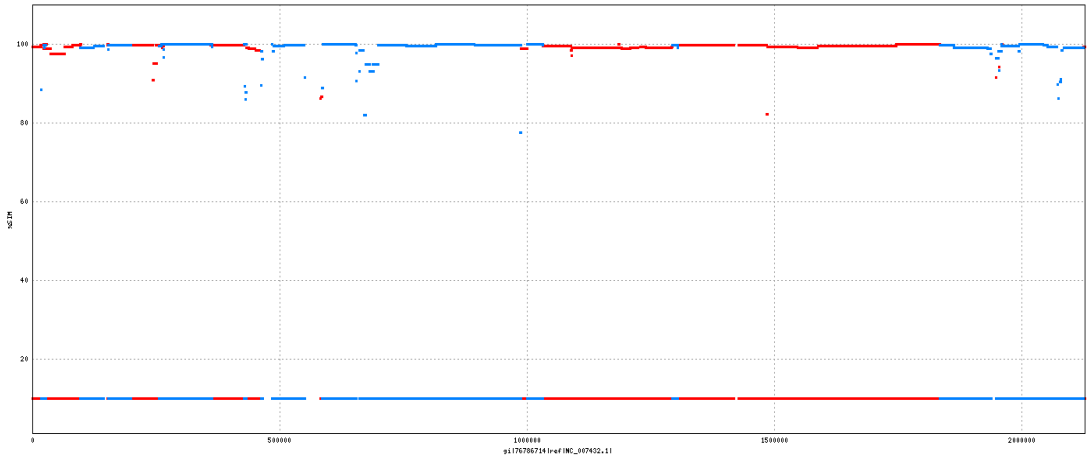

Figure 4.6. Linear diagram of the coverage of the alignment between contigs and the *S. agalactiae* A909 reference genome, generated by MUMmer. The x-axis represents the position in the reference sequence, and the y-axis, the identity percentage of the mappings at that position. The red segments indicate that the depicted contigs are correctly oriented, while the blue ones represent contigs whose orientation is inverted with respect to the reference sequence. The lower line allows to visualize the portion of the reference genome covered by the contigs.

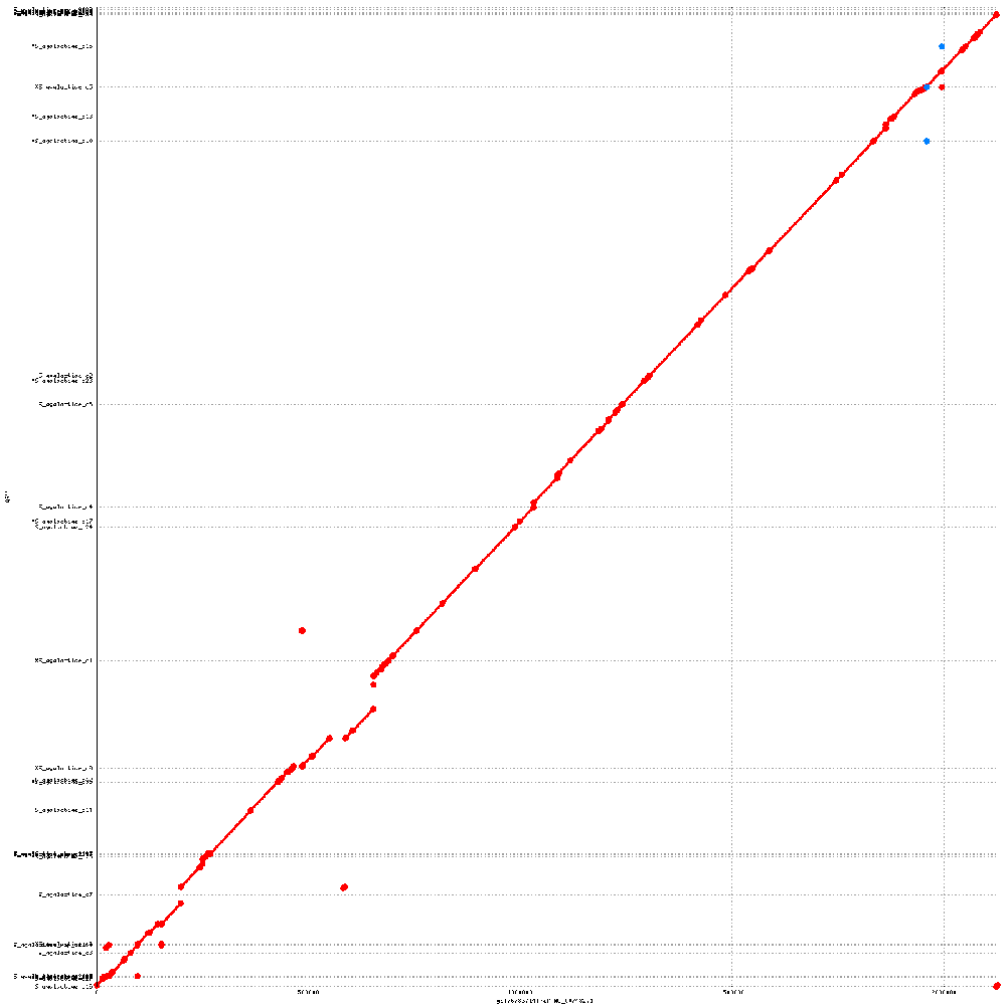

Figure 4.7. Transversal diagram of the alignment between contigs and the *S. agalactiae* A909 reference genome, generated by MUMmer. The x-axis represents the position in the reference sequence, and the y-axis, the position in the different contigs (from bottom to top). The diagonal line shows the mapping of contigs to the reference sequence (contigs are ordered and orientated in order to achieve the best possible alignment).

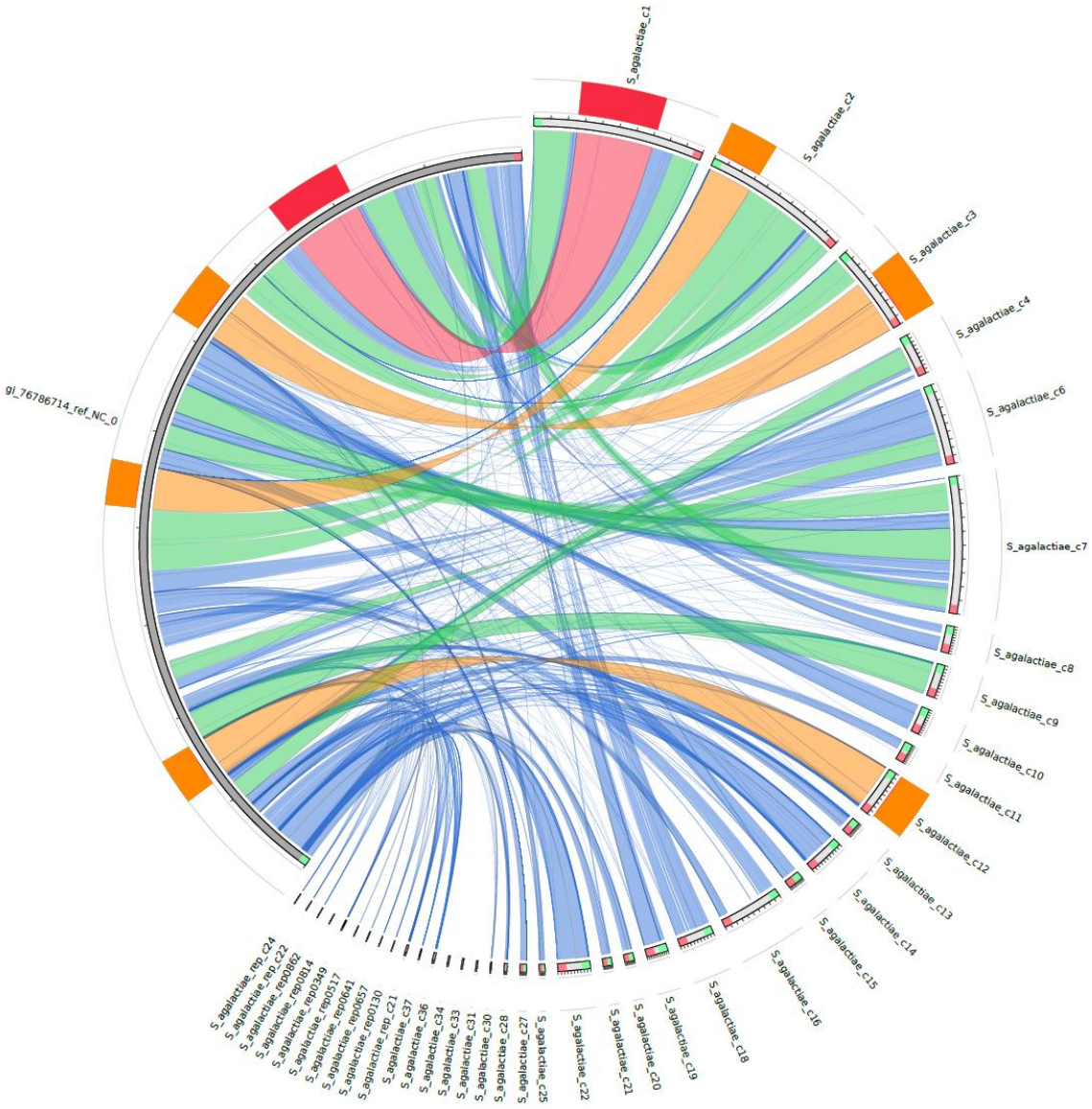

Figure 4.8. Circular diagram of the DNA alignment between contigs (right) and the *S. agalactiae* A909 reference genome (left), generated by Circos, BLAST and Circoletto. Matches are colored according to the identity percentage (from higher to lower: red, orange, green, blue), and have a fold in the middle if the contig is inverted with respect to the reference sequence.

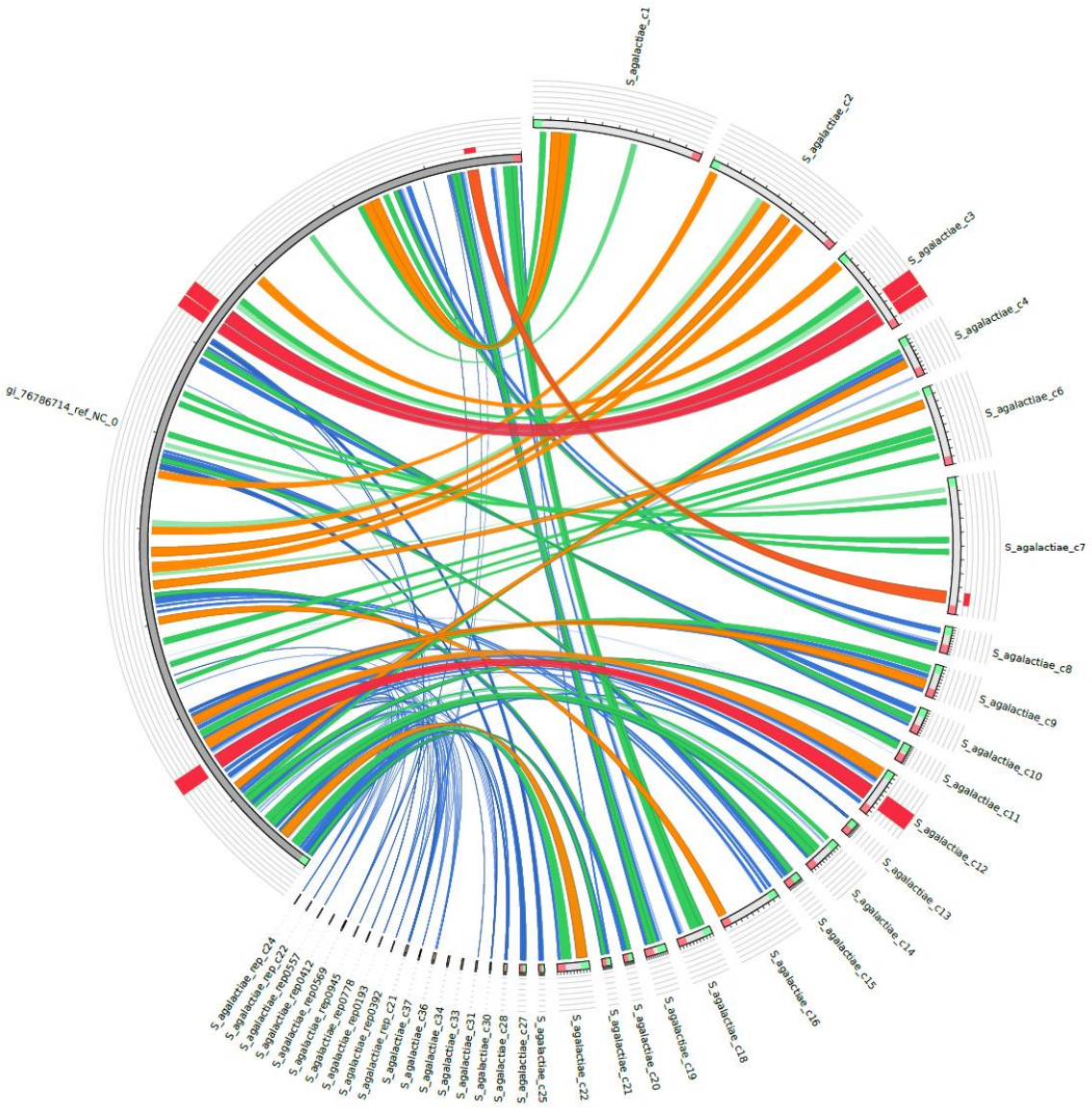

Figure 4.9. Circular diagram of the protein alignment between contigs (right) and the *S. agalactiae* A909 reference genome (left), generated by Circos, BLAST and Circoletto. Matches are colored according to the identity percentage (from higher to lower: red, orange, green, blue).

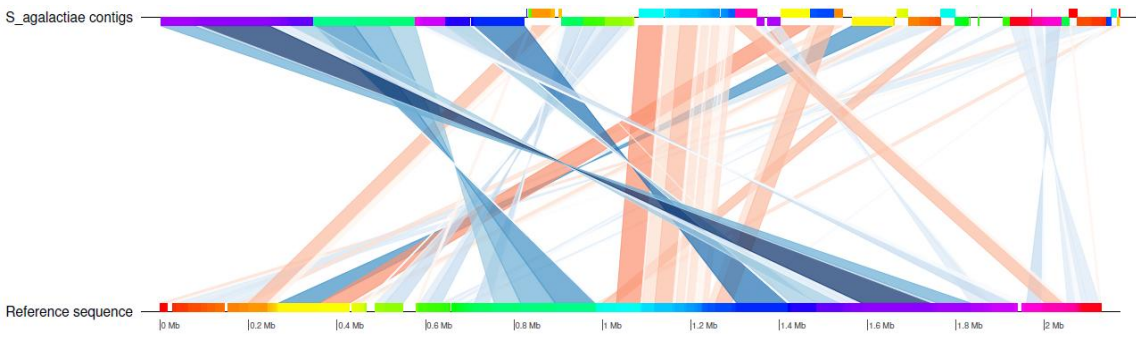

Figure 4.10. Linear diagram of the alignment between contigs (above) and the *S. agalactiae* A909 reference genome (below), generated by Mauve and genoPlotR. The matches are colored according to the identity percentage (the higher the darker) and the contig orientation (blue if it is inverted with respect to the reference sequence, and red otherwise).

4.5.5 Contig reordering

If a reference genome is supplied by the user, a complement of interest is the relative **reordering and reorientation of contigs** using the reference sequence as a basis. Facing the impossibility of grouping contigs into scaffolds, due to the absence of mate-pair reads (see section 4.4, *Result postprocessing*), this reordering process, performed by **Mauve**, is suitable for increasing the reliability of the assembly, by organizing the contigs so the union of its sequences is as similar as possible to the complete genome of the organism. This optional step does not eliminate the original contigs file resulting from the assembly. After reordering, the new contigs file is also annotated using **Prokka**, so the user is free to choose the more suitable contig set (original or reordered) for a particular end.

Chapter 5. IonGAP Web service

Abstract

- The transformation of the developed system into a Web platform makes it accessible to researchers from all over the world.
- The designed pipeline has been used as a base for a Web application developed with Django and deployed on an Apache server.
- The stages of genome assembly and analysis are managed by two independent modules that are periodically executed on the collection of existing assembly projects.
- The Web interface is aimed to offer an easy, likable use of the platform, and it has been developed using HTML, CSS and JavaScript.

5.1 Functioning and implementation of the Web service

The IonGAP system was initially conceived as an integrated tool for facilitating the assembly and study of bacterial genomes to the researches at the Applied Genomics Group of the IUETSPC. A consideration which has been taken into account since the start of the project is the publication of the set of tools and configurations chosen for the implementation of the internal pipeline of the platform, together with the series of comparative studies that have served as a base for the election of the former. All this information can be very useful in order to design new workflows for data coming from Ion Torrent sequencing platforms. This knowledge spreading intention has ended up motivating the transformation of the IonGAP platform into a Web service, so it can be freely accessed by researches from anywhere in the world.

The **Django** framework [67] has been utilized to develop the Web application, which has been subsequently deployed on an **Apache** Web server [68]. The Web service runs on a virtual machine equipped with a 64 bits dual core processor, 17 GB RAM and the Ubuntu 12.04 operating system, and hosted in a server kindly offered by the ETSII's **Computing Center**. The functioning scheme of the Web service, detailed in Figure 5.1, is based on a modular breakdown of the pipeline described in Chapter 4. The input data for each assembly project (project name, files and options) are provided by the user through a HTML Web form (see section 5.2, *User interface and usability*) and collected by the framework Django. This generates an object containing this data and stores it in its own database. The Web application then calls a **Bash** [69] shell script, which is responsible for creating the project base directory and the set of configuration files needed to carry out the different phases of the pipeline. Once the project preparation is complete, a status file is added into this directory, in order to indicate that it is time to proceed to genome assembly.

The assembly and analysis modules are managed by another two Bash shell scripts that are periodically run (each 20 minutes) by the operating system. The **assembly module** is responsible for searching projects which are ready to assemble, as well as performing genome assembly using MIRA together with the configuration file generated for that particular project. The time limit established for the assembly process is 36 hours, in order to avoid a possible system deadlock. Once the assembly is completed, if it finished without errors, another status

file is created to indicate that the project is ready for genome analysis. Otherwise, the project is labeled as failed.

The **analysis module**, in turn, regularly checks the existing projects looking for those which have been assembled but not analyzed, and then runs the different genome analysis routines, some of which depend on whether the sequence of a reference genome have been supplied (see section 4.5, *Integration of genome analysis tools*). If the project has not been correctly assembled, the script limits itself to perform a quality analysis of the reads file submitted by the user, in order to clarify the reasons of the failure. Independently of whether the project has been successful or not, results from the assembly and analysis phases are packaged into a ZIP file and, finally, the user is notified about the project result by an email, which includes a Web link through which the compressed results file can be downloaded. The platform has been made robust against power failures by the utilization of **status files**, which are used to indicate the progress of the project to the system modules, so the workflow can be resumed after a unexpected interruption. Status files are also useful for implementing **concurrency control** methods, which prevents two instances of a same module (assembly or analysis) from being simultaneously executed, due to the limited system resources.

5.2 User interface and usability

The main objective of the developed platform is to offer a balance between usefulness and usability. To this end, it is necessary to wrap the pipeline up in a **user interface** as friendly and light as possible. The Web site pages that make up the visible interface are just three: a start page, another that contains the Web form for the creation of a new assembly project, and a last one for informing the user that the project has been successfully created. These Web pages have been developed using free software tools and standards such as **HTML** [70], **CSS** [71] and **JavaScript** [72], and are displayed in Figures 5.2, 5.3 and 5.4.

The IonGAP Web service is currently operative and accessible through the provisional address **http://193.145.101.223**. To create a new assembly project using the graphical interface, it is just needed to supply a project name, an email address (to which the link to the results file will be sent), a reads file in FASTQ or BAM format and, optionally, a file in FASTA format containing the reference sequence. In order to provide the service with more flexibility and usefulness, users are allowed to configure all the parameters studied during the optimization of the MIRA assembler (although it is not recommended), so the assembler can adjust its configuration to the requirements of a particular data set whose properties are different from that of the data utilized in the platform development (as long as it is composed of Ion Torrent single-end reads). In addition, users are able to choose between 11 different output file formats for the assembly, therefore they only receive the files they need. Currently, a user manual that explains the usage of both the Web service and a set of useful external genome analysis and visualization tools, is being written.

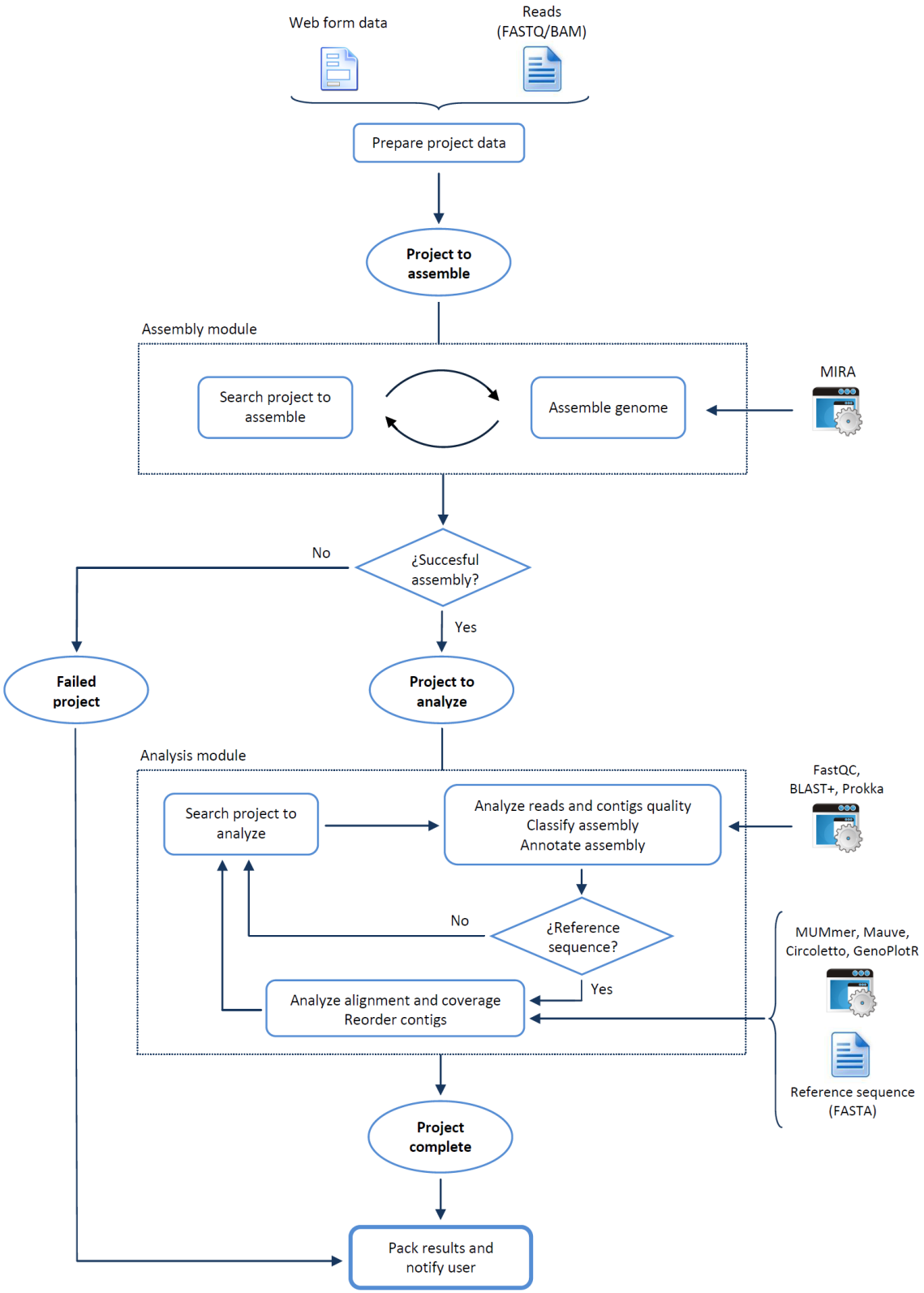

Figure 5.1. Internal functioning diagram of the Web service.

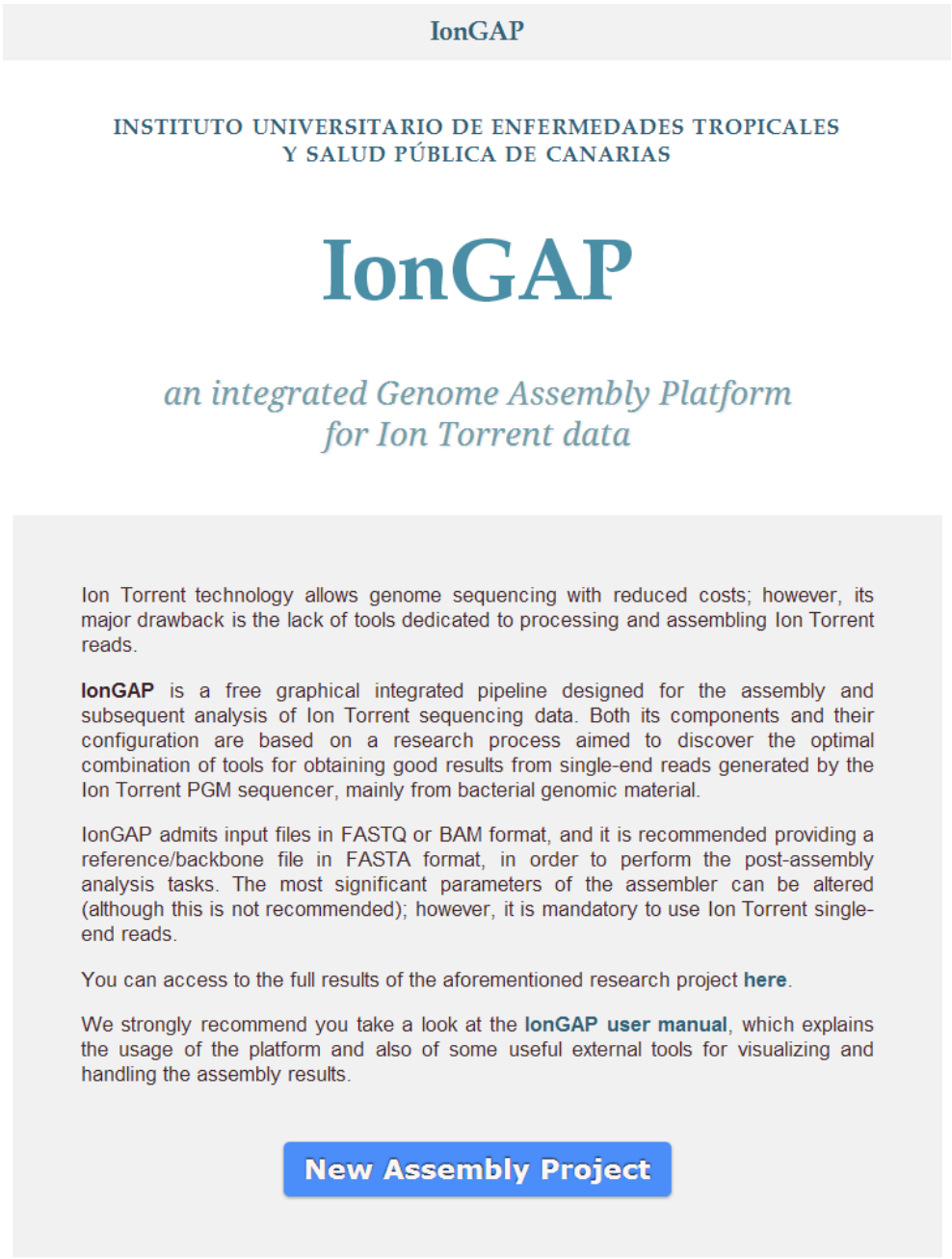

Figure 5.2. Start page of the Web service.

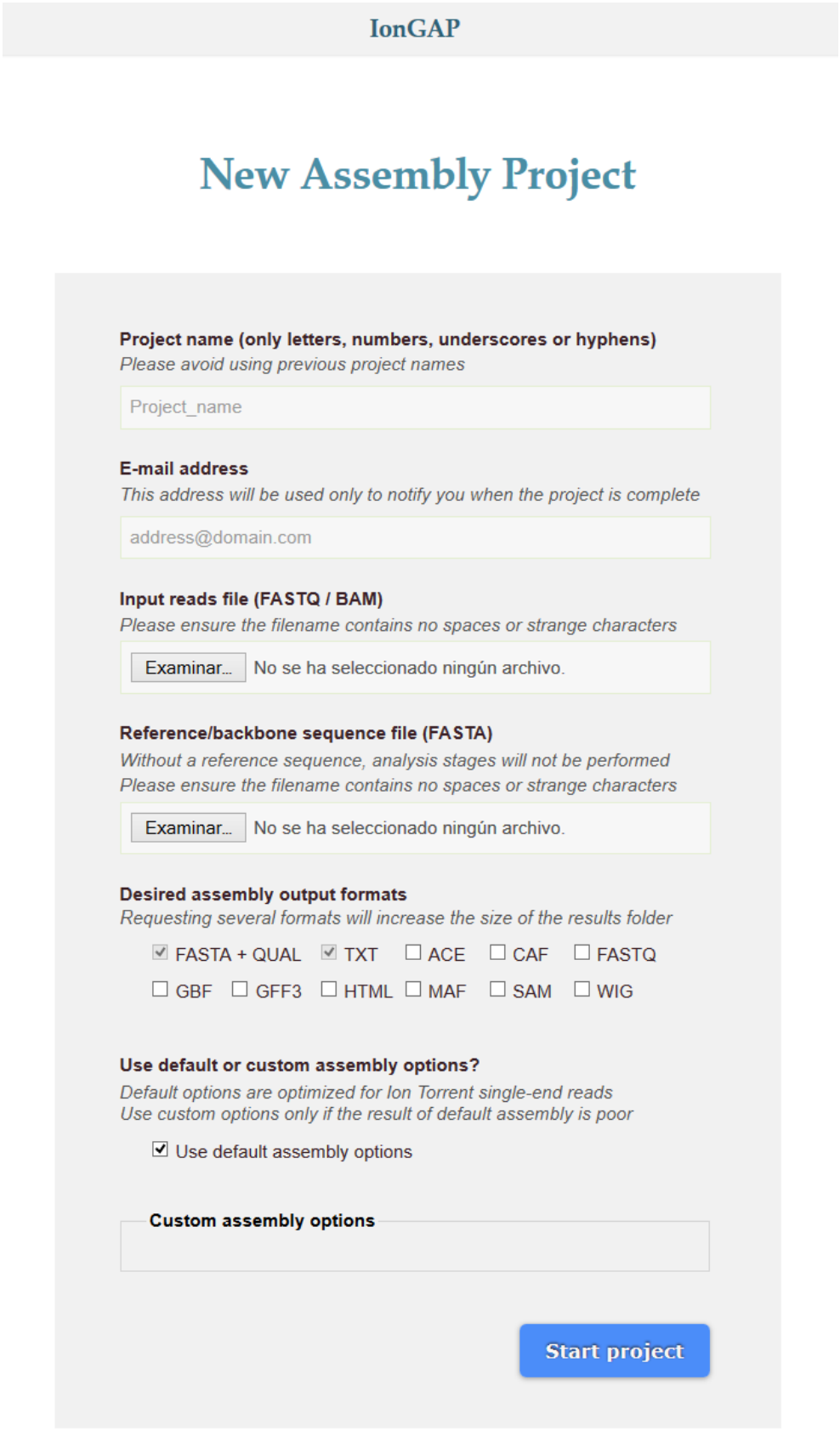

Figure 5.3. Web form for the creation of a new assembly project.

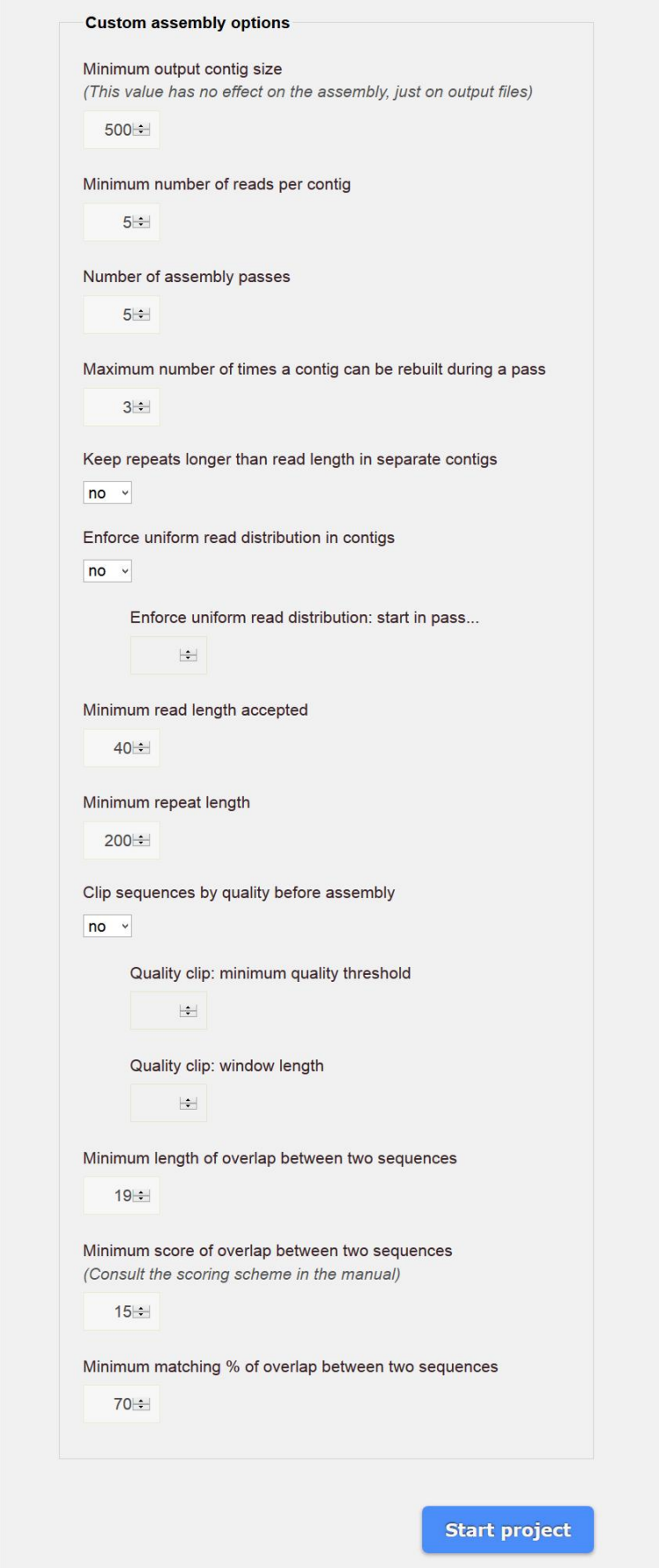

Figure 5.4. "Custom assembly options" expandable section of the New Assembly Project form (see Figure 5.3).

5.3 Genome assembly with IonGAP

The pipeline of bioinformatics tools defined for the IonGAP system has been tested by the **assembly of genomes of different organisms**, besides the *S. agalactiae* bacterial isolate. All of them use reads sequenced with the Ion Torrent PGM platform owned by the IUETSPC, and have been assembled on the server provided by the ETSII's Computing Center. The results obtained from each one of the assemblies are detailed in Tables 5.1 and 5.2.

Organism	Contigs ≥ 500 bp	Contigs ≥ N50	Min	N80	N50	N20	Max	Sum
Escherichia coli	2767	725	500	947	1684	2982	10308	3892829
Streptococcus agalactiae	36	4	559	71366	229335	526123	526123	2189373
Columba junoniae	2477	867	452	550	723	1167	4712	1853234
Trypanosoma cruzi	31174	5987	497	716	1504	4434	22689	39.30E6

Table 5.1. Results from the assembled genomes with respect to their quality. The second and third columns represent the number of contigs, and the next ones, total lengths expressed in bp.

Organism	Data size	RAM consumption	Running time
Escherichia coli	82.4 MB	1.64 GB	90 minutes
Streptococcus agalactiae	259.0 MB	4.07 GB	3 hours 45 minutes
Columba junoniae	1.0 GB	8.38 GB	3 hours 40 minutes
Trypanosoma cruzi	1.0 GB	13.75 GB	27 hours 40 minutes

Table 5.2. Results from the assembled genomes with respect to computational resources consumption.

It should be noted that the quality of the assemblies of *E. coli* and *C. junoniae* is in part the result of the particularities of the sequencing process. In both cases, the quality distribution of the reads is considerably inferior than that of the *S. agalactiae* reads; besides, in the case of *C. junoniae* the absence of a reference genome (since there is not any), together with the large size of the genome, has probably originated a set of reads that has a low genome coverage, because of the fact that they are scattered over a too large total genome, instead of concentrated on a specific area of it. The *T. cruzi* assembly, on the other hand, has been particularly hard because the genome of this organism has approximately 50% of repetitive sequences, which turns the assembly process extremely complex and implies the requirement of a large amount of computational resources, to such an extent that it has been necessary to reduce the amount of input data in order to perform the assembly with the available resources.

Chapter 6. Parallel assembly of large genomes

Abstract

- The computational complexity of the assembly of large genomes compels to abandon sequential genome assembly approaches in favor of parallel computing techniques.
- Contrail is a parallel genome assembler that permits *de novo* assembly of large genomes on the Apache Hadoop framework, employing a computer cluster.
- Assembling the genome of *T. cruzi* with Contrail takes less time and allows to use the whole read set. However, the DBG approach of this assembler makes it difficult to obtain quality results from sets of long reads.

6.1 Limitations of sequential genome assembly

Genome assembly performed by one program on one computer is probably the simpler and more logical conception of this process, and has been implemented by a great number of genome assemblers. Nevertheless, assembly is far from being a simple problem: depending on the particularities of the type of reads employed and the structure of the studied genomes (presence of repeats, mobile genetic elements, highly ambiguous regions, etc.), the input data volume can rapidly expand in several orders of magnitude. This effect has a critical impact considering that the type of memory employed for assembly is RAM, which is very limited in the majority of present computers. This situation is aggravated by the rapid evolution of genome sequencing technologies, that allow to obtain a much higher quantity of raw data than that handled some years ago.

There exist four fundamental ways to face this ***data explosion*** phenomenon. The first and most evident one is to increase the capacity of the machine wherein assembly is performed. However, given that the assembly of large genomes can require several hundreds of gigabytes of RAM, this option is usually too expensive. The second solution is to reduce the input data; this measure was carried out in the case of the *Trypanosoma cruzi* assembly (see section 5.3, *Genome assembly with IonGAP*), discarding 45% of the reads so that the memory consumption do not exceed the capacity of the server. Even so, it is not reasonable not to use the full power of the genome sequencer by such a severe data filtering; besides, discarding part of the reads will probably have an impact on the accuracy of the assembly results. A third option is to employ the sort of programs known as memory-efficient genome assemblers, whose aim is precisely to reduce the memory consumption of the assembly process [50]. Unfortunately, this type of assemblers (some of which are Minia and SparseAssembler) follow the de Bruijn graph paradigm, that has proven itself not being suitable for the assembly of Ion Torrent single-end long reads (see section 3.2, *Comparative study of genome assemblers*). Finally, there is one last alternative to the increase in computer power: dividing the execution among a set of machines, performing a **parallel genome assembly**. This sophisticated technique, that has been tested and included as a brief addition to this project, will be explained below.

6.2 Parallel *de novo* assembly: Contrail

Parallel computing simultaneously uses multiple computers to solve a problem. This can be achieved by splitting the problem into independent parts, so each machine can execute its part of the algorithm at the same time as the others. Analogously, **parallel genome assembly** is based on the breakdown of the assembly process in independent tasks, which are concurrently performed by different computers. This type of computing requires the use of a **computer cluster**, a group of loosely coupled computers (connected by a network) that work in close collaboration, so in some respects they can be considered a single computer. The machines in a cluster do not have to be symmetrical, which makes the load balance between them difficult.

Currently, one of the few genome assemblers that are able to perform a parallel *de novo* assembly is **Contrail** [73], which permits the assembly of large genomes on a computer cluster by means of a de Bruijn graph, carrying out the series of stages outlined in Figure 6.1.

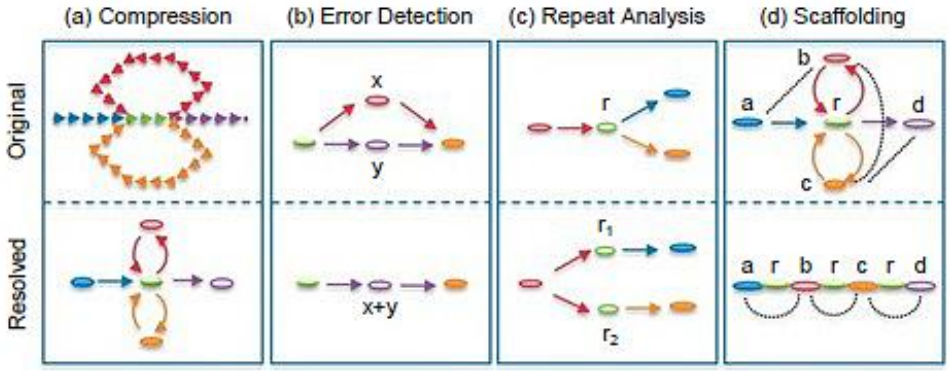

Figure 6.1. Outline of the processes that make up a typical Contrail execution [73].

Contrail runs on the **Apache Hadoop** framework [74], which supports distributed applications under a free license. Hadoop is inspired in the Google MapReduce algorithm (which is composed of the processes shown by the progress diagram in Figure 6.2) and the Google File System, and it allows the applications to work with hundreds or thousands of nodes (that can be ordinary low-cost computers) and petabytes of data. This parallel processing technology has been applied in order to try to solve the computational resource problem introduced by the *T. cruzi* assembly, thanks to the fact that the ETSII's Computing Center possesses its own Hadoop installation, which runs on a cluster of 108 computers. On the server that hosts the IonGAP service, the successful assembly of the genome of *T. cruzi* required discarding 45% of the reads so that the memory capacity of the machine (17 GB) was not exceeded.

6.2.1 Results and discussion

A comparison between the results from the assembly using Contrail with the whole input data, and the one employing MIRA with approximately 55% of the data reveals that, although parallel genome assembly may be a considerably powerful development line, the fact that Contrail uses a de Bruijn graph has (once more) repercussions on the assembly quality, given that **DGB approach is not suitable for the employed type of reads**. Contrail's own development team points out that they are working in a version of this program based on an overlap graph, which is "a strategy that will be especially valuable as read lengths increase to 100 bp and beyond"

[75]. Figure 6.3 and Tables 6.1 and 6.2 present the results obtained from the different assemblies carried out on the genomic data from *T. cruzi*, an organism whose genome has such a complexity level that a complete genome assembly may be very hard and expensive.

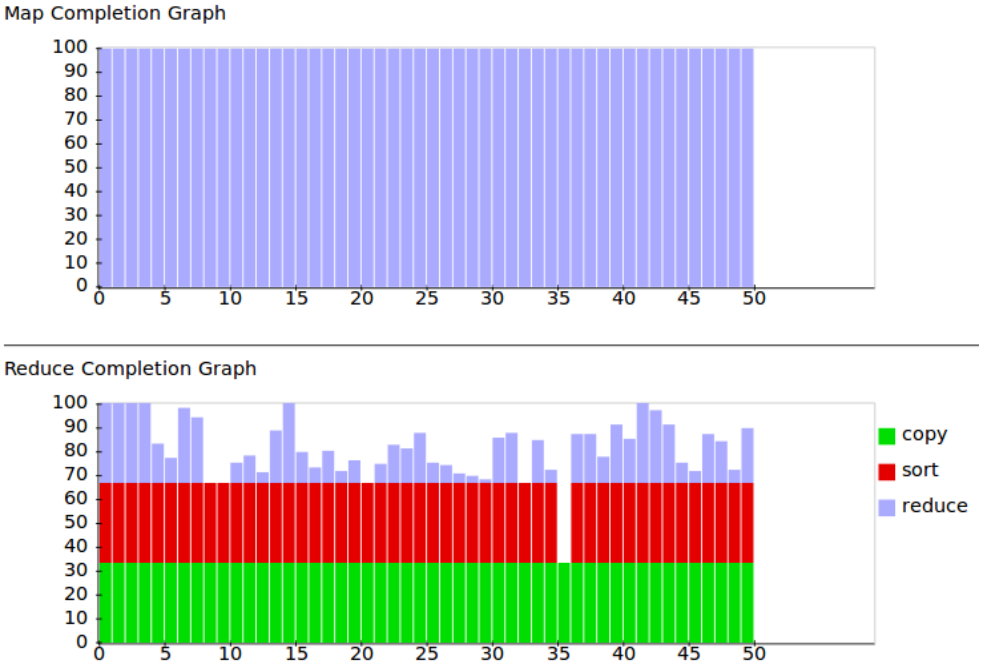

Figure 6.2. Informative diagram extracted from the user interface of the Apache Hadoop framework during the execution of one of the tasks belonging to a Contrail assembly. It shows the progress of each one of the consecutive stages (map, copy, sort and reduce) that make up the MapReduce execution scheme, for each involved node. The x-axis represents the number of the node, and the y-axis, the completeness percentage.

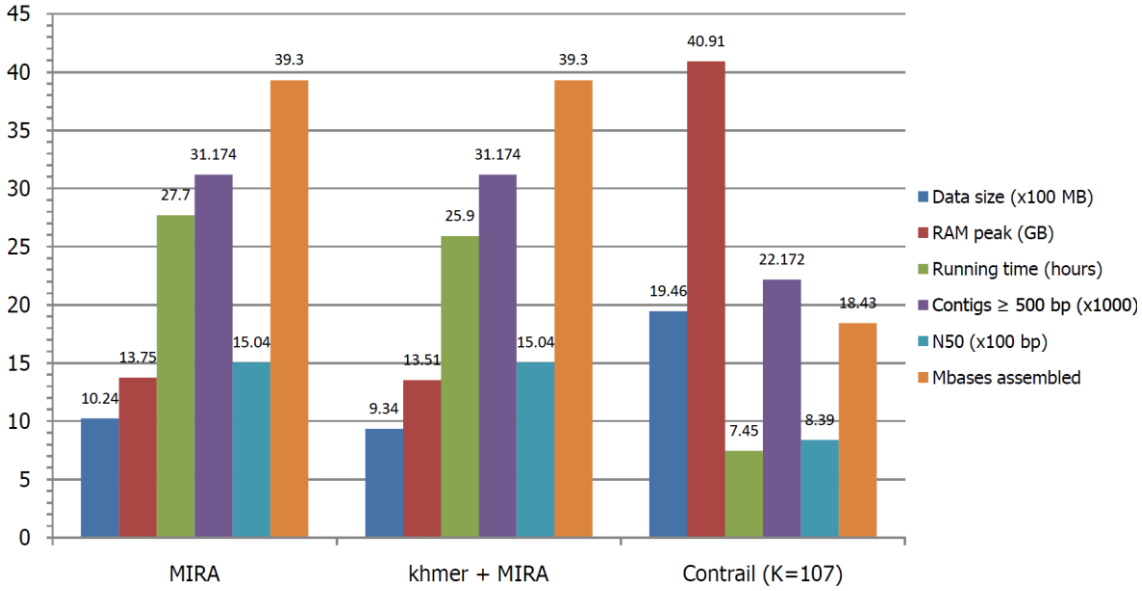

Figure 6.3. Results from the different *T. cruzi* assemblies with respect to their quality and computational resources consumption.

Tools	Contigs ≥ 500 bp	Contigs ≥ N50	Min	N80	N50	N20	Max	Sum
MIRA	31174	5987	497	716	1504	4434	22689	39.30E6
khmer + MIRA	31174	5987	497	716	1504	4434	22689	39.30E6
Contrail (K=107)	22172	7590	500	604	839	1307	4242	18.43E6

Table 6.1. Results from the different *T. cruzi* assemblies with respect to their quality. The second and third columns represent the number of contigs, and the next ones, total lengths expressed in bp.

Tools	Data size	RAM consumption	Running time
MIRA	1.0 GB	13.75 GB	27 hours 40 minutes
khmer + MIRA	934 MB	13.51 GB	25 hours 55 minutes
Contrail (K=107)	1.9 GB	40.91 GB	7 hours 25 minutes

Table 6.2. Results from the different *T. cruzi* assemblies with respect to computational resources consumption.

It has to be pointed out that Contrail is able to assemble the genome from the totality of the raw data and in a time period almost four times less than MIRA, but at the expense of consuming about three times more memory. However, parallel computing by means of a cluster can satisfy a such a large memory requirement, proving itself to be a very promising study avenue. The K-mer size used for the assembly with Contrail (K=107) has been determined as the optimal value by the utility **KmerGenie** [76], which performs tests on a read set employing different K values along a specified range. On the other hand, the second row in the tables corresponds to a data preprocessing stage that makes use of **khmer** [32], a program that implements the digital normalization technique (see section 2.3, *Data preprocessing techniques*) in order to eliminate overrepresented or supposedly erroneous reads from the data set, followed by a typical assembly with MIRA. It is interesting to observe that results from the two MIRA assemblies are identical with respect to the generated set of contigs. The fact that performing digital normalization produces the same result than just discarding the last million and a half reads from the data set implies that reads have a very uniform distribution over the genome, as well as good quality levels. If an assembly with MIRA using the totality of reads were possible, the result would probably improve very slightly or absolutely nothing. However, finding a **parallel overlap-layout-consensus assembler** (see section 2.2.2, *Overlap-layout-consensus assemblers*) would be desirable, so it were possible to effectively assemble genomes of this size and complexity, as well as more challenging ones.

Chapter 7. Conclusions

Abstract

- The IonGAP platform has solved the need for an automated genome assembly and analysis tool suffered by the IUETSPC.
- Making this platform available to the scientific community has as its purpose to stimulate low-cost genome research, as well as the development of other customized solutions that work with Ion Torrent data.
- The main considered improvements consist of the inclusion of new functionalities, the replacement of the sequential assembler in favor of a parallel one, the integration in the cloud with Amazon Web Services, or the assembly module migration to the Teide HPC supercomputer.
- The project is expected to have a noticeable impact on its field, considering that bioinformatics and biotechnology are two of the areas with greatest repercussions in the near future.

7.1 Project overview

The present work describes the creation of IonGAP, an open-access Web service for the processing, assembly and analysis of bacterial genomes, having as a base the genome sequencing data generated by the IUETSPC's Ion Torrent PGM platform. The optimization process of this service has been based on the sequencing reads originating from a genomic DNA sample of a strain of the Gram-positive bacterium *Streptococcus agalactiae*, isolated at the *Hospital Universitario Ramón y Cajal* (Madrid).

In order to build the assembly module of the system, a series of genome assemblers, based on both overlap-layout-consensus and de Bruijn graph strategies, was selected. Then, a study was carried out with the purpose of optimizing each assembler and comparing the results from all of them, using as a base aspects such as the number of assembled contigs and genome statistics like the N50 length. This study led to the conclusion that MIRA is, by far, the most suitable application for the genome assembly module, besides the fact that it is optimally configured by default.

The next step was the formation of a data processing pipeline around the assembler. Various data preprocessing options, both internal and external to the assembler, were tried out, as well as the postprocessing of the assembly results, but the obtained results had not a better quality that those derived exclusively from assembly. After this, a series of genome analysis tools was added in order to generate complementary textual and graphical reports that make the interpretation of results easier.

Once the platform workflow was defined, it was possible to proceed to integrate it inside a Web service, provided with a simple, friendly graphical interface that allows the rapid creation of genome assembly projects. This Web service has been implemented using the Django framework and the HTML, CSS and JavaScript standards, as well as Bash shell scripts. Thanks

to the server supplied by the ETSII's Computing Center, the IonGAP Web service is currently operative and openly accessible to any researcher, and it has been already tested by means of the assembly and analysis of genomes of various sizes.

As a last addition to the project development, the possibility of using a parallel genome assembler like Contrail has been studied, in order to make the assembly of large genomes feasible. The use of parallel computing has proven itself to be promising with respect to the permitted data volumes and running times; however, Contrail is not suitable for assembling the employed type of data, since it is based on a de Bruijn graph, which is not effective for long-read assembly.

7.2 General conclusions

The IonGAP integrated genome assembly platform has solved the need for an **automated tool for the assembly and preliminary analysis of Ion Torrent data** suffered by the *Instituto Universitario de Enfermedades Tropicales y Salud Pública de Canarias*. However, due to limitations of the different employed methods and tools, for the moment the platform is intended only for the assembly of small genomes belonging to prokaryotic organisms.

Beyond being a specific response to a local problem, making this platform available to the scientific community has as its purpose to **stimulate low-cost genome research**. The lack of tools dedicated to a minority sequencing technology like Ion Torrent is a hindrance to the investment in this type of equipment; nevertheless, the publication of the data and conclusions from the present study can be also useful to make the development of different customized solutions possible.

The specific case set out in this project, the determination of a *S. agalactiae* bacterial genome, has been successfully resolved; such is the case that a manuscript is been currently prepared for its publication in a high-impact scientific journal. The comparative genome analysis that have been carried out up to now reveal the high quality of the assembly obtained using the IonGAP platform. This conclusion is supported by the fact that the genomic regions which are exclusively present (~10%) in either the reference strain (A909) or in the studied strain contain mobile genetic elements such as insertion sequences (ISSag8, IS1381, IS10R) or integrated prophages (Lambda Sa03, Sa04 and Sa05).

7.3 Future work

The considered improvements and expansions of the platform are the inclusion of new options and features (for instance, antibiotic or virulence gene finding and multilocus sequence type definition), as regards to configuration possibilities and types of results and, above all, the integration of the platform in an interactive cloud environment in which users can, among other actions, browse their different assembly projects and obtain personalized reports and graphs in real time using their own Web browser. Considering the resource consumption and utilization frequency that are expected of the service, it might be possible to integrate it in the cloud with **Amazon Web Services** [77] at a very low cost.

Another especially desirable improvement consists of replacing the sequential genome assembler (MIRA) with a **parallel assembler that follows an OLC strategy** (**CloudBrush** [78] is a very interesting option to be studied), which would make the assembly of much larger and more complex genomes possible, employing parallel computing on a computer cluster. An alternative to parallelizing the problem is the migration of the assembly module of the system to the **Teide HPC** supercomputer, which is located in the ***Instituto Tecnológico y de Energías Renovables*** (ITER) [79] and it is currently the second most powerful computer in Spain, and the 169th most powerful in the whole world (as of June 2014 [80]). This possibility will be tackled in a **collaboration project** between the ITER, the IUETSPC and the *Hospital Universitario Nuestra Señora de Candelaria* (Tenerife), that will start up next September with the aim of improving the capacity of the Canary Islands in the field of genome analysis and research, offering the possibility of carrying out sequencing projects on large genomes (probably even on large regions of the human genome) by using the infrastructures and resources of these two specialized centers.

7.4 Impact of the project

There exist many possibilities that the project has a noticeable impact on its field, considering that **bioinformatics and biotechnology are currently two of the areas with greatest repercussions in biomedical research and clinical applications**. These sciences are expected to allow advances in the understanding and treatment of hereditary diseases that are associated with problems in the human genome, such as cancer, Alzheimer's disease, congenital immunodeficiency or cystic fibrosis, as well as the beginning of a new personalized medicine framework in which treatments will be adjusted to patients according to their genetic characteristics. Nowadays, events such as the election of the Barcelona Supercomputing Center as the new home of the European Genome-phenome Archive [81], which stores genome and phenome data on over 100000 people from all around the world, reinforce the position of Spain in the vanguard of research and innovation in biomedical computing.

On the other hand, another requirement that has been internationally pointed out as fundamental for the development of a **clinical molecular diagnostic based on bacterial whole-genome sequencing** is precisely the existence of a standardized workflow made up of a selection of available bioinformatic methods, programs and configurable options, and aimed to obtain an optimized efficiency for the employed applications while reducing configurable options and combining analysis steps into a **single automated pipeline** to increase both user-friendliness and potential for high-throughput application, as well as to reduce training requirements [3]. As of today, the IUETSPC possesses such automated pipeline, which is also openly available to the scientific community. This has been possible thanks to the efforts and collaboration between two groups of professionals from different areas, and to the development of a **multidisciplinary work strategy** that is becoming consolidated around the world as the way to tackle the new twenty-first-century science.

Capítulo 7. Conclusiones

Resumen

- La plataforma IonGAP ha resuelto la necesidad del IUETSPC de una herramienta automatizada para el ensamblado y análisis preliminar de genomas.
- La puesta a disposición de la comunidad científica de esta plataforma pretende incentivar la investigación genómica a bajo coste, así como el desarrollo de otras soluciones personalizadas que trabajen con datos procedentes de secuenciación por Ion Torrent.
- Las principales mejoras consideradas consisten en la inclusión de nuevas funcionalidades, la sustitución del ensamblador secuencial en favor de uno en paralelo, la integración en la nube con Amazon Web Services, o la migración del módulo de ensamblado al superordenador Teide HPC.
- Se espera que el proyecto tenga un impacto apreciable en su campo, teniendo en cuenta que la bioinformática y la biotecnología son dos de las áreas con mayor proyección en un futuro próximo.

7.1 Recapitulación del proyecto

El presente trabajo describe la creación de IonGAP, una plataforma web integrada de acceso libre para el procesado, ensamblado y análisis de genomas bacterianos, partiendo de datos obtenidos mediante secuenciación masiva en el equipo Ion Torrent PGM del IUETSPC. El proceso de optimización de dicha plataforma se ha basado en lecturas de secuenciación generadas a partir de una muestra de ADN genómico de una cepa de la bacteria Gram-positiva *Streptococcus agalactiae*, aislada en el Hospital Universitario Ramón y Cajal (Madrid).

Para constituir el módulo de ensamblado del sistema, fue seleccionada una serie de ensambladores genómicos, basados tanto en la aproximación *overlap-layout-consensus* como en la de grafos de de Bruijn. A continuación, se realizó un estudio para la optimización de cada ensamblador y la comparación entre sí de los resultados de todos ellos (basándose en aspectos como el número de contigs generados y estadísticas como el N50), el cual llevó a la conclusión de que MIRA es, con diferencia, la aplicación más adecuada para llevar a cabo el ensamblado genómico, encontrándose además configurada por defecto de manera óptima.

El siguiente paso fue la formación de un *pipeline* de tratamiento de los datos en torno al ensamblador. Se intentó preprocesar los datos de entrada externa e internamente al ensamblador, así como postprocesar los resultados de éste, pero no se obtuvieron resultados de mayor calidad que los derivados exclusivamente del ensamblado genómico. Tras esto, se añadió una sucesión de herramientas de análisis genómico para generar informes y gráficas complementarias que faciliten la interpretación de los resultados.

Una vez conformado el flujo de trabajo de la plataforma, se procedió a integrar la misma dentro de un servicio web, dotado de una interfaz gráfica simple y cómoda que permitiese la creación de proyectos de ensamblado rápidamente. Este servicio web ha sido implementado usando el framework Django y los estándares HTML, CSS y JavaScript, así

como scripts del intérprete Bash. Gracias al servidor proporcionado por el Centro de Cálculo de la ETSII, el servicio web IonGAP se encuentra actualmente operativo y libremente accesible por cualquier investigador, y ya ha sido puesto a prueba mediante el ensamblado y análisis de genomas de diferentes tamaños.

Como último añadido al desarrollo del proyecto, se ha estudiado la posibilidad de utilizar un ensamblador genómico en paralelo, Contrail, para llevar a cabo el ensamblado de grandes genomas. El uso de computación distribuida ha demostrado ser prometedor en cuanto a volúmenes de datos permitidos y tiempos de ejecución; sin embargo, Contrail no es adecuado para el ensamblado del tipo de datos empleado, dado que está basado en un grafo de de Bruijn, el cual no es efectivo para ensamblar lecturas largas.

7.2 Conclusiones generales

La plataforma integrada de ensamblado genómico IonGAP ha resuelto la necesidad de una **herramienta automatizada para el ensamblado y análisis preliminar de datos de Ion Torrent** por parte del Instituto Universitario de Enfermedades Tropicales y Salud Pública de Canarias. No obstante, debido a las limitaciones de los distintos métodos y herramientas empleadas, de momento la plataforma está pensada únicamente para ensamblado y análisis de genomas de pequeño tamaño pertenecientes a organismos procariotas.

Más allá de dar respuesta a un problema específico a nivel local, la puesta a disposición de la comunidad científica de esta plataforma gratuita pretende ser de ayuda para **incentivar la investigación genómica a bajo coste**. La falta de herramientas dedicadas a una tecnología de secuenciación minoritaria como es Ion Torrent constituye una traba a la inversión en este tipo de equipos; sin embargo, la publicación de los datos y conclusiones del presente estudio podrán servir también para posibilitar el desarrollo de diferentes soluciones personalizadas por parte de terceros grupos.

El caso concreto que se planteó en este trabajo, para la resolución de un genoma bacteriano de la especie *S. agalactiae*, se ha resuelto exitosamente; tal es así que en la actualidad se está preparando un manuscrito para su publicación en una revista científica de impacto. Los análisis de genómica comparativa realizados hasta el momento ponen de manifiesto la elevada calidad del ensamblado obtenido mediante la plataforma IonGAP. Esta conclusión queda avalada por el hecho de que las regiones genómicas que se encuentran exclusivamente (~10%), bien en la cepa de referencia (A909) o bien en la cepa de estudio, están representadas por elementos móviles tales como secuencias de inserción (ISSag8, IS1381, IS10R) o profagos integrados (Lambda Sa03, Sa04 y Sa05).

7.3 Trabajos futuros

Como **mejoras y ampliaciones a la plataforma**, se consideran la adición de nuevas opciones y características en cuanto a posibilidades de configuración y tipos de resultados y, sobre todo, la integración de la plataforma en un entorno interactivo en la nube en el que el usuario sea capaz, entre otras acciones, de navegar entre sus diferentes proyectos de ensamblado y obtener informes y gráficas personalizadas en tiempo real desde su propio navegador web. Dado el consumo de recursos del servicio y la frecuencia de utilización que se espera del

mismo, sería posible integrarlo en la nube empleando **Amazon Web Services** [77] con un coste muy bajo.

Otra mejora especialmente destacable consistiría en la sustitución del ensamblador secuencial (MIRA) por un ensamblador en paralelo que siga la aproximación OLC (una opción interesante podría ser **CloudBrush** [78]), el cual posibilitaría el ensamblado de genomas mucho mayores y más complejos, tanto procariotas como eucariotas, en un tiempo reducido empleando programación en paralelo sobre un *cluster* de ordenadores. Una alternativa a paralelizar el problema es el traslado del módulo de ensamblado del sistema al supercomputador **Teide HPC**, localizado en el **Instituto Tecnológico y de Energías Renovables** (ITER) [79] y que es actualmente el segundo ordenador más potente de España y el 169º de todo el mundo (a junio de 2014 [80]). Esta posibilidad será abordada mediante el emprendimiento de un **proyecto de colaboración** entre el ITER, el IUETSPC y el Hospital Universitario Nuestra Señora de Candelaria (Tenerife), que dará comienzo en septiembre del presente año con el objetivo de mejorar las capacidades de Canarias en materia de análisis y estudio genómico, ofreciendo la posibilidad de llevar a cabo proyectos de secuenciación de grandes genomas (posiblemente incluso de regiones amplias del genoma humano) usando los propios recursos e infraestructura de estos dos centros especializados.

7.4 Impacto del proyecto

Existen muchas posibilidades de que el proyecto tenga un impacto apreciable en su campo, teniendo en cuenta que **la bioinformática y la biotecnología son dos de las líneas con mayor proyección en el futuro más próximo**. Se espera que estas ciencias posibiliten avances en la comprensión y el tratamiento de enfermedades hereditarias asociadas a problemas del genoma humano, tales como el cáncer, la enfermedad de Alzheimer, la inmunodeficiencia congénita o la fibrosis quística, así como el inicio de un nuevo marco de medicina personalizada capaz de adaptar los tratamientos a cada paciente según sus características genéticas. Hoy en día, acontecimientos como la elección del Barcelona Supercomputing Center como nueva sede del Archivo Europeo del Genoma [79], el cual almacenará datos del genoma y fenoma de más de 100000 pacientes de todo el mundo, refuerzan la posición de España en la vanguardia en investigación e innovación en el campo de la computación biomédica.

Por otra parte, uno de los requisitos señalados a nivel internacional como fundamentales para el desarrollo de un **diagnóstico molecular clínico basado en secuenciación genómica** es precisamente la existencia de un flujo de trabajo estandarizado compuesto por una selección integrada de métodos, programas y configuraciones bioinformáticas, destinado a obtener una eficiencia optimizada de las aplicaciones específicas empleadas a la vez que se reducen las opciones configurables, a fin de combinar los pasos de ensamblado y análisis en un solo *pipeline* automatizado que resulte cómodo para el usuario y no requiera de aprendizaje para su manejo [3]. A día de hoy, el IUETSPC dispone de un *pipeline* de estas características, que se ofrece además, de forma libre, a la comunidad científica. Esto es posible gracias al esfuerzo y la colaboración entre dos grupos de profesionales de diferentes áreas, y al desarrollo de una **estrategia de trabajo multidisciplinar** que viene consolidándose en todo el mundo como la forma de abordar la nueva ciencia del siglo XXI.

References

[1] Lewin B. ***Genes IX***. *Jones & Bartlett Learning* (2007).

[2] Miller JR, Koren S, Sutton G. ***Assembly algorithms for next-generation sequencing data***. *Genomics* 15: 315–327 (2010).

[3] Fricke WF, Rasko DA. ***Bacterial genome sequencing in the clinic: bioinformatic challenges and solutions***. *Nature Reviews Genetics* 15: 49–55 (2014).

[4] ***Shinichi Morishita's academic Web page*** [online]. Graduate School of Frontier Sciences, University of Tokyo. Available at: http://www.k.u-tokyo.ac.jp/pros-e/person/shinichi_morishita/shinichi_morishita.htm

[5] ***Instituto Universitario de Enfermedades Tropicales y de Salud Pública de Canarias*** [online]. Universidad de La Laguna, Spain. Available at: http://www.ull.es/view/institutos/tropicales/Inicio/es

[6] ***Life Technologies*** [online]. Available at: http://www.lifetechnologies.com

[7] ***Escuela Técnica Superior de Ingeniería Informática*** [online]. Universidad de La Laguna, Spain. Available at: http://www.ull.es/view/centros/etsii/Inicio/es

[8] Liu L, Li Y, Li S, Hu N, He Y, Pong R, Lin D, Lu L, Law M. ***Comparison of next-generation sequencing systems***. *Journal of Biomedicine and Biotechnology*, 251364 (2012).

[9] ***Illumina – sequencing and array-based solutions for genetic research*** [online]. Available at: http://www.illumina.com

[10] ***Ion Torrent: amplicon sequencing*** [online]. Life Technologies (2011). Available at: http://www3.appliedbiosystems.com/cms/groups/applied_markets_marketing/documents/generaldocuments/cms_094273.pdf

[11] ***Pacific Biosciences*** [online]. Available at: http://www.pacificbiosciences.com

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

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

[14] Jeck WR, Reinhardt JA, Baltrus DA, Hickenbotham MT, Magrini V, Mardis ER, Dangl JL, Jones CD. ***Extending assembly of short DNA sequences to handle error***. *Bioinformatics* 23: 2942–2944 (2007).

[15] Myers EW, Sutton GG, Delcher AL, Dew IM, Fasulo DP, Flanigan MJ, Kravitz SA, Mobarry CM, Reinert KH, Remington KA, Anson EL, Bolanos RA, Chou HH, Jordan CM, Halpern AL, Lonardi S, Beasley EM, Brandon RC, Chen L, Dunn PJ, Lai Z, Liang Y, Nusskern DR, Zhan M, Zhang Q, Zheng X, Rubin GM, Adams MD, Venter JC. ***A whole-genome assembly of Drosophila***. *Science* 287: 2196–2204 (2000).

[16] Batzoglou S, Jaffe DB, Stanley K, Butler J, Gnerre S, *et al*. ***ARACHNE: a whole-genome shotgun assembler***. *Genome Research* 12: 177–189 (2002).

[17] Margulies M, Egholm M, Altman WE, Attiya S, Bader JS, et al. ***Genome sequencing in microfabricated high-density picolitre reactors***. *Nature* 437: 376–380 (2005).

[18] Hernandez D, François P, Farinelli L, Osterås M, Schrenzel J. ***De novo bacterial genome sequencing: millions of very short reads assembled on a desktop computer***. *Genome Research* 18: 802–809 (2008).

[19] Chevreux B, Wetter T, Suhai S. ***Genome sequence assembly using trace signals and additional sequence information***. *Computer Science and Biology: Proceedings of the German Conference on Bioinformatics (GCB)* 99: 45–56 (1999).

[20] ***PHAST: Phage Assembly Suite and Tutorial*** [online]. Genome Consortium for Active Teaching, Davidson College, North Carolina. Available at: http://gcat.davidson.edu/phast

[21] Pevzner PA, Tang H, Waterman MS. ***An Eulerian path approach to DNA fragments assembly***. *Proceedings of the National Academy of Sciences of the USA* 9: 9748–9753 (2001).

[22] Zerbino DR, Birney E. ***Velvet: Algorithms for de novo short read assembly using de Bruijn graphs***. *Genome Research* 18: 821–829 (2008).

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

[24] Butler J, MacCallum I, Kleber M, Shlyakhter IA, Belmonte MK, Lander ES, Nusbaum C, Jaffe DB. ***ALLPATHS: de novo assembly of whole-genome shotgun microreads***. *Genome Research* 5: 810–820 (2008).

[25] Li R, Zhu H, Ruan J, Qian W, Fang X, Shi Z, Li Y, Li S, Shan G, Kristiansen K, Li S, Yang H, Wang J, Wang J. ***De novo assembly of human genomes with massively parallel short read sequencing***. *Genome Research* 20: 265–272 (2010).

[26] Schmieder R, Edwards R. ***Quality control and preprocessing of metagenomic datasets***. *Bioinformatics* 27: 863–864 (2011).

[27] Kelley DR, Schatz MC, Salzberg SL. ***Quake: quality-aware detection and correction of sequencing errors***. *Genome Biology* 11: R116 (2010).

[28] Bolger AM, Lohse M, Usadel B. ***Trimmomatic: a flexible trimmer for Illumina sequence data***. *Bioinformatics*, in press (2014).

[29] Smeds L, Künstner A. ***ConDeTri – a content dependent read trimmer for Illumina data***. *PLoS ONE* 6: e26314 (2011).

[30] ***FastQC – a quality control tool for high throughput sequence data*** [online]. Babraham Bioinformatics, Babraham Institute, Cambridge (2010). Available at: http://www.bioinformatics.babraham.ac.uk/projects/fastqc

[31] Brown CT, Howe A, Zhang Q, Pyrkosz AB, Brom TH. ***A reference-free algorithm for computational normalization of shotgun sequencing data*** [online] (2012). Available at: http://arxiv.org/pdf/1203.4802v2.pdf

[32] ***khmer – K-mer counting & filtering FTW*** [online]. Available at: http://khmer.readthedocs.org

[33] Ronen R, Boucher C, Chitsaz H, Pevzner P. ***SEQuel: improving the accuracy of genome assemblies***. *Bioinformatics* 28: i188–i196 (2012).

[34] Koren S, Treangen TJ, Pop M. ***Bambus 2: scaffolding metagenomes***. *Bioinformatics* 27: 2964–2971 (2011).

[35] Gao S, Sung WK, Nagarajan N. ***Opera: reconstructing optimal genomic scaffolds with high-throughput paired-end sequences***. *Journal of Computational Biology* 18: 1681–1691 (2011).

[36] Boetzer M, Henkel CV, Jansen HJ, Butler D, Pirovano W. ***Scaffolding pre-assembled contigs using SSPACE***. *Bioinformatics* 27: 578–579 (2011).

[37] Assefa S, Keane TM, Otto TD, Newbold C, Berriman M. ***ABACAS: algorithm-based automatic contiguation of assembled sequences***. *Bioinformatics* 25: 1968–1969 (2009).

[38] Darling AC, Mau B, Blattner FR, Perna NT. ***Mauve: multiple alignment of conserved genomic sequence with rearrangements***. *Genome Research* 14: 1394–1403 (2004).

[39] Rissman AI, Mau B, Biehl BS, Darling AE, Glasner JD, Perna NT. ***Reordering contigs of draft genomes using the Mauve aligner***. Bioinformatics 25: 2071–2073 (2009).

[40] Ramos RT, Carneiro AR, Azevedo V, Schneider MP, Barh D, Silva A. ***Simplifier: a Web tool to eliminate redundant NGS contigs***. *Bioinformation* 8: 996–999, (2012).

[41] ***Bacterial genome sequence analysis*** [online]. BGI Tech Solutions Japan. Available at: http://www.bgisequence.com/jp/services/sequencing-services/microbial-sequencing/bacterial-sequencing

[42] ***Streptococcus*** [online]. Photograph by Martin Oeggerli, with support from School of Life Sciences, FHNW. Copyright National Geographic Society. All rights reserved (2013). Available at: http://ngm.nationalgeographic.com/wallpaper/img/2013/01/08-streptococcus_1600.jpg

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

[44] Boisvert S, Laviolette F, Corbeil J. ***Ray: simultaneous assembly of reads from a mix of high-throughput sequencing technologies***. *Journal of Computational Biology* 17: 1519–1533 (2010).

[45] Ye C, Ma ZS, Cannon CH, Pop M, Yu DW. ***Exploiting sparseness in de novo genome assembly***. *BMC Bioinformatics* 13: S1 (2012).

[46] Chikhi R, Rizk G. ***Space-efficient and exact de Bruijn graph representation based on a Bloom filter***. *Algorithms for Molecular Biology* 8: 22 (2013).

[47] Earl D, Bradnam K, St John J, Darling A, Lin D, Fass J, Yu HOK, Buffalo V, Zerbino DR, Diekhans M, Nguyen N, Ariyaratne PN, Sung W-K, Ning Z, Haimel M, Simpson JT, Fonseca NA, Birol I, Docking TR, Ho IY, Rokhsar DS, Chikhi R, Lavenier D, Chapuis G, Naquin D, Maillet N, Schatz MC, Kelley DR, Phillippy AM, Koren S, *et al*. ***Assemblathon***

1: a competitive assessment of de novo short read assembly methods. *Genome Research* 21: 2224–2241 (2011).

[48] Bradnam KR, Fass JN, Alexandrov A, Baranay P, Bechner M, Birol I, Boisvert S, Chapman JA, Chapuis G, Chikhi R, Chitsaz H, Chou W-C, Corbeil J, Del Fabbro C, Docking TR, Durbin R, Earl D, Emrich S, Fedotov P, Fonseca NA, Ganapathy G, Gibbs RA, Gnerre S, Godzaridis E, Goldstein S, Haimel M, Hall G, Haussler D, Hiatt JB, Ho IY, *et al*. ***Assemblathon 2: evaluating de novo methods of genome assembly in three vertebrate species***. *Gigascience* 2: 10 (2013).

[49] Salzberg SL, Puiu D, Magoc T, Koren S, Treangen TJ, Schatz MC, Delcher AL, Roberts M, Marçais G, Pop M, Yorke JA. ***GAGE: a critical evaluation of genome assemblies and assembly algorithms***. *Genome Research* 15: 557–567 (2012).

[50] Kleftogiannis D, Kalnis P, Bajic VB. ***Comparing memory-efficient genome assemblers on stand-alone and cloud infrastructures***. *PLoS ONE* 8: e75505 (2013).

[51] Smith TF, Waterman MS. ***Identification of common molecular subsequences***. *Journal of Molecular Biology* 147: 195–197 (1981).

[52] Chevreux B. ***MIRA: an automated genome and EST assembler*** [online]. Ph.D. thesis, Medical Faculty of Heidelberg, Ruprecht-Karls-University (2005). Available at: http://megasun.bch.umontreal.ca/cours/BIN-6002/Publications/MIRA_chevreux_thesis-2005.pdf

[53] Lander ES, Waterman MS. ***Genomic mapping by fingerprinting random clones: A mathematical analysis***. *Genomics* 2: 231-239 (1988).

[54] Chen C, Gertz J. ***Lander-Waterman sequencing statistics*** [online]. Department of Genetics, Washington University, St. Louis (2005). Available at: www.ucdenver.edu/academics/colleges/medicalschool/departments/biochemistry/GraduatePrograms/genomics/Documents/sequence_coverage_math.doc

[55] Tesler G. ***Lander-Waterman statistics for shotgun sequencing*** [online]. Department of Mathematics, University of California, San Diego (2013). Available at: http://www.math.ucsd.edu/~gptesler/186/slides/shotgun_f13-handout.pdf

[56] Del Fabbro C, Tardivo F, Policriti A. ***A parallel algorithm for the best k-mismatches alignment problem***. *22nd Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*: 586–589 (2014).

[57] Darling AE, Tritt A, Eisen JA, Facciotti MT. ***Mauve Assembly Metrics***. *Bioinformatics* 27: 2756–2757 (2011).

[58] Del Fabbro C, Scalabrin S, Morgante M, Giorgi FM. ***An extensive evaluation of read trimming effects on Illumina NGS data analysis***. *PLoS ONE* 8: e85024 (2013).

[59] Chevreux B. ***Sequence assembly with MIRA 4 – the definitive guide*** [online] (2013). Available at: http://mira-assembler.sourceforge.net/docs/DefinitiveGuideToMIRA.html

[60] Altschul S, Gish W, Miller W, Myers E, Lipman D. ***Basic local alignment search tool***. *Journal of Molecular Biology* 215: 403–410 (1990).

[61] Seemann T. ***Prokka: rapid prokaryotic genome annotation***. *Bioinformatics*, in press (2014).

[62] Kurtz S, Phillippy A, Delcher AL, Smoot M, Shumway M, Antonescu C, Salzberg SL. ***Versatile and open software for comparing large genomes***. *Genome Biology* 5: R12 (2004).

[63] Krzywinski M, Schein J, Birol I, Connors J, Gascoyne R, Horsman D, Jones SJ, Marra MA. ***Circos: an information aesthetic for comparative genomics***. *Genome Research* 19: 1639–1645 (2009).

[64] Darzentas N. ***Circoletto: visualizing sequence similarity with Circos***. *Bioinformatics* 26: 2620–2621 (2010).

[65] Guy L, Roat Kultima J, Andersson SGE. ***genoPlotR: comparative gene and genome visualization in R***. *Bioinformatics* 26: 2334–2335 (2010).

[66] ***The R Project for Statistical Computing*** [online]. Available at: http://www.r-project.org

[67] ***Django – the Web framework for perfectionists with deadlines*** [online]. Available at: http://www.djangoproject.com

[68] ***The Apache HTTP Server Project*** [online]. Available at: http://httpd.apache.org

[69] ***GNU Bash*** [online]. GNU Operating System. Available at: http://www.gnu.org/software/bash

[70] ***HTML*** [online]. W3C Web Education Community Group Wiki. Available at: http://www.w3.org/community/webed/wiki/HTML

[71] ***Cascading Style Sheets*** [online]. W3C Web Education Community Group Wiki. Available at: http://www.w3.org/Style/CSS

[72] ***Your first look at JavaScript***. W3C Web Education Community Group Wiki [online]. Available at: http://www.w3.org/wiki/Your_first_look_at_JavaScript

[73] ***Contrail*** [online]. SourceForge. Available at: http://sourceforge.net/apps/mediawiki/contrail-bio/index.php?title=Contrail

[74] ***Apache Hadoop*** [online]. The Apache Software Foundation. Available at: http://hadoop.apache.org

[75] Schatz MC, Sommer D, Kelley D, Pop M. ***De novo assembly of large genomes using cloud computing*** [online]. Center for Bioinformatics and Computational Biology, University of Maryland. Available at: http://www.cbcb.umd.edu/~mschatz/Posters/BOG.2010%20-%20Contrail.pdf

[76] Chikhi R, Medvedev P. ***Informed and automated K-mer size selection for genome assembly***. *Bioinformatics* 30: 31-37 (2014).

[77] ***Amazon Web Services (AWS)*** [online]. Available at: http://aws.amazon.com

[78] Chang YJ, Chen CC, Chen CL, Ho JM. ***A de novo next generation genomic sequence assembler based on string graph and MapReduce cloud computing framework***. *BMC Genomics* 13: S28 (2012).

[79] ***Instituto Tecnológico y de Energías Renovables S.A.*** [online]. Santa Cruz de Tenerife, España. Available at: http://www.iter.es

[80] ***TEIDE-HPC – Fujitsu PRIMERGY CX250 S1, Xeon E5-2670 8C 2.600GHz, Infiniband QDR*** [online]. TOP500 Supercomputer Sites. Available at: http://www.top500.org/system/178254

[81] ***Barcelona is the new home of the European Genome-phenome Archive, a fundamental resource for biomedical research*** [online]. Barcelona Supercomputing Center, Spain (2014). Available at: http://www.bsc.es/about-bsc/press/bsc-in-the-media/barcelona-new-home-european-genome-phenome-archive-fundamental

Appendices

A1 Comparative study of genome assemblers

The tables shown below present the results from the optimization of the different evaluated assemblers. In all the tables, the second, third and fourth columns represent the number of contigs, and the next ones, total lengths expressed in base pairs. The row corresponding to the optimal adjustment has been highlighted in bold type.

Table A1.1. Results from **ABySS** optimization by adjusting the *K* (K-mer length) parameter.

K	Contigs	Contigs ≥ 500 bp	Contigs ≥ N50	N80	N50	N20	Max	Sum
17	26333	635	257	545	665	835	1593	425827
18	12736	1553	509	674	996	1569	4268	1468225
19	8163	1319	353	993	1770	3068	7584	1924924
20	6747	962	225	1453	2938	4996	11943	2025602
21	5829	778	172	1882	3758	6841	16758	2065246
22	5492	699	155	2130	4168	7355	15682	2078301
23	5197	666	142	2269	4506	8275	19646	2085349
24	4942	657	138	2319	4710	8794	19644	2089228
25	4849	651	136	2313	4828	8941	19631	2089416
26	4639	642	138	2397	4875	8742	15886	2093516
27	4335	628	131	2416	4968	9034	20978	2094355
28	4390	626	133	2416	5085	8957	15888	2094228
29	4127	614	136	2524	5031	8740	15889	2095247
30	4026	611	133	2524	5086	8973	15890	2097016
31	3905	598	127	2542	5135	9210	28270	2096657
32	3867	601	126	2524	5095	9431	21703	2095983
33	**3654**	**583**	**121**	**2643**	**5448**	**9930**	**28270**	**2099228**
34	4292	625	129	2410	4853	9344	28270	2091295
35	4209	639	139	2428	4712	8662	19665	2093959
36	4163	635	136	2466	4838	8728	18827	2091269
37	4090	636	138	2465	4764	8670	28270	2092941
38	3989	646	138	2444	4753	8728	21197	2095929
39	3894	647	134	2406	4684	9321	24717	2097156
40	3867	654	136	2399	4696	8989	24717	2099055
41	3736	652	135	2410	4684	8957	28243	2098294
42	3666	649	137	2466	4650	8728	28981	2097847
43	3605	658	136	2371	4684	8728	26980	2100746
44	3556	661	140	2399	4650	8957	19812	2100998
45	3391	654	136	2407	4662	9194	22849	2101301

46	3411	660	140	2396	4635	8803	19812	2101180
47	3322	653	135	2407	4650	9103	19812	2101018
48	3319	652	139	2356	4684	9003	19592	2095402
49	3179	640	137	2479	4744	8518	22092	2095198
50	3445	673	143	2294	4516	8614	21369	2094159
51	3587	674	145	2294	4537	7932	19812	2090226
52	3558	681	148	2272	4326	7985	19812	2091063
53	3438	673	143	2276	4340	8707	21891	2088865
54	3320	670	141	2291	4451	8805	21302	2090132
55	3247	665	138	2276	4575	8806	21302	2090602
56	3292	672	140	2309	4516	8700	21302	2092331
57	3258	680	141	2227	4516	8461	19381	2093981
58	3140	673	141	2278	4583	8200	24114	2096463
59	3057	682	145	2266	4438	8345	18698	2097030
60	3575	714	152	2090	4255	7738	19382	2086482
61	3431	703	151	2196	4309	7738	19383	2088595
62	3337	681	146	2272	4503	8002	19383	2087965
63	3241	665	142	2324	4631	8200	19384	2089569
64	3318	654	137	2349	4675	8864	19062	2096234

Table A1.2. Results from **SparseAssembler** optimization by adjusting the *K* (K-mer length) parameter. SparseAssembler does not permit values of K higher than 63.

K	Contigs	Contigs ≥ 500 bp	Contigs ≥ N50	N80	N50	N20	Max	Sum
15	9189	1104	423	582	742	1000	2335	824514
19	1838	337	61	5456	10143	21868	46359	2125516
23	1535	220	36	8184	17274	34595	56437	2128706
27	1449	210	35	8067	19156	38526	56098	2128536
31	**1412**	**184**	**27**	**10962**	**26067**	**44142**	**59909**	**2130395**
35	1319	209	29	7640	22072	44108	69027	2132819
39	1300	232	36	7278	17501	34960	59310	2130713
43	1254	270	42	6196	15026	29625	59325	2130399
47	1189	283	44	5948	13473	30805	59049	2129647
51	1175	323	56	5342	11710	22969	37536	2124977
55	1156	381	70	4132	9009	18576	38410	2124185
59	1162	444	87	3710	7505	14979	29136	2118170
63	1187	506	98	3195	6559	12900	29608	2115686

Table A1.3. Results from **Ray** optimization by adjusting the *K* (K-mer length) parameter. A value higher than 49 produces a segmentation fault, in a system equipped with 8 GB RAM.

K	Contigs	Contigs ≥ 500 bp	Contigs ≥ N50	N80	N50	N20	Max	Sum
17	8131	1418	357	997	1835	3364	9679	2112894
19	7420	870	165	1818	4185	8104	21289	2308078
21	6798	298	32	9585	20605	41596	75290	2194340
23	6817	264	26	13264	26602	52141	75301	2215950
25	6864	252	25	13767	29265	54919	72970	2264387
27	6489	236	26	14908	28780	46669	77955	2207527
29	**6034**	**208**	**26**	**17171**	**30593**	**48921**	**77955**	**2261808**
31	5628	215	30	15411	27600	41585	66650	2319392
33	5167	219	28	13214	26601	47511	81263	2301460
35	4777	233	37	13130	22995	32029	72212	2417561
37	4354	241	40	11880	22289	31964	57505	2459567
39	4012	251	47	10624	18493	28506	48018	2525444
41	3606	249	49	9814	16877	29241	48020	2454409
43	3243	265	51	8573	16432	25607	43204	2430710
45	2902	285	56	7358	14485	23152	48618	2362286
47	2647	332	63	6105	12015	21357	27580	2323122
49	2402	396	75	4649	9233	19192	41883	2294159

Table A1.4. Results from **SGA** optimization by adjusting the *OL* (overlap length) parameter. The *MOL* (overlap length used in the FM-merge phase) parameter has been adjusted to (OL – 20).

OL	Contigs	Contigs ≥ 500 bp	Contigs ≥ N50	N80	N50	N20	Max	Sum
75	22760	1232	468	556	689	993	2297	880170
85	19229	1439	534	563	717	1057	2742	1061584
95	15939	1655	598	574	764	1124	2391	1266656
105	13055	1783	629	590	805	1228	3081	1425104
115	10720	1932	673	598	837	1273	3483	1581566
125	**9079**	**2020**	**695**	**601**	**843**	**1288**	**2983**	**1671949**
135	7866	2097	738	600	837	1220	2748	1710024
145	7141	2062	751	593	789	1134	2989	1620597
155	6622	1954	736	576	739	1044	2492	1462304
165	6284	1705	667	557	691	934	2692	1199149
175	6089	1274	522	545	644	829	2617	842496
185	5654	764	332	529	593	719	1525	466929
195	4875	333	149	519	561	670	965	194179
205	3820	85	39	513	557	623	756	48204
215	2586	17	8	509	538	631	664	9341

Table A1.5. Results from **Minia** optimization by adjusting the *K* (K-mer length) parameter.

K	Contigs	Contigs ≥ 500 bp	Contigs ≥ N50	N80	N50	N20	Max	Sum
21	3878	546	113	2835	5917	10271	24397	2096908
23	3242	346	72	4946	9169	15964	28257	2111793
25	2894	346	70	5191	9435	17757	26243	2115908
27	2647	336	63	4949	9765	20044	47397	2118140
29	2379	322	64	5253	9654	21236	47400	2117602
31	**2163**	**314**	**61**	**5106**	**10093**	**21391**	**47434**	**2115387**
33	5445	478	65	4131	9604	21238	47434	2223152
35	6141	548	69	3863	9293	19319	39086	2262014
37	6406	634	76	3462	8487	17400	35586	2300658
39	6050	683	86	3167	7600	16871	33913	2316137
41	5795	700	92	3098	7244	15760	32870	2316339
43	5409	717	92	2756	7073	15330	32877	2316498
45	5204	745	100	2597	6330	14459	28336	2311673
47	5091	838	107	2244	6006	13765	28337	2352640
49	4964	889	125	2040	5256	10541	28338	2339993
51	4854	933	141	1807	4991	9321	20945	2341919
53	4783	1005	156	1542	4651	8595	20947	2354645
57	4539	1114	192	1345	3710	7054	19207	2340257
59	4526	1186	217	1245	3236	6264	18179	2341370
61	4571	1280	240	1130	2831	5634	13665	2352709
63	4585	1354	265	1057	2583	5261	13666	2349722
65	4435	1416	289	997	2404	4738	13666	2326947
67	4400	1504	319	953	2235	4307	11392	2338606
69	4462	1501	327	943	2101	4051	11392	2279552
71	4461	1568	349	896	1932	3725	11393	2267731
73	4465	1633	383	868	1766	3339	11393	2258398
75	4531	1703	426	837	1602	2921	11367	2227034
77	4614	1788	462	803	1468	2670	11367	2229053
79	4675	1863	498	769	1365	2526	7880	2196570
81	4752	1896	523	751	1292	2289	7880	2149265
83	4866	1966	561	722	1193	2115	7076	2120606
85	4945	2009	590	692	1125	1926	5918	2071385
87	5100	2030	614	680	1060	1812	5910	2024224
89	5208	2007	624	669	1025	1695	5910	1938826
91	5335	2007	638	651	967	1564	5910	1861146
93	5464	1988	651	641	924	1487	4228	1783503
95	5621	1971	661	626	888	1404	4054	1712710
97	5752	1929	668	616	836	1305	3152	1618998
99	5870	1872	661	610	823	1245	3152	1538072
101	5872	1774	636	603	796	1176	3152	1421034
103	5903	1675	620	590	765	1108	3152	1298068

105	5959	1575	589	586	751	1053	3146	1194411
107	6089	1452	554	574	718	986	2875	1067259
109	5962	1323	511	564	691	942	2875	945745
111	5821	1175	464	559	681	918	2742	826367
113	5762	1081	427	555	671	903	1882	750701
115	5532	923	369	546	657	879	1705	626868
117	5342	809	330	549	645	851	1564	542982
119	4949	680	281	543	626	801	1602	444226
121	4619	578	240	541	618	793	1602	373049
123	4234	447	189	537	608	769	1602	283518
125	3936	367	158	529	601	723	1602	227969
127	3513	266	114	532	598	715	1602	165294
129	3039	197	86	524	589	691	1602	120029
131	2625	144	63	529	578	683	1499	87615
133	2127	105	45	530	589	692	1499	64326
135	1740	74	33	528	578	700	1431	44811
137	1364	47	21	543	597	713	1431	29316
139	1081	32	14	546	598	713	886	19757
141	813	21	9	531	649	740	886	13071
143	565	14	6	520	673	740	886	8805
145	381	12	6	531	632	775	886	7549
147	227	9	4	603	724	1034	1310	6883
149	157	9	4	621	724	1539	1539	7194
151	115	7	3	604	769	859	961	5003
153	74	7	3	604	742	859	961	4973
155	41	5	2	615	961	1362	1362	4284
157	24	5	3	615	692	952	952	3745
159	19	7	4	551	615	699	896	4559
161	13	3	1	551	1281	1281	1281	2505
163	7	3	1	550	1215	1215	1215	2344
165	8	2	1	510	918	918	918	1428

Table A1.6. Results from **Velvet** optimization by adjusting the *K* (K-mer length) parameter. Given that the long-read (454/Roche) assembly option does not generate any contigs, the results correspond to the execution using the short-read assembly option. All assembled contigs are of length equal to or larger than 500 bp.

K	Contigs ≥ 500 bp	Contigs ≥ N50	N80	N50	N20	Max	Sum
17	29	14	523	566	626	770	16701
23	500	207	546	651	825	1451	330142
29	553	225	545	648	844	1628	369875
35	557	228	544	630	825	1649	366242
41	568	234	544	635	810	1623	374222
47	573	236	543	652	820	1700	381111
53	513	213	550	650	813	1455	340659
59	493	202	542	638	826	1413	324349
65	462	195	536	621	788	1367	295303
71	436	186	531	606	749	1557	272536
77	489	213	528	596	714	1639	299958
83	604	255	532	619	766	1333	382041
89	804	327	544	650	851	1698	537408
95	1027	404	565	700	947	2330	731233
101	1219	460	582	755	1049	2162	917950
107	1139	432	574	729	1019	2478	842028
113	1402	506	605	814	1177	2778	1129813
119	1583	541	636	907	1352	2766	1378800
125	1668	554	668	979	1512	4105	1552134
131	**1393**	**418**	**800**	**1358**	**2119**	**4300**	**1626257**
137	1709	565	650	953	1524	4542	1570086
143	1642	564	621	872	1338	3076	1403425
149	1510	553	595	784	1146	2461	1180998
155	1283	496	571	709	978	2461	927267
161	991	402	545	647	850	1903	661218
167	657	280	533	607	736	1982	412843
173	305	135	546	619	722	1475	192313
179	99	46	562	599	685	1089	61182
185	36	17	567	596	650	1109	22398
191	10	5	582	611	647	787	6269
199	1	1	766	766	766	766	766

A2 Assembly parameter optimization in MIRA

The tables shown below present the results from the optimization of the different assembly parameters studied in the MIRA assembler. In all the tables, the second, third and fourth columns represent the number of contigs, and the next ones, total lengths expressed in base pairs. The row corresponding to the optimal adjustment has been highlighted in bold type.

Table A2.1. Number of assembly passes (parameter *-AS:nop*).

A) nop = 4
B) nop = 5 (default)
C) nop = 6
D) nop = 7

	Contigs ≥ 500 bp	Contigs ≥ N50	Min	N80	N50	N20	Max	Sum
A	36	4	512	68610	168837	580365	580365	2185778
B	**36**	**4**	**559**	**71366**	**229335**	**526123**	**526123**	**2189373**
C	36	5	571	70546	167153	231381	397507	2191110
D	38	4	580	68996	220549	305168	320494	2192642

Table A2.2. Uniform read distribution (parameters *-AS:urd*, *-AS:urdsip*).

A) urd = no (default)
B) urd = yes, start in pass 3/5 (urdsip = 3)

	Contigs ≥ 500 bp	Contigs ≥ N50	Min	N80	N50	N20	Max	Sum
A	**36**	**4**	**559**	**71366**	**229335**	**526123**	**526123**	**2189373**
B	53	4	510	52723	163382	526169	526169	2201948

Table A2.3. Keep long repeats in separate contigs (parameter *-AS:klrs*).

A) klrs = no (default)
B) klrs = yes

	Contigs ≥ 500 bp	Contigs ≥ N50	Min	N80	N50	N20	Max	Sum
A	**36**	**4**	**559**	**71366**	**229335**	**526123**	**526123**	**2189373**
B	39	4	537	52409	231492	469479	469479	2190975

Table A2.4. Maximum number of times a contig can be rebuilt during a pass (parameter *-AS:rbl*).

A) rbl = 2
B) rbl = 3 (default)
C) rbl = 4
D) rbl = 5

	Contigs ≥ 500 bp	Contigs ≥ N50	Min	N80	N50	N20	Max	Sum
A	35	3	525	65149	163612	803227	803227	2194033
B	**36**	**4**	**559**	**71366**	**229335**	**526123**	**526123**	**2189373**
C	36	5	520	65122	120434	284182	333793	2194048
D	36	5	544	71142	180927	281761	389058	2187642

Table A2.5. Automatic repeat detection: Minimum read length in bp (parameter *-AS:ardml*).

A) ardml = 150
B) ardml = 180
C) ardml = 200 (default)
D) ardml = 220
E) ardml = 250

	Contigs ≥ 500 bp	Contigs ≥ N50	Min	N80	N50	N20	Max	Sum
A	35	4	519	65121	211040	462020	462020	2188347
B	38	6	523	64333	150730	231493	305461	2189728
C	**36**	**4**	**559**	**71366**	**229335**	**526123**	**526123**	**2189373**
D	39	5	568	60428	168834	240988	349512	2193023
E	39	4	540	65703	137085	569622	569622	2189295

Table A2.6. Minimum size of "large" contigs, expressed in bp (parameter *-MI:lcs*). Given that this parameter only affects the output set of contigs, an optimal value cannot be determined. Its default value has been maintained, leaving its variation, if necessary, in the user's hands.

A) lcs = 500 (default)
B) lcs = 1000
C) lcs = 1500
D) lcs = 2000
E) lcs = 2500
F) lcs = 3000
G) lcs = 3500
H) lcs = 4000

	Contigs ≥ 500 bp	Contigs ≥ N50	Min	N80	N50	N20	Max	Sum
A	36	4	559	71366	229335	526123	526123	2189373
B	29	4	1264	71366	229335	526123	526123	2184683
C	28	4	1646	71366	229335	526123	526123	2183419
D	27	4	2619	71366	229335	526123	526123	2181773
E	24	4	2880	71366	229335	526123	526123	2173699
F	22	4	4639	71366	229335	526123	526123	2167414
G	22	4	4639	71366	229335	526123	526123	2167414
H	27	4	2619	71366	229335	526123	526123	2181773

Table A2.7. Minimum overlap length and minimum overlap score (parameters *-AL:mo*, *-AL:ms*). The scoring scheme can be consulted at the MIRA user manual [59].

A) mo = 13, ms = 10
B) mo = 15, ms = 12
C) mo = 16, ms = 13
D) mo = 17, ms = 13
E) mo = 18, ms = 14
F) mo = 19, ms = 15 (default)
G) mo = 24, ms = 20
H) mo = 29, ms = 24
I) mo = 36, ms = 30
J) mo = 38, ms = 34

	Contigs ≥ 500 bp	Contigs ≥ N50	Min	N80	N50	N20	Max	Sum
A	40	6	577	67780	144253	225714	333880	2195197
B	38	3	508	65123	211047	682471	682471	2192529
C	36	3	589	68637	266509	682470	682470	2192189
D	33	4	505	71366	228839	307930	437252	2191795
E	33	5	511	71404	168841	298650	345599	2189842
F	**36**	**4**	**559**	**71366**	**229335**	**526123**	**526123**	**2189373**
G	40	6	537	51630	117292	229329	305661	2190101
H	37	5	558	70378	120850	284200	308195	2190117
I	32	4	502	71420	240989	296500	345600	2191370
J	34	4	514	53207	114343	803406	803406	2194572

Table A2.8. Minimum number of reads per contig (parameter *-AS:mrpc*).

A) mrpc = 3
B) mrpc = 4
C) mrpc = 5 (default)
D) mrpc = 6
E) mrpc = 8
F) mrpc = 15
G) mrpc = 25
H) mrpc = 35
I) mrpc = 45
J) mrpc = 55
K) mrpc = 65
L) mrpc = 80
M) mrpc = 100
N) mrpc = 100
O) mrpc = 140

	Contigs ≥ 500 bp	Contigs ≥ N50	Min	N80	N50	N20	Max	Sum
A	37	4	512	71404	229346	307618	345596	2190782
B	36	4	521	53207	178034	333799	423793	2194288
C	**36**	**4**	**559**	**71366**	**229335**	**526123**	**526123**	**2189373**
D	37	3	557	68800	198259	803389	803389	2195555
E	36	5	503	65121	156319	254643	363249	2190910
F	35	4	500	65664	220667	462030	462030	2183022
G	37	3	532	68497	231887	574833	574833	2189624
H	36	4	520	65121	120849	617884	617884	2190812
I	35	5	520	70377	157173	277040	349517	2192782
J	44	5	538	64988	95480	462029	462029	2195087
K	39	5	521	65509	108623	536926	536926	2185315
L	37	4	573	64988	211047	484286	484286	2189905
M	36	5	512	66251	144214	254689	363249	2186562
N	35	4	511	71426	218138	286614	345002	2189224
O	37	4	572	65124	162408	484287	484287	2192332

Table A2.9. Minimum read length, expressed in bp (parameter *-AS:mrl*).

A) mrl = 20
B) mrl = 30
C) mrl = 40 (default)
D) mrl = 50
E) mrl = 60

	Contigs ≥ 500 bp	Contigs ≥ N50	Min	N80	N50	N20	Max	Sum
A	38	5	514	65630	163619	231475	397331	2189879
B	34	4	543	65813	229259	437187	437187	2183922
C	**36**	**4**	**559**	**71366**	**229335**	**526123**	**526123**	**2189373**
D	37	4	516	65125	180809	501902	501902	2190319
E	39	4	506	65149	180216	501889	501889	2196082