MetaGen-FRAME

Metagenomics Data Analysis Framework Focused on Stressed Microbial Communities

Miguel Carvalho Valente Esaguy Coimbra

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Examination Committee
Chairperson: Prof. Joaquim Armando Pires Jorge
Supervisor: Prof. Luís Manuel Silveira Russo
Co-supervisor: Prof. Ana Teresa Correia de Freitas
Member of the Committee: Doutor Pedro Tiago Gonçalves Monteiro

November 2013
Resumo

A área da metagenómica consiste no estudo de metagenomas, que são materiais genéticos não processados, encontrados nos mais variados ambientes e contendo a informação de diversos organismos individuais em conjuntos. As abordagens da metagenómica ao estudo de comunidades biológicas estão a mudar rapidamente a nossa compreensão das relações entre organismos e das suas funções em ecossistemas. Os avanços na área da metagenómica são, em grande parte, resultantes da acen- tuada evolução das plataformas de sequenciação (high throughput platforms) de ADN, exigindo, como consequência, avanços significativos nas técnicas de análise de dados.

Com este trabalho, tencionou-se desenvolver e aplicar novas técnicas de análise de dados que possam ser usadas no processamento dos vastos volumes de dados gerados no contexto da metagenómica. Este documento apresenta uma proposta (e respetiva resolução) para lidar com os desafios inerentes ao armazenamento e manipulação deste género de dados, bem como a necessidade de desenvolver técnicas de análise de dados que possam ser aplicadas diretamente a este problema. Indo ao encontro deste propósito, este projeto foi desenvolvido com a intenção adicional de explorar os potenciais da computação paralela.

Este projeto resultou na criação da MetaGen-FRAME, uma framework metagenómica capaz de gerir um leque heterogéneo de tipos de dados (de sequências de ADN a anotações genómicas, proteómicas e metabolómicas) através do uso de diferentes estruturas de dados e abordagens computacionais.

Keywords: Metagenómica, Anotação, Taxonomia, Resequenciação, Mapeamento, Framework
Abstract

Metagenomics is the study of metagenomes, unprocessed genetic material residing in the most varied sites, without separation into individual organisms. Metagenomic approaches to the study of biological communities are quickly changing our understanding of the function and inter-relationships among living organisms in ecosystems. The rapid advances in metagenomics are largely due to the hasty development of high throughput platforms for deoxyribonucleic acid (DNA) sequencing, that need to be accompanied by significant advances in data analysis techniques.

With this work, I intended to develop and apply new techniques for data analysis that can be applied to large amounts of data generated by metagenomics. This document presents a proposal to address the challenges posed by the storage and manipulation of such information types and the need to develop new data analysis techniques that can be applied directly to this problem. For this purpose, there was an intention to harness the power of parallel computing.

The target-result of this thesis was MetaGen-FRAME, a metagenomic framework capable of handling heterogeneous data types (from DNA sequences to genome, proteome and metabolome annotations) though the use of different data structures and computational approaches.

Keywords: Metagenomics, Annotation, Taxonomy, Resequencing, Mapping, Framework
Acknowledgements

This document was the final product of an exhausting year of work as a researcher. It was a journey filled with hardships, which I was able to conquer thanks to the presence of many people in my life.

At the top of the list, I would like to thank my supervisors for always finding a slot for me on their busy schedules and providing me with valuable guidance. Their availability for discussing the most varied topics and the continuous encouragement they bestowed upon me were crucial to seeing this project through.

I am also thankful for some insight provided by Prof. Doctor João Carriço during the initial development stages of the project and also the staff of the Knowledge Discovery and Bioinformatics (KDBIO) group at Instituto de Engenharia de Sistemas e Computadores: Investigação e Desenvolvimento em Lisboa (INESC-ID) for valuable technical and infrastructural support. The group's researchers were also keen on stimulating discussions and providing important feedback on my presentations and work. In particular, I would also like to thank Francisco Fernandes for his constant availability regarding the improvements developed for the TAPyR aligner. Additionally, I would like to thank the faculty of the Computational Biology course \(^1\) at Instituto Superior Técnico for granting permission to use some of their material in this document.

The development of this work occurred in collaboration with Brasil’s Laboratório Nacional de Computação Científica \(^2\), with its valuable expertise. This collaboration took place in the context of Petrobras’ oil and gas projects.

I was fortunate to be conceded a scholarship by FCT to support my work, by national funds through FCT Fundação para a Ciência e a Tecnologia, under project TAGS PTDC/EIA-EIA/112283/2009.

Lastly, I hold eternal gratitude towards my parents for spurring me into accomplishing this project and for giving me the tools in life which allowed me to accomplish it.

---

\(^1\) Course page available at [https://fenix.ist.utl.pt/disciplinas/bc/2012-2013/1-semestre](https://fenix.ist.utl.pt/disciplinas/bc/2012-2013/1-semestre)

\(^2\) Its web page can be found at [http://www.labinfo.incc.br/](http://www.labinfo.incc.br/)
It wasn’t the New World that mattered ... Columbus died almost without seeing it; and not really knowing what he had discovered. It’s life that matters, nothing but life — the process of discovering, the everlasting and perpetual process, not the discovery itself, at all.

— Fyodor Mikhailovich Dostoyevsky
# Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resumo</td>
<td>i</td>
</tr>
<tr>
<td>Abstract</td>
<td>ii</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>iii</td>
</tr>
<tr>
<td>1 Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Motivation</td>
<td>1</td>
</tr>
<tr>
<td>1.1.1 Historical Background of Metagenomics</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Objectives</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Contributions</td>
<td>4</td>
</tr>
<tr>
<td>1.4 Notation</td>
<td>6</td>
</tr>
<tr>
<td>1.5 Outline</td>
<td>7</td>
</tr>
<tr>
<td>2 Related Work</td>
<td>8</td>
</tr>
<tr>
<td>2.1 An Evolving Problem</td>
<td>8</td>
</tr>
<tr>
<td>2.2 Example Frameworks</td>
<td>9</td>
</tr>
<tr>
<td>2.3 Data Quality Control</td>
<td>10</td>
</tr>
<tr>
<td>2.4 Taxonomy, Homology and Annotations</td>
<td>11</td>
</tr>
<tr>
<td>2.5 Gene Identification and Annotations</td>
<td>14</td>
</tr>
<tr>
<td>2.6 Assemblies: What do They Represent?</td>
<td>17</td>
</tr>
<tr>
<td>2.7 Alignment/Resequencing</td>
<td>18</td>
</tr>
<tr>
<td>2.8 De Novo Assembly</td>
<td>21</td>
</tr>
<tr>
<td>3 MetaGen-FRAME Architecture</td>
<td>23</td>
</tr>
<tr>
<td>3.1 Final Revision</td>
<td>23</td>
</tr>
<tr>
<td>3.2 Additional Properties</td>
<td>26</td>
</tr>
</tbody>
</table>
List of Tables

5.1 Taxonomic Analysis: MetaPhlAn (defaults). Statistics for the taxonomy of the listed metagenomes. For each data set, the two identified OTUs with the highest identity are shown. Further analysis was performed on these top references. ........................................ 46

5.2 Alignment: P-TAPyR. This table has statistics for the resequencing-based assembly with the previous top references. ................................................................. 46

5.3 De Novo Assembly: MetaVelvet (defaults). Contig count and average sequence lengths after processing sets of unaligned sequences using MetaVelvet, which is based on the decomposition of de Bruijn graphs. ................................................................. 46
List of Figures

2.1 The most relevant subset of the major taxonomic ranks in the hierarchy of biological classification. The most relevant part of the hierarchy for MetaGen-FRAME software. 

2.2 Burrows-Wheeler Transform: suffixes of string $S=GTGATAACGATATAC$ in a suffix array, shown in increasing index order. 

2.3 Burrows-Wheeler Transform: lexicographically-ordered suffixes of string $S$ and the algorithm's output string $L$, under the column of the special terminating (16-th) character $. 

2.4 Burrows-Wheeler Transform: resolving the first step of query word $Q$ on the top-left side. 

2.5 Burrows-Wheeler Transform: resolving the second step of query word $Q$. 

2.6 Burrows-Wheeler Transform: resolving the last step of query word $Q$, terminating the search procedure. 

3.7 Overview of the main processes of the framework, based on the final stages of its implementation with the support of Taverna. This is purely a biological-oriented scheme of the framework. The Remote WS box represents the process of communicating with an external entity (such as the National Center for Biotechnology Information (NCBI) servers) to obtain information and data sets associated with specific operational taxonomic references. Before the task of Gene Structure Prediction, one can see the addition of the De Novo Assembly task, whose output contigs are then used in the gene structure prediction. These last two sequential tasks form a de novo task branch. 

3.8 Example of the internal taxonomy standard to be output by the Taxonomic Analysis bioinformatics task. In this case, through the use of the taxonomic threshold input parameter of the framework, it was specified that only the two highest percentage identity detected ranks present would be shown. 

3.9 Example of the internal strain reference path list standard to be output by the Remote WS task. The FASTA file names are of the form $X_{\text{ref}.fasta}$, where $X$ is the NCBI GI number of the strain’s genome. 

4.10 This scheme illustrates the way software tools are executed. The bioinformatics task composition is represented in Taverna (through the configuration file Skeleton.t2flow). Every concrete software tool implemented interfaces with the Taverna-based workflow (L1 in the figure) through the use of Python (L2 in the figure) scripts and a set of configuration files to be present in the local machine.
4.11 The illustration serves the purpose of highlighting the approach of separating software tools from the bioinformatics concept they aim to satisfy.

4.12 Example: generated set of directories during an execution of the framework in Taverna.

4.13 Overall view of the Taverna-based workflow which defines MetaGen-FRAME.

4.14 The Java-based application used to configure the taverna_path.ini file which is used to store a valid path to Taverna.

4.15 The configuration of the web service composition used in the framework to obtain reference genomes from the online resources of the NCBI. The cyan squares represent constant values to be input to each processing task in this scheme. The brown ones are simple beanshell scripts used to convert data formats in order to bridge NCBI queries.

4.16 A short segment highlighting the possible contents of the .metaframe configuration file. As shown, it can contain two categories of sections, with one type dedicated to defining remote servers to be targeted during execution and the other serving the purpose of defining the execution mode and other specific properties of programs. The first two sections displayed in the figure show the typical properties to be defined for a desired remote server, such as the RSA key (used when communicating with the remote server using the SSH2 variant of the Secure Shell (SSH) protocol). Additional detail given in section 4.5). The last two sections are for program configurations: the first one specifies that MetaPhlAn is to be run locally and to return taxonomic results in decreasing value of identity; the second one configures remote parallel TAPyR execution.

4.17 Example of the contents of the configuration file for MetaGen-FRAME’s bioinformatics task of Taxonomy. As visible in the image, this template format always begins with a header section with two variables, one defining the amount of programs that have been added to the framework and the other showing the concrete program selected for execution in a workflow. The value of chosenprogram must be the name of one of the sections that appear in the following lines of the document.

4.18 Illustration of a hypothetical execution scenario for MetaGen-FRAME. It may be configured so that different bioinformatics task software executes on separate remote machines, which may be directly-accessible over the Internet, or may require an indirect connection through a gateway and further into a private network.

5.19 Measurements of metagenomic conch (against indexed Pseudomonas fluorescens and Sulfurospirillum deleyianum) data set speedups.

5.20 Timing of the metagenomic conch (against indexed Pseudomonas fluorescens and Sulfurospirillum deleyianum) data set execution.

5.21 A phylogenetic tree we produced during collaboration efforts between KDBIO and LNCC. It shows a grouped analysis for the metagenomes previously mentioned. The purple area A on the top-left side refers to the genus Sphingobacteriaceae unclassified while the cyan B area on the bottom is associated to the genus Candidatus Pelagibacter.
5.22 A depiction of OTU distribution through the analyzed metagenomes. It is particularly relevant to note the high presence (denoted by the yellow coloring) of the Candidatus Pelagibacter ubique, important for some of its capabilities for recycling dissolved organic carbon (dissolved organic carbon plays an important role in the global carbon cycle and is also relevant in the transport of metals in aquatic systems).

5.23 Species distribution for the contigs related to the valid alignments obtained with the different strains of E. coli. The longest cyan block at the top illustrates the dominance of this organism in terms of percentage of BLAST hits.

5.24 Counts of Gene Ontology biological processes and the amount of sequences associated to each of them. Although not the most present biological process term, it should be noted the considerable presence of phosphorylation-related terms (fourth red rectangle from the top). Phosphorylation is relevant for its ability to activate and deactivate many protein enzymes, altering their activity and function as a result.

5.25 Distribution of Gene Ontology terms related to cellular components.

5.26 Sequence counts for several Gene Ontology terms, obtained with Blast2GO. Particularly important is the presence of ATP binding terms, which are responsible for the molecular function of enzyme and coenzyme regulation (second red rectangle from the top).

B.27 Measurements of Caenorhabditis elegans index data set speedups.

B.28 Timing of the Caenorhabditis elegans index data set execution.

B.29 Measurements of Drosophila pseudoobscura index data set speedups.

B.30 Timing of the Drosophila pseudoobscura index data set execution.

B.31 Measurements of Plasmodium falciparum index data set speedups.

B.32 Timing of the Plasmodium falciparum index data set execution.
Acronyms

**BAC** Bacterial Artificial Chromosome. 25, *Glossary: Bacterial Artificial Chromosome*

**BLAST** Basic Local Alignment Search Tool. 15, 16, 18, 19, 28, 29, 46–48, 54, *Glossary: BLAST*

**CPU** Central Processing Unit. 51

**DNA** Deoxyribonucleic acid. iii, 12, 15, 19–21, 23, *Glossary: DNA*

**EST** Expressed Sequence Tag. 19, *Glossary: EST*

**GPU** Graphics Processing Unit. 62

**GUI** Graphical User Interface. 28, 38, 43, 44

**HMM** Hidden Markov Model. 10, 16, 34, *Glossary: HMM*

**HSP** High-Scoring Segment Pair. 18, *Glossary: HSP*

**HTML** HyperText Markup Language. 6, 11, 26, 29, 32, 33, 41

**HTTP** Hypertext Transfer Protocol. 44

**INESC-ID** Instituto de Engenharia de Sistemas e Computadores: Investigação e Desenvolvimento em Lisboa. v, 43, 46

**KDBIO** Knowledge Discovery and Bioinformatics. v, 4, 43, 46, 51

**NCBI** National Center for Biotechnology Information. xiii, xiv, 10, 14, 15, 18, 29, 31–34, 39, 42, 46–48, 54

**OTU** Operational Taxonomic Unit. 13, 54, *Glossary: OTU*

**RDF** Resource Description Framework. 62, *Glossary: RDF*

**RNA** Ribonucleic acid. 12, 18, *Glossary: RNA*

**RNL** Rede das Novas Licenciaturas. 43
SAM  Sequence Alignment/Map. 22, Glossary: SAM
SCP  Secure Copy. 46
SFF  Standard Flowgram Format. 24, Glossary: SFF
SSH  Secure Shell. xiv, 44–47, 49, Glossary: SSH
XML  Extensible Markup Language. 16, 29, 44, 47, 48, Glossary: XML
Glossary

**Bacterial Artificial Chromosome** is a DNA construct that is typically used in genome projects to sequence the genome of organisms. A short piece of the organism’s DNA is amplified as an insert BACs and then sequenced.

**BAM** is the compressed version of the SAM format.

**BLAST** is an algorithm used to compare primary biological sequence information, such as amino-acid sequences of proteins or nucleotides of DNA sequences. A user provides the algorithm with a given sequence, and in turn it is compared with the sequences of a set of libraries or databases, returning the sequences with at least a given degree of similarity. The algorithm may be mentioned (in its actual program name in the framework, blast) with slight variations in name. This is because there are versions of BLAST for different specific biological purposes.

**Codon** is a sequence of nucleotide triplets which specifies amino acid composition in protein synthesis.

**Contig** is the name typically given to a sequence which results from reads progressively (and usually in an iterative fashion) added to extend a growing sequence. Contig creation is a common step during assembly of genomes or metagenomes.

**CpG Islands** are regions of DNA characterized by a large number of adjacent cytosine and guanine nucleotides linked by phosphodiester bonds.

**DNA** is a family of biological molecules which carry information, essentially encoding genetic instructions that play a role in the development and functioning of all living organisms known, as well as many viruses. Nucleotides of DNA molecules are represented by one of four letters: A (adenine), C (cytosine), T (thymine) and G (guanine). DNA has the sugar deoxyribose.

**EST** is a short sub-sequence of a cDNA sequence. They may be used to identify gene transcripts, and are instrumental in gene discovery and gene sequence determination.

**Eukaryotes** are organisms whose cells contain complex structures enclosed within membranes. The nucleus is the membrane-bound structure of the eukaryotic cell, which distinguishes it from a prokaryotic cell.
**FASTA** files, typically ending with `.fna` and `.qual` (fasta nucleotide sequence and fasta amino acid, in case of separate quality and sequence files) or `.fasta/.FASTA`, are a text format used to represent nucleotides or amino acids with single-letter codes. It is a standard in the field of bioinformatics. For 454 sequencing, `.fna` are sequence files and the `.qual` files are complementary base quality score files. This format stores both the sequences and the quality scores in the same file. Sequence letters and quality scores are both encoded with a single ASCII character. xiv, 10, 11, 31–33, 39, 40

**FASTQ** files, typically ending with `.fastq/.FASTQ`, store both the sequences and the quality scores in the same file. Sequence letters and quality scores are both encoded with a single ASCII character. 11, 31

**FIGfam** is a protein family generated by the Fellowship for Interpretation of Genomes. They are based on collections of subsystems or correspondence between genes in closely related strains. It is considered that two PEGs (Protein-Encoding Genes) that occur in a single FIGfam have the same function. Typically, conservative procedures are used to decide whether a gene may be safely assigned to the function associated with the FIGfam. 13

**Functional Roles** are associated with the logical roles that gene products or genes may assume in a cell's operation. 15

**GC-content** refers to a DNA molecule's percentage of nitrogenous bases which are one of guanine or cytosine. In the context of genetics, this is relevant to the identification of gene structures. Genes in a genome often have a GC-content higher in proportion to the GC-content of the whole genome. Related to this is the concept of CpG islands, which are DNA regions with the characteristic of having a large number of adjacent C and G nucleotides. 12

**HMM** is a statistical method with a state machine-based representation in order to compute probabilities associated with desirable outcomes.

**Homologous Sequences** are sequences related to one another by diverging from a common ancestor.

**Homopolymers** are sequences of identical bases (such as "AAAA"). In the 454 sequencing method, bases are not called directly. Light signals are used in this technology to indicate flows, where the flow represents a homopolymer and the light's brightness indicates the length. Due to difficulties with calibration of the light, sometimes homopolymer-length sequencing errors arise. 11

**HSP** refers to sub-segments of two sequences that are considered highly similar. Similarity between sequences depends on several factors such as the type of alignment desired (local or global), parameters given to algorithms, among others.

**K-mers** are k-tuples of nucleic acid or amino acid sequences, which may be used to identify certain regions within biomolecules like DNA or proteins. 20

**Metabolic Network** is the collection of metabolic pathways that may co-exist within a given cell.
Metabolic Pathway is a complex series of chemical reactions (occurring within cells) that are sequentially catalyzed by various enzymes, leading to the synthesis of a target molecule. The majority of nutrients that are essential to microorganism growth and survival are synthesized from environmental sugars by hundreds of metabolic pathways. Every pathway contains a principal chemical, which is modified by these series of chemical reactions. The enzymes that trigger these reactions often require dietary minerals, vitamins and other cofactors in order to function properly. Catabolic (break-down) and Anabolic (synthesis) pathways often work interdependently to create new biomolecules as the final end-products. Metabolic pathways play an important role in maintaining the homeostasis within an organism. 5

Metabolite is the name attributed to each of the many chemicals that may play a role in the chemical reactions of metabolic pathways. They can make the metabolic pathways quite elaborate.

N50 is a statistic typically associated with contig lengths within a draft assembly. For a set of sequences with different lengths, it is defined in the literature as the contig length for which the set of contigs with greater or equal lengths hold roughly fifty percent of the bases in the assembly. 25

Nucleic Acid molecules have, when considering a single strand of nucleic acid molecule, a phosphoryl end (5'-end, "five-prime") and a hydroxyl (3'-end, "three-prime"). It is in the 5'-end to the 3'-end direction that sequencing is performed. In this direction, there are three possible reading frames that may be read, each beginning from a different nucleotide in a triplet. RNA and DNA are nucleic acids, both being assembled as a chain of nucleotides, which are the building blocks of their molecules. However, there are biological differences that set them apart. Firstly, their names are derived from a specific type of sugar each. Together with protein, they form a set of three major macromolecules essential to all forms of life.

OTU is defined by the NCBI: "Taxonomic level of sampling selected by the user to be used in a study, such as individuals, populations, species, genera, or bacterial strains.”.

Paired-end Read is obtained through a sequencing technology by sequencing the ends of both strands of the same molecule. 23

Phylogeny is the study of relations of evolution among sets (species, populations) of organisms. 12

Prokaryotes are organisms whose cells do not have a nucleus or mitochondria: its intracellular water-soluble components are all together in the same area enclosed by the membrane of the cell. 1

Protein is a macromolecule essential to life, responsible for a plethora of functions within living organisms, including catalyzing metabolic reactions, replicating DNA and responding to stimuli. 13, 15–19

RDF is a family of specifications with a design originally intended as a data model for metadata. With various syntax notations and formats for data serialization, it has become a general method for the conceptual description or information modeling implemented in web resources.

Read is a term used interchangeably with sequence. 3, 4, 10–12, 14, 19–25, 28, 30, 34, 35, see sequence
**Reading Frames** are a way of dividing a sequence of nucleotides in a nucleic acid into a set of consecutive, non-overlapping triplets. Where the triplets equate to amino acids or stop signals during translation, they are called codons.

**RNA** is a family of biological molecules that perform important gene-related tasks, such as coding, decoding, regulation and expression of genes. There are different types of RNA associated with formation of proteins, such as mRNA, tRNA and rRNA. The nucleotides of this type of molecule are represented by one of four letters as well, all equal to those which represent DNA molecule nucleotides, the only exception being that instead of T (thymine), there is the letter U (uracil). RNA has the sugar ribose.

**RSA** is an algorithm based on public-key cryptography, based on the presumed difficulty of factoring large integers, which constitutes the so called *factoring problem*. The algorithm is a sequence of steps: generation of a key, encryption and decryption. xiv, 43, 45, 49

**SAM** file names typically end with .sam/.SAM. It is a format used for storing sequence data in a series of tab-delimited ASCII columns. Currently, this format is produced by aligners that assign sequences to a position regarding a known reference genome.

**Scaffolding** is the process of constructing a scaffold, which is a set of contigs with an assigned orientation. 17

**Sequences** of nucleotides are obtained through sequencing technologies (such as the ones described in this document). iii, 3–5, 9–21, 24, 28–35, 39–41, 43, 44

**SFF** file names are generally terminated with .sff/.SFF. It is a binary file format which is used to encode the results of pyrosequencing from 454 Life Sciences’ high-throughput sequencing platform.

**SSH** is a network protocol with cryptographic properties for communicating with security of data, remote mode of both command-line login and command execution, and other secure network services between two networked computers that connects, via a secure channel over an insecure network, server and client machines (executing SSH server and SSH client programs).

**Subsystems** are sets of related genes. They are formally defined as an abstract set of functional roles, that together implement a specific biological process or structural complex. The term "subsystem" is used with flexibility, representing for example a metabolic pathway or a physical structure. 13

**XML** is a language for markup which defines a set of rules for encoding documents readable both by humans and machines. The aim of its design was to simplify, generalize and increase usability over the Internet.
Chapter 1

Introduction

1.1 Motivation

Metagenomics is a field which targets bacterial organisms, also known as prokaryotes. A metagenome consists of genetic material samples, recovered from a given environment. As opposed to the traditional analysis of cultivated clonal cultures, metagenomes cannot be reproduced in a laboratory environment for isolated analysis. The study of metagenomes provides important information associated with the way organisms relate to one another when coexisting in our planet's many natural environments (like acid mine drainage\(^1\) or human gut microbial communities). This field also provides insight on the major metabolic reactions that occur within a given community, deepening our comprehension of its influence on the environment where it dwells. Some information on the historical background of this field is given in section 1.1.1.

As a consequence of the rich information that may be extrapolated from the analysis of a metagenome, this field is of high relevance in many different contexts, having spanned several projects with the goal of taking advantage of its possibilities. There is a plethora of applications of the field of metagenomics. A prime example of this is the Human Microbiome Project \([1]\), which is one of many international efforts started and designed to apply metagenomic analysis to the study of human health. This project, in a way similar to that of the Human Genome Project, is expected, through worldwide cooperation, to help build a rich and comprehensive data set that is publicly available to investigators working towards the improvement of human health.

Another important application of metagenomics is industrial biofuel production, with the necessity to analyze microbial communities in order to isolate efficient and novel enzymes responsible for biomass deconstruction. Additionally, to properly ascertain control over this process, the perspective brought by metagenomics provides information on how these processes are triggered and influenced by microbial communities as a whole.

One other activity that benefits from the field of metagenomics is the agricultural sector. Plants grow in soils whose microbial communities are some of the most complex known to science. Despite this, and

\(^1\)Sources of flowing acidic water (with the existence of, for example, sulfur oxidation, iron oxidation and nitrogen fixation), such as mines and construction sites, among others.
the fact that they possess high economic value, there is still little of them known. These soil communities participate in a set of processes affecting plant growth, such as the fixation of nitrogen from the atmosphere, cycling of nutrients and disease suppression, amongst others. Knowing how these communities work would provide knowledge on how they influence these processes, which would benefit tasks such as crop disease detection (and cattle production as consequence).

The development of this project therefore focused on data sets belonging to environments subjected to biodegradation and certain levels of pressure, alike those partaking in the process of crude extraction. This work benefited from collaborations in the context of Petrobras\(^1\) oil and gas projects, through liaison with researchers from Brazil’s Laboratório Nacional de Computação Científica.

1.1.1 Historical Background of Metagenomics

This area of study may be viewed as a natural extension of the field of genomics, which focuses on studying the genomes of organisms. These disciplines rely on the use of sequencing technologies, which have been in constant development ever since their initial invention. One individual in particular, Craig Venter, should be credited for his efforts which spurred this initial development. Shotgun sequencing, one of the most intensively-used sequencing methods, was thought by most geneticists and researchers to lack the necessary accuracy to be applied to such a complex genome as that of human beings. Dissatisfied with the state of affairs at the time (circa 2000), Craig Venter sought support funding from the private sector, effectively amassing enough to fund Celera Genomics\(^2\). The main goal of this company was the sequencing of the entire human genome and its subsequent release into the public domain, a task which was achieved three years before the initially-estimated deadline. Thanks to the company’s contributions, and despite the controversy and skepticism surrounding shotgun sequencing at the time, it became a widely used and de facto standard.

1.2 Objectives

This section describes tasks that need to be performed in metagenomics, which vary in terms of biological scope. The majority of them require efficiency regarding the ability to handle large amounts of information in a robust way. This requirement is mostly tied to the constant evolution of sequencing technologies and the large amount of data they produce. There is also room for additional benefits, not only in the way these tasks are performed by making use of higher degrees of information, but also in the way they are integrated in software suites as a whole.

Many of the existing working solutions for metagenomic analysis have some sort of characteristic that proves limiting to researchers. A grand percentage of commercial solutions’ architectures are completely or highly-closed, rendering researchers and other professionals unable to pinpoint all the steps occurring inside the software platforms with enough transparency (besides what they can muster through official support documentation and publications). There are also open-source solutions freely-available


\(^2\) Celera Genomics, [https://www.celera.com/](https://www.celera.com/)
CHAPTER 1. INTRODUCTION

to everyone. These types of platforms, despite achieving considerable transparency in terms of integrated modules and parametrization, tend to incur some penalties, which originate from the fact that such platforms usually attempt to make use of separate software packages for specific needs, leading to additional complexity because different (and often orthogonal) submodules are articulated together to form pipelines or to provide a top-level functionality. Without proper documentation and automation mechanisms, this sometimes implies an increased overhead in configuring these platforms.

MetaGen-FRAME attempts to differentiate itself from other solutions regarding these negative characteristics. An open and flexible architecture was devised for it in order to allow for a clean and intuitive articulation (and configuration potential) of different bioinformatics modules as automatically as possible. By defining specific input and output formats for each bioinformatics task, as well as implementing consistent mechanisms to configure software execution, we believe this framework becomes capable of minimizing and avoiding these characteristics. Through these constructs, we aimed to produce an outcome that distances itself from these undesirable generic properties that commercial and open-source solutions typically have. It is also necessary to emphasize the aspect of configuration of the framework. Several areas of study, such as that of bioremediation, may require domain-specific criterion, for which customization capabilities would be most interesting. Attempting to satisfy these necessities was therefore another objective which MetaGen-FRAME aimed to fulfill.

In the field of metagenomics, there aren’t many approaches which attempt to integrate the specificities of the different biological concepts in existence. Therefore, delving into this fact, a proposal was made to define and implement a metagenomic framework that handles the types of metagenomic information contexts mentioned in (the subsections of) section 2, with a particular focus on processing, for the target test scenarios, data generated by 454 sequencing (pyrosequencing) technologies (MetaGen-FRAME’s range of applications is not limited to this, however). These target data sets are also associated to environments subjected to certain pressure and stress configurations. So, in essence, a major objective was to design and implement a metagenomic framework that uses and outputs information on different biological contexts, in order to produce relevant knowledge associated with the metabolism of the processed samples. Projects associated with oil and gas extraction and production processes are of particular relevance as a direct example of applying the capabilities of the framework.

Working towards the construction of such a framework, existing software tools (some of which are mentioned in this document, the most important ones being those described in section 2) were incorporated to satisfy the specific tasks. Another important objective was to define and implement the framework in a way that enables researchers and professionals without a strong background in computer engineering and information systems to configure the framework. This configuration was intended to be done at the bioinformatics task implementation level: for each bioinformatics task, there would be a well-defined relatively-easy and accessible method (its definition and implementation is described in section 3.2) by which an individual could replace the software package performing a specific bioinformatics task with another one. Given the different natures that bioinformatics projects may have, it was believed this was an ambitious engineering problem whose benefits far out-weighted its inherent difficulties, as there was confidence this extra flexibility would prove useful to any and whatsoever party that may use this project whilst needing to cater to specific needs.

Finally, the execution of this project was intended to produce a platform that, when analyzing data sets
retrieved from controlled environments, would be capable of outputting genetic information, valuable for its ability to aid in the identification of metabolic pathways (inside families and species) for the purpose of furthering investigation in laboratory facilities. If possible, it was desirable that MetaGen-FRAME would also contemplate some automated mechanisms with which to extrapolate this metabolism and metabolic pathway information from said genetic information.

1.3 Contributions

MetaGen-FRAME was defined with the prospects of performing the following tasks (as well as combining their results to acquire a more complete view of the data set’s information), described in high-level:

- **Data Quality Control**: for replicate sequence detection, treatment and overall filtering. This step cleanses the data, eliminating undesirable replicated reads (whose existence may be justified by reasons such as errors within the sequencing technology used). Throughout this work, solutions for replicate removal were explored, like the application of the Rabin-Karp algorithm \[2\]. This algorithm has an average running time of \(O(n+m)\) with a space usage bound by \(O(p)\) and a worst-case running time of \(O(m)\)\(^1\). The treatment task will eventually produce, besides the filtered data set, an intermediate set of files with information about the replicate sequences detected and other aspects of the quality standards enforced. The task of filtering is essential to ensure other tasks in MetaGen-FRAME are working on a data set satisfying minimal quality constraints.

- **Taxonomic Analysis**: for producing an overview of the biological composition of the processed sample and the major organisms and group classifications present. This task is quite important, as it will define the alignment targets of the task that follows.

- **Alignment**: this task employs the growing databases of (studied) genomes and metagenomes. Execution of this task has the goal of comparing (as per the definition of resequencing-based assembly given in section 2.6) the data set’s sequences against established and well-known organisms, to analyze how much they identify with them. In this resequencing context, there was an intention to use the program TAPyR. This program was developed as a state-of-the-art aligner by researchers at the Knowledge Discovery and Bioinformatics (KDBIO) group\(^2\) of INESC-ID. It was specifically designed for pyrosequencing technologies and proved to be far more efficient (in terms of both result quality and execution time) than other similar-purpose mainstream applications \[3\] (TAPyR’s applicability is not exclusive to pyrosequencing). However, the sequence alignment problem is still a computationally intensive task. Considering further performance improvements for this tool, adaptations to it were proposed to make use of multi-core systems. Specifically, it was set to allow for the mapping of read files concurrently against a single index. A consequence of this is that where a single file would have to be mapped serially, it would be possible to decompose it into fragments and submit each as a parallel mapping job. This had two important predicted advantages: a theoretical reduction in mapping time, proportional to the number of physical cores

\(^1\)For a text of length \(n\) and \(p\) patterns of combined length \(m\).

available, and reduced memory consumption. The later would be manifested by the fact that only a single reference index would be stored in memory at any time. With the serial version, if one wishes to map multiple read files against the same index, this requires several process instances of TAPyR running at the same time, which means that the system ends up having to store as many copies of the index in memory as the number of instances launched.

- **Functional Annotation:** as a means to connect biological information to existing genomic elements pertaining the sequences that aligned against the reference organisms with sufficient identity. (obtained in the resequencing task).

- **De novo Assembly:** to attempt to (as detailed in section 2.6 and its description of the de novo approach to assembly) compute a statistically-relevant set of consensus sequences and contigs from the initial (filtered) sample. These generated sequences will serve to perform further prediction-based methods to extract genetic information. The de novo assembly task is necessary to assess and evaluate sequences that, for other tasks, did not produce relevant results: lack of results in functional annotation, no mapping hits against well-known reference genomes that were retrieved, among others.

- **Gene Structure Prediction:** for speculation, statistical analysis and other forms of predicting the presence of genetic information on the computed de novo sequences.

- **Metabolic Reconstruction:** for attempting to harness the information obtained by the last tasks of the framework to produce data and results about the metabolic properties and networks observed in the sample, which are likely to be of interest to biologists and other professionals.

With this work, the metagenomic framework MetaGen-FRAME was implemented. It performs the following bioinformatics tasks: Data Quality Control followed by Taxonomic Analysis leading to Alignment, Functional Annotation and the branch of De Novo Assembly and that of Gene Structure Prediction. For each one, a software elected out of the studied candidates was incorporated as a default sub-module to come prepackaged with MetaGen-FRAME. A formal listing of said choices and the reasoning behind each one of them is provided in section 3.1. In the filtering task, the replicate removal operation is performed with support of the Python dict data structure. This approach was inspired by the concepts inherent to the Rabin-Karp algorithm. The replicate removal operation is therefore implemented as an individual Python module (detailed in section 4.3). The task of Metabolic Reconstruction was not implemented due to factors such as time constraints. Formally, a step toward the direction of supporting this task was planned, albeit modest: showing, at the end of a sample's processing, a unified view of the information produced by Functional Annotation and Gene Structure Prediction. Ultimately, the implementation of this step only went as far as covering Functional Annotation: an HyperText Markup Language (HTML)-based overview for the results of this task is produced when it finishes. MetaGen-FRAME may be obtained without cost at [http://web.ist.utl.pt/miguel.e.coimbra/metagen.html](http://web.ist.utl.pt/miguel.e.coimbra/metagen.html)\(^1\).

The proposed parallel modification of TAPyR, named P-TAPyR\(^2\), was also successfully implemented. The output of this effort was a software module written in C/C++ consisting of a parallel aligner designed

---

\(^1\)In the future, it shall be available at [https://kdbio.inesc-id.pt/~mcoimbra/metagenomics/MetaGen-FRAME.tar.gz](https://kdbio.inesc-id.pt/~mcoimbra/metagenomics/MetaGen-FRAME.tar.gz).

\(^2\)It may be obtained on the same website of the original version: [http://www.tapyr.net/](http://www.tapyr.net/).
CHAPTER 1. INTRODUCTION

with a server-client architecture. The implementation and testing of this parallel version of TAPyR by the author was a very important phase of this project, as care and additional precautions were taken to ensure the correct output formats and internal functionality were properly defined. Coordinated with the author’s development of MetaGen-FRAME, P-TAPyR still occupied a relevant position in development effort, as our collaborations with the staff at LNCC led to the identification of problems and errors in P-TAPyR, which were then relayed and fixed by the author as a follow-up to the meetings that took place. We partook in the COST-sponsored event SeqAhead Omics & Data Integration Workshops in Barcelona\(^1\) on February, 2013, where a poster session (the poster is presented at the end of Appendix B) featured P-TAPyR, showcasing the work performed. The development of this objective sub-project later culminated with the writing and submission of an article to the workshop PBio 2013: International Workshop on Parallelism in Bioinformatics\(^2\), held as part of the EuroMPI 2013 conference in Madrid on September. The article [4], after registration and submission, was accepted and included in the workshop proceedings published by the ACM Digital Library. Afterward, the subject and topics of the article were presented at the workshop. As part of the conference process, the article and presentation were evaluated by a jury. It was considered part of the best presented workshop papers, having spurred an invitation to be further extended and improved (work in progress), in order to be submitted to a special issue in the journal Parallel Computing (Elsevier).

The development of MetaGen-FRAME also benefited from close collaboration with the staff of INESC-ID’s Information System Group (ISG)\(^3\). Specifically, a researcher of ISG produced a project exploring the concepts of risk and data management using MetaGen-FRAME as a studied case, leading to the writing of the article Filipe Ferreira, Miguel E. Coimbra, Ricardo Vieira, Diogo Proença, Ana T. Freitas, Luis M. S. Russo, José Borbinha: Risk aware Data Management in Metagenomics, submitted to the 2013 edition of INForum - Simpósio de Informática, an event focused on divulgação, discussion and recognition of scientific works.

1.4 Notation

As the core of this work is a convergence of biology and informatics, a decision was made to employ a different font when mentioning certain concepts. This notation applies to specific biologically-relevant tasks mentioned to be in MetaGen-FRAME’s scope of functionality: Data Quality Control followed by Taxonomic Analysis, Remote WS, Alignment, which is a precursor for Functional Annotation, De Novo Assembly on which Gene Structure Prediction depends and finally the task of Metabolic Reconstruction. This notation also applies when referring to names of files contained in MetaGen-FRAME’s software distribution as well as some expressions and constructs used in the framework’s configuration files. Lastly, foreign terms, as well as programming and scripting language names are presented in italic.

\(^1\)Additional detail can be found at \texttt{http://www.seqahead.it/cost-bcn-2013/}
\(^2\)More information available at \texttt{http://arco.unex.es/mavega/pbio2013/}
\(^3\)The ISG homepage is available at \texttt{http://isg.inesc-id.pt/Home@1.aspx}
1.5 Outline

This report is divided in six chapters. Section 1.2 states the objectives set to accomplish with the work undertaken for this project. Section 2 covers the challenging tasks that typical metagenomic frameworks have to deal with, in order to successfully extract and harness information to achieve the benefits mentioned before. It also establishes a connection between each task and and a set of existing candidate software programs for it. Consequently, this section gives a general view on existing software tools and platforms for metagenomic studies, as well as the methods they employ to execute the tasks described. Section 3 showcases the final implementation of the project’s architecture. It explains and justifies certain design decisions and also compares it against the initial architecture proposal to highlight a process of constant evaluation. Section 4 then documents the implementation-related aspects of the principal components of MetaGen-FRAME and some of its more intrinsic and relevant modules. Afterward, section 5 details the execution scenarios used for testing the final implementation, showing obtained results and their evaluation. Lastly, section 6 begins with the most solid conclusions of this project. This document then terminates in 6.1 with remarks of prospects for future works based on what was achieved during the project.
Chapter 2

Related Work

2.1 An Evolving Problem

With the constant improvement of sequencing technologies, the quantity of information available on organisms and environments has increased at a dramatic rate, in both size and complexity. The improvement of these sequencing technologies represents new difficulties pertaining the analysis and processing of greater amounts of data. To address these difficulties, there is a growing need to adapt and combine existing algorithms and methodologies, in order to efficiently manipulate information. Since the initial introduction of shotgun sequencing, improved technologies have emerged, such as the next-generation sequencing platforms like the Roche 454 sequencer\(^1\), Illumina's Genome Analyzer\(^2\) and the SOLiD system of Applied Biosystems\(^3\), which had a considerable impact on metagenomic research. Another contribution by Craig Venter was the Sorcerer II Global Ocean Sampling project, one of the many metagenomic sequencing projects that added to the large data sets used to explore taxonomy, functional biodiversity and other aspects of diverse ecosystems [5].

Therefore, research and further reading must so be mindful that questions posed in subsequent sections take the context of data sizes with great orders of magnitude (hundreds of thousands of sequences spanning 30-500 base pairs, resulting from the scenario of modern sequencing technologies).

Before proceeding, it is relevant to remember that the field of metagenomics is influenced by numerous factors. One concrete example of this is the evolution of sequencing technologies and the appearance of new techniques, which lead to the creation of new data types and standards. It is also not an uncommon scene to encounter different software packages storing similar biological information in internal formats, which are typically in-house and not interchangeable, making cooperative efforts much more troublesome and time-consuming.

There is a multitude of interesting points when performing analysis on metagenomic data (and the biological communities they potentially contain). Many existing tools in this field share a subset of tasks for the retrieval and processing of specific information. Such particular corners of metagenomics include, though not limited to: community taxonomic analysis; metabolic reaction and metabolic pathway

\(^1\) 454 Life Sciences, http://www.my454.com/

\(^2\) Illumina, http://www.illumina.com/

\(^3\) Applied Biosystems by Life Technologies, http://www.appliedbiosystems.com/
identification within the community; attempted identification of community members by resequencing or predictive attempts by reconstruction of the whole metagenomic sample. In the literature, the tasks of sample identification by comparison and that of sample representation by reconstruction are referred to as different means of performing an assembly (a concept further detailed in this document).

The comparative identification of community members attempts to make use of the information built so far by the many research groups all over the world, stored in repositories such as that of the National Center for Biotechnology Information\(^1\). The aim of this task is to infer the sample’s properties by comparing the sample’s information with that of organisms which are already well-known (comparative assembly). Typically, this task’s goal is to obtain a genomic reconstruction.

Methods of sample reconstruction are to be used after exhausting analysis techniques that rely on known information. This implies that much of the reconstruction that is to occur is to be done in terms of the sample itself, with frequent employment of statistical methods like, for example, those based on Hidden Markov Model (HMM)s.

Metabolic analysis of the sample is relevant because it conveys details on the major chemical reactions occurring within the community. This is relevant in the context of learning about the community’s functioning as a system and how it impacts its environment (if applicable).

The following section will provide an overview of existing metagenomic pipeline solutions and the subsequent ones will be dedicated to specific bioinformatics tasks, consisting of building blocks used in many approaches to metagenomics, making use of task composition.

### 2.2 Example Frameworks

We now provide some examples of existing metagenomic platforms and their properties:

- **SABIA**, System for Automated Bacterial Integrated Annotation: a web-based software suite [6] which combines the process of genome assembly with that of genome annotation in a way that increases the quality of results. Through the use of different scripts and packages, it manages information in a relational database and is able to generate HTML reports and interfaces, allowing the users to access biological information and load results. It possesses two modules: one for contig assembly and another for annotations.

- **MetAMOS**, created as a step towards metagenomic analysis automation [7]. From NGS sequences, it is able to produce scaffolds, open-reading frames and functional or taxonomic annotations. MetAMOS is a package of software tools available publicly and capable of performing assembly and analysis. It is a pipeline with three major sections. A pre-processing step first builds a set of contigs using dynamic library size re-estimation based on read mappings. A second step follows, where contigs without mapping will be removed and scaffolds built. Lastly, MetAMOS performs result and statistic visualization, generating an HTML report.

- **RAMMCAP**, Rapid Analysis of Multiple Metagenomes with a Clustering and Annotation Pipeline [8], was developed with the aim of addressing the computational challenges imposed by the great size and diversity of metagenomic data. It employs clustering methods on sequences based on

---

\(^1\)National Center for Biotechnology Information (NCBI), http://www.ncbi.nlm.nih.gov
percentages of sequence identity over percentages of covered sequence length. The mechanisms employed in this pipeline allowed for the processing of the Global Ocean Sampling [9] study with a cost two orders of magnitude less than the original project, in terms of CPU hours.

- Galaxy [10], an open web-based platform for genomic research. It is capable of automating the tracking and management of data provenance and provides support for capturing the context and intent of computational methods. Workflows for processing metagenomic information have been built for this platform, performing interactions with Perl scripts, MySQL and other software executed in Galaxy servers, interfaced with Galaxy’s web-based support.

2.3 Data Quality Control

When working with either a genome or a metagenome, it is desirable to ensure the data set being studied falls under certain degrees of quality control. Throughout the sequencing process and storage of data, many aspects of the data may be altered one way or another, constituting factors that may influence the reliability and significance of an experiment. Among these factors one can include sequence artifacts (read errors), poor quality reads, primer/adaptor contamination and other types of errors. It is therefore important to employ methods that attempt to reduce the impact of these factors and filter only reads of high quality. These methods include tasks such as trimming the ends of reads (which may be lower-quality regions), calculation of average quality score for each sequence in a FASTA file, among others. Typically, before any form of processing is to occur over a data set, it is desirable to ensure the information falls within certain quality and control thresholds. Tools were developed in the past as a means to achieve this preparation. An example of such a tool is LUCY [11], which was developed and used to clean sequence data from sequencing technologies like Sanger. Another tool used for quality control is FastQC, with a specific focus on Illumina and ABI SOLiD technologies but also supporting sequences produced by 454 and PacBio. The group responsible for FastQC, Babraham Bioinformatics, claims it is not a research group. Their staff further state on their website that generally they do not produce their own articles and publications. As a consequence, publications which make use of FastQC were produced by unrelated work groups and focus on biological objectives. Some of them have included the creators of FastQC in their author lists, and that is as far as this tool is represented in the literature1. This program supports both interactive and non-interactive execution modes (this last mode is essential for integration in an automated framework such as MetaGen-FRAME) and has the ability to produce a set of results with a centralized presentation in an HTML-based report.

Moving through a set of candidates, another mention-worthy tool is PRINSEQ2, a tool which was designed with the particularities of 454 data in mind, but capable of being executed on the output of other sequencing technologies. It may be used with a web interface or through a standalone version. The NGS QC Toolkit [12] is a more recent application, supporting newer types of sequencing technologies. It specifically supports the Illumina-based and 454-based sequences. While it does not contain specific modules to process Ion Torrent sequences, they can be used for example in a Illumina-specific

---

1For a set of references which mention the use of FastQC and include their authors in the publication, please visit http://www.bioinformatics.babraham.ac.uk/publications.html.

2Some references regarding the usage of PRINSEQ for the task of quality control may be found at http://prinseq.sourceforge.net/manual.html#QCREFERENCES.
module assuming they are converted to that platform’s format (FASTQ). It uses the base-specific quality score Phred [13, 14], filtering as high-quality reads those that yield a score equal or higher than the one specified by the user. Pertaining this central quality check, there are different operations performed depending on the sequencing technology:

- **Illumina**: receives files in the FASTQ format as input. It begins with the quality step mentioned previously, and then executes a primer/adaptor check to remove contaminated reads.

- **454**: receives as input FASTA file pairs (.fna and .qual). First, it filters reads whose length is less than desired. Afterward, homopolymer trimming is performed, to minimize the impact of incorrect homopolymer lengths from 454 sequencing. The length of the homopolymer may be specified as an optional parameter. After this, the same steps of the Illumina files are performed: quality filtering and then primer/adaptor checking, in this case to do additional read trimming.

### 2.4 Taxonomy, Homology and Annotations

In metagenomics, besides the aim of assembling genomes and attempting reconstructions, it is also important to assess the richness of a community’s diversity in an environment (if possible, before the assembly stage itself is attempted). This characteristic may be measured by reconstructing the phylogeny of the sample, generally through the processing of 16S ribosomal Ribonucleic acid (RNA), which is a component of both prokaryotes (a component of the 30S small subunit of prokaryotic ribosomes) and eukaryotes (a component that in humans is encoded by the MT-RNR2 gene). For microbial communities specifically, there is a noteworthy computational tool called MetaPhlAn (Metagenomic Phylogenetic Analysis) [15], which profiles the compositions of communities originating from metagenomic shotgun sequencing information. This tool has species-level resolution for archaeal and bacterial organisms, having performed an extensive validation of profiling accuracy on several data sets and on thousands of real metagenomes. MetaPhlAn is relevant from a biological point of view because it is not limited to the use of 16S ribosomal RNA. Essentially, each read is assigned to the most plausible microbial lineage, often achieving a resolution greater than that obtained by profiling the universal 16S rRNA marker gene alone. It maps reads against a set of clade-specific marker sequences which are computationally pre-selected from coding sequences that unequivocally identify specific microbial clades at the species or higher taxonomic level.

In the task of taxonomic classification, the analysis of metagenomic data sets with considerable size is typically called binning. The reads originating from different organisms are associated with different groups of phylogeny, according to their taxonomic origins. There are two categories of methods for binning [5]:

- **Similarity**: these methods perform searches in databases to classify deoxyribonucleic acid (DNA) sequences based on their homology. This method relies on databases based on sequences of known origin and gene function. Presently, most common databases are biased toward model organisms or readily cultivable microorganisms, which diminishes the application of this method to microbial communities on ecosystems.
• Composition: binning by composition works by extracting intrinsic features from the sequence, such as GC-content, codon usage which are then compared with reference sequences whose taxonomies are already-known.

There are quite a few tools in the literature that are associated with taxonomy, homology-based tasks and other types of information processing, such as gene annotation. The most relevant to the task of annotation are, among others, Blast2GO, BG7, MEGAN and MG-RAST. Among these tools, it is not uncommon to encounter packages that make use of specific organic information databases, such as:

• GO: as an effort to standardize the representation of gene information across species and certain databases, a bioinformatics initiative was started, which is the Gene Ontology project [16]. It provides a vocabulary of terms used in the description of the characteristics of gene products and their associated annotation data. Tools to manipulate and process this data were also created and provided as part of the project.

• InterPro: it is a functional analysis resource for protein sequences [17]. It works by classifying proteins into families and predicting the presence of domains and important sites. InterPro makes use of predictive models (signatures), relying on the use of other databases (referred to by the authors as member databases). Its selling point is the fact that it aggregates different databases, decreasing redundancy and capturing the strength of the various databases.

• KEGG: the Kyoto Encyclopedia of Genes and Genomes [18] is a database integrating chemical, genomic and systemic functional information resources. Specifically, a link is established between gene catalogs from completely sequenced genomes and higher-level systemic functions of the cell, the organism and the ecosystem. Through the organization of captured empirical knowledge in computable forms, there has been an effort to manually create a knowledge base for these systemic functions. As a consequence of these efforts, KEGG is widely used as a knowledge base of reference regarding the integration and interpretation of large-scale data sets generated by genome sequencing and other high-throughput experimental technologies.

• SEED: a project started by collective effort in 2003, its focus being the development of a comparative genomics environment (SEED) and development of curated genomic data. This curation of genomic data is performed by expert annotators on subsystems, not on a gene-by-gene basis but across multiple genomes [19]. The curated subsystems are then used to extract freely-available protein families, which are typically called FIGfams in the literature. These families are central to the activity of certain packages, such as the RAST automated annotation technology.

Scientific classification in biology is used to group and categorize organisms. Typically, programs which aim to perform scientific taxonomy attempt to classify organisms with varying degrees of rank (Operational Taxonomic Units (OTUs)). For this purpose, Figure 2.1 is presented to display and keep in mind the major taxonomic ranks in the context of this project and task.

There are many candidates for this task, among which the following were particularly considered for the job:
• Genometa: a program which may be run on a personal machine as a Java application or remotely on the authors’ webserver [20]. It was developed as a set of modifications applied to the Integrated Genome Browser (IGB) [21] genome browser, whose functionalities are used to display information and results. Its functionality is based on a metagenomic reference sequence that resulted from the concatenation of 1190 prokaryotic chromosomes from various sources (e.g. the NCBI RefSeq collection, the Human Genome Project, among others). The procedure is then described by the authors as a mapping of query sequences against this metagenomic reference sequence in a fashion alike that of traditional genome resequencing. Accuracy and performance testing was done by the authors on a simulated ocean metagenome dataset created with Metasim using a hundred thousand reads from ten marine strains. The task of mapping in this tool makes use of the Bowtie mapper [22].

• MetaPhlAn: another program that may be run on the authors’ online server as well as other computers by configuration of local installations. Instead of the typical use of 16S rRNA, it uses an approach based on clade-specific marker genes of bacterial and archaeal phylogenies [15]. By the authors’ definition, the clade-specific markers correspond to coding sequences which are conserved within the clade’s genomes and not possessing substantial local similarity with any sequence outside the clade. Such a definition is sufficient considering that clades are groups of genomes (groups which can be as specific as species or as general as phyla) whose sequences are available (through resources such as the NCBI or the IMG\textsuperscript{1}), and the authors demonstrated the effectiveness of the definition even when coverage of reference genomes is only partial. A set of clade-specific marker genomes is constructed from these available reference genomes and when a metagenomic sample is processed by MetaPhlAn, its sequences are mapped against the set.

\footnote{\textsuperscript{1}Integrated Microbial Genomes and Metagenomes, whose website can be accessed at \url{http://img.jgi.doe.gov/}.}
• MEGAN4: a taxonomic analysis program, which was developed with the aim of integrating the taxonomic and functional analysis of metagenomic, metatranscriptomic, metaproteomic and rRNA data [23]. It performs functional analysis of environmental sequencing data sets, using both the SEED classification of functional roles and the KEGG for classifying enzymes and pathways. The program provides mechanisms to view taxonomy or functional content of the several types of data. It is multi-threaded, able to perform separate calculations concurrently, and is also able to use a PostgreSQL database which may be local or remote.

2.5 Gene Identification and Annotations

For a cell or an organism, its genome is its total complement of genes. Genes are important as they encode enzymes and proteins, which are essential to a plethora of functions within organisms, like catalyzing metabolic reactions, transporting molecules and replication of DNA, as well as others. As a consequence, retrieving and inferring upon the genetic configuration of a metagenome is extremely important, as it provides insight on the main chemical reactions that occur in a sample’s community as well as their potential impacts on the sample’s environment. Building upon the variants of assembly detailed previously, two major forms of obtaining information can be considered:

- Functional annotation: associates the identified genomic information of the sample with biological properties such as biochemistry and expression, by making use of algorithms such as Basic Local Alignment Search Tool (BLAST). It serves the purpose of classifying sequences (which are often protein-coding) in order to describe biological and enzymatic functionality, among other properties.

- Gene prediction: typically pertains to combining information from different measurements of contents and signals (sometimes also called ab initio gene finding). More refined methods for this form of obtaining genetic information use complex probabilistic methods such as HMMs.

In functional annotation, several programs were analyzed. These tools use web services but were designed with modularity in mind, in order to allow for the implementation of web service infrastructure on local machines. This is quite important, as it allows for the reduction of communication bottlenecks in most cases, provided that enough local computing resources are available. The tools are the following:

- Blast2GO: a tool created for all-in-one functional annotation of (novel) sequences and annotated data analysis. It features both a graphical user interface and a pipeline-oriented terminal mode. Blast2GO supports several databases, such as Gene Ontology (GO) [16], Enzyme Codes, InterPro [17] and KEGG [18]. Access by the main program to these databases may be done (default mode) online to known sites, or it may be done locally. This requires that the desired databases be properly configured using MySQL and with accurate copies of information that would otherwise be accessible on the default online repositories. To maximize performance, local access is to be given preference over the default mode. Blast2GO performs functional annotation in three steps. First, BLAST is used to find homologous sequences. Then, a mapping phase retrieves GO terms, and lastly, annotation is performed to select reliable functions. It should be noticed that this annotator may make use of a parallel version of BLAST. This may be achieved by breaking down a given
file in smaller pieces and running in parallel. The outputs of the BLAST instances, in Extensible Markup Language (XML) format, shall then be joined together in a single file which will then be provided to Blast2GO.

- GOblet: a free web service initially created to annotate sequences with GO terms and later extended to perform integration of pathway annotations [24]. It is also available as an application to be used on local machines for high-throughput processing. It annotates nucleotide and protein sequences with sequence comparisons based on BLAST, using the UniProt web server’s proteins with annotations based on GO. GOblet’s authors mention development effort was being channeled to produce a SOAP-web service to allow for integration into tools like Taverna, capable of connecting web services. Although this was stated, no version with such capabilities was found during MetaGen-FRAME’s development cycle.

- MG-RAST: useful for the task of taxonomy (and also metabolism). It contributes by creating metabolic maps [25, 26]. It is a server that functions as an automated metagenomic analysis platform. It operates on user-submitted metagenomic sequence data, for which it generates insight into the microbial population contained within. This approach produces quantitative results, allowing for a higher-level comparison of samples using the models derived from them. MG-RAST’s procedure uses the following execution script:
  
  - Normalization, which consists of removing duplicate sequences.
  
  - Search of protein-encoding genes (PEGs), by the use of blastx on several databases. These queries are performed in parallel on databases such as GREENGENES [27], RDP-II [28] and Silva [29]. Specific search criteria are used for each accessed database (for example, blastn is applied to rRNA databases).

  - Once this stage is done, the platform is able to create a phylogenetic reconstruction, by use of phylogenetic information of a non-redundant (NR) database and similarities to the rRNA database. Based on the results of the similarity searches, functional annotation of the PEGs is executed by projecting the results against SEED [30, 31], FIGfams and subsystems [30]. The functional annotations are then used as input to the automated metabolic reconstruction phase. Metabolic reconstruction employs the subsystem-based dataset. As mentioned earlier in this document, metagenomics has the potential to shed light on the critical biochemical mechanisms in each environment. The functional roles of the previous stage are mapped to reactions in the SEED database, which then are used to create a model of existing reactions for the sample being analyzed.

  - With the information extracted in the previous steps, the platform’s functionality for genomic comparison may be employed, with the following variations: comparison between analyzed metagenomes (heat map), which uses annotated metagenome sequences as queries, showcasing the relative abundance of metabolic reactions, calculated as the number of sequences per subsystem/tax class as a fraction of the total sequences in a data set; comparison between the reconstruction of a metagenome and a genome, through the recruitment plot functionality. This uses the same query elements as the heat map. The platform chooses a genome as a scaffold against which it attempts to map sequences derived from the studied metagenome. It is in fact a metabolic comparison of a predicted organism of the metagenome;
hierarchical comparison using KEGG maps [18]. The mapping of functional roles (obtained from analysis against the SEED) to KEGG maps is performed in a way that they are hierarchical, alike the subsystems. This allows for browsing in different levels of the sample or to compare it with other metagenomes.

In the scope of gene structure prediction, different options were considered:

- **BG7**: an annotator focused exclusively on prokaryotes, relying on the principle of using as much information as possible. It was designed in a way that attempts to use the capabilities of recent forms of computing such as cloud computing. This may be particularly noted by the fact that the authors provide a description on how to use it with the Amazon Cloud. Alike Blast2GO, it may also be configured for local use. BG7 employs a sequence of steps:
  - Firstly, the user provides a set of reference proteins which shall be searched for in the genomic sequence that is to be annotated (tblastn is used in this case). This shall produce a lot of High-Scoring Segment Pair (HSP) for each hit.
  - Each hit’s HSPs shall be merged to define a unique similarity region between a protein and the contig in question. After the merges, start and stop signals are searched for, both upstream and downstream, attempting to define in a preliminary way CDS genes. However, these CDS genes may suffer from malformations such as non-canonical start and stop codons, intragenic stop codons and possible frame shifts. Rather than simply discarding these genes, they will be annotated as having these deficiencies.
  - Then, comes the phase of conflict resolution. In the case that two or more proteins map to the same region of a contig or that overlapping genes are detected, action must be taken. For conflicting proteins, the gene of the protein most similar to the contig is chosen, and the remaining genes are marked as “dismissed genes”. For genes that have a percentage of overlap greater than a specific user-defined threshold, the same tie-breaking procedure is used.
  - Once the genes are well defined (limited overlapping and each gene being annotated by a single protein), an RNA gene search is performed. Search is performed using blastn on the contigs and the RNA genes are identified in the same way as the protein genes previously were. Finally, RNA and protein genes are then integrated in such a way that an RNA gene is always given preference over a protein gene.

- **HMMER**: a software tool created for making protein sequence alignments and searching sequence databases for homolog protein sequences\(^1\). It uses probabilistic models called profile Hidden Markov Models, achieving performance and accuracy figures [32] very close to that of NCBI BLAST. After several iterations of the software with additional improvements, a web server version [33] was made public with web services available, allowing for the integration of HMMER in scripted workflows such as Taverna.

- **Maker**: a genome annotation pipeline [34], whose purpose is to facilitate the annotation of eukaryotic genomes by researchers, allowing for the creation of genome databases. It identifies repeats,
aligns Expressed Sequence Tag (EST)s and proteins to a genome, is able to generate ab initio gene predictions and also synthesizes the data into gene annotations, having evidence-based quality indexes. With the prospect of making this pipeline easy-to-use, the authors considered and underlined key-constraints in the design of the pipeline that must be supported. In between other requirements, it was intended that it would be possible to configure it with minimal bioinformatics and computer resources, that it would be able to encapsulate the results of its capabilities into a feature-rich gene annotation. Also, given the uniqueness of every genome, the pipeline would also have to be easily configurable and trainable. Maker uses BLAST internally, and all of its parameters may be configured. It also uses GeneMark internally for gene prediction.

2.6 Assemblies: What do They Represent?

An assembly of a set of sequences represents a reconstruction of the target (or targets in the case of a metagenome) from which the set was derived by means of sequencing technologies. To perform an assembly, an analysis of the target sequence data must be done. There is a variety of algorithms that may be employed for this purpose [35]. Several studies on this field highlight two major approaches to assembly, the comparative/resequencing method and the de novo/ab initio type of approach.

The comparative/resequencing approach, also called templated assembly, is based on mapping algorithms and the concept of aligning sequences to reference genomes. As the number of sequenced genomes has grown, so did the count of organisms that have two or more closely-related and already-sequenced species [36]. This enabled the development and use of such algorithms, performing comparison and alignment of newly sequenced genomes with a known reference genome. Among these algorithms, one may consider those more traditional and well-established, like the FAST All algorithm [37], the BLAST [38] (with its DNA and protein query frame variation and encoding variants), the Smith-Waterman [39] local alignment algorithm, as well as more modern methods such as the Burrows-Wheeler Transform [40] (it is relevant to mention that this algorithm is typically applied to single-genome reconstruction). Some comparative platforms (such as MG-RAST) for metagenome assembly integrate in their pipelines a module based on the Burrows-Wheeler Transform for comparative single-genome assembly, which is then applied with very specific purposes, such as comparing the sampled reads to the genome of a host organism (in case one is present) to detect potential cases of host contamination [26]. The results from these comparisons are used to infer the structure with which to reconstruct the target. The previous approach falls short when there aren't readily available references to use as a basis for comparison. As a consequence, it is also necessary to develop methods for assembly that attempt to identify information of the sample's elements and relationships between the query reads in terms of the sample itself. This approach is named de novo/ab initio and formally, it falls in the category of NP-hard (Non-deterministic Polynomial-time hard) problems, a set of problems for which no efficient computational solution is known [41]. One instance of this is the classical Greedy Algorithm [41]. This method attempts to iteratively join together the best overlapping reads to form contigs. Like many greedy solutions for varied problems, this method tries to join reads in a locally-optimal form, which may compromise the quality of the global assembly. Another example of this type of approach is the Overlap-Layout-Consensus [41]. As the name implies, this is divided into three phases:
• **Overlap:** compares reads against each other to detect which reads overlap with which. Usually, this information serves to construct an overlap graph, where each node represents a read and an edge connects two nodes if an overlap was detected between their corresponding reads;

• **Layout:** establishes a precise positioning of the reads regarding one another. This is done by identifying, in the overlap graph, paths that correspond to segments of the genome being assembled. This graph problem consists in a Hamiltonian Path problem, which falls in the NP-hard category of problems. The main objective is to identify a path that traverses all nodes of the overlap graph exactly once, which represents a reconstruction of the target using all of its reads.

• **Consensus:** from the relative positioning of the reads on the chosen path, determine the sequence of DNA that is implied.

There is also the method of the Eulerian Path approach, making use of de Bruijn graphs [42]. It begins by computing a k-mer spectrum of the sample, noting that each k-mer that originates from the set of reads will correspond to an edge in the de Bruijn graph. The nodes represent the suffixes and prefixes with length $k - 1$ of the originally computed k-mers. Since all the k-mers will be present as edges in the graph, the problem is now one of finding a path through the graph that passes through each edge exactly once. Efficient algorithms for Eulerian Path problems exist, and despite possibly returning more than one solution, the desired result may be obtained by providing additional guidance to algorithms that operate on this data structure [43]. Typically, these methods of guidance include mechanisms that attempt to simplify the graph. They include operations such as: trimming dead-end edges in the graph, which originated from errors at the end of reads, correcting errors in the middle of the reads (“popping bubbles”) and clipping edges that connect low-coverage nodes (chimeric nodes). This last part of detection of chimeric nodes is particularly important, as they result from the fact that two species contain a common or very similar sub-sequence in their genomes, or that a single species’ genome contains repetitions of a sub-sequence. Sequencing by hybridization technologies [44] inspired the use of the Eulerian Path in the context of these biological problems.

### 2.7 Alignment/Resequeencing

Many of the existing aligners mentioned in this section make use of the Burrows-Wheeler Transform (BWT) [40]. It is an algorithm used in techniques for data compression, employed in bioinformatics by several more recent alignment programs. This algorithm receives a string $S$ as input and produces a permutation of its characters, in a way that if the input string had certain substrings with higher frequencies, the result of the transformation will have several regions where an individual character is repeated several times in a row. It is a reversible algorithm, in the sense that the results obtained by its execution may be processed to re-generate the original input that was used. It has a time and space complexity that is $O(n)$ in the length $n$ of the input string provided. The algorithm performs $n$ cyclic rotations of the input string, sorts them in lexicographical order and then extracts the last character of each cycle-based permutation of the string, forming the output string $L$ composed by those characters. The $i$-th character of $L$ is the last character of the $i$-th sorted rotation. The following quote from the authors’ work [40] details their initial hypothesis of applying string $L$ in compression algorithms:
"The sorting operation brings together rotations with the same initial characters. Since the initial characters of the rotations are adjacent to the final characters, consecutive characters in L are adjacent to similar strings in S. If the context of a character is a good predictor for the character, L will be easy to compress with a simple locally-adaptive compression algorithm."

This algorithm, alike others that manipulate strings and suffixes (such as the suffix array or suffix tree), usually defines a special terminating character that does not occur anywhere in the input string S and is considered lexicographically inferior to all other characters, being placed at the end of S. In the literature (and as can be verified in the figures of this document), there is a tendency to designate the character $ for this purpose. This document includes examples of this algorithm in action: Fig. 2.2 illustrates the initial suffix distribution of the input string, Fig. 2.3 contains the output of the algorithm and Fig. 2.4, 2.5 and 2.6 showcase the series of steps taken to process a word query with this algorithm.

![Figure 2.2: Burrows-Wheeler Transform: suffixes of string $S=GTGATAACGATATAC$ in a suffix array, shown in increasing index order.](image)

Among the aligners and mappers seen, we highlight the following:

- **Bowtie**: a fast and memory-efficient program [22] for aligning short DNA sequence reads against large genomes. It employs a Burrows-Wheeler index based on the full-text minute-space (FM) index [45, 46], providing a considerably small memory (RAM) footprint, such as a consumption of about 1.3 GB for the whole human genome. This in turn allows Bowtie to run in most desktop machines. Bowtie is an aligner that also makes use of multiple processor cores to run several threads to increase performance.

- **BWA - Burrows-Wheeler Alignment tool**: another fast and accurate short read aligner [47] that makes use of the Burrows-Wheeler Transform. It is able to perform gapped alignment for single
end reads and paired-end mapping. It is also capable of generating multiple hits if required and generates mapping quality. The default alignment format for BWA’s output is Sequence Alignment/Map (SAM) [48].

• BWA-SW, Burrows-Wheeler Aligner’s Smith-Waterman Alignment: also the authors of BWA. This is a version [49] of the algorithm which was proposed with longer read lengths in mind (in the range of 1 Mb) whilst attempting to minimize memory consumption. It is important to note that long-read alignment differs from short-read alignment regarding the objectives. In the short case, spaced seed templates may be designed, spanning the entire read. Poor matches may also be filtered out by using, for example, q-gram filtration [49]. This is done because full-length read alignment is preferred. In contrast, when dealing with the alignment of longer reads, local matching is preferred, as this type of read is more prone to suffer structural variations and misassemblies in the reference.

• SOAP2 (Short Oligonucleotide Alignment Program): is a short read aligner [50] that uses BWT like previously described aligners. It is an aligner compatible with both single-end and paired-end reads, tested on the whole human genome with a peak memory consumption of around 5.7 GB. SOAP2 was designed for very fast alignment in large-scale resequencing projects.

• TAPyR - Tool for the Alignment of Pyrosequencing reads: an aligner which has a particular focus on the alignment of pyrosequencing reads, like the ones generated by the 454 GS FLX platform. This tool proved to be more efficient, in terms of both performance and result quality, than other mainstream applications included in this section [3]. TAPyR relies heavily on data compression techniques, using an implementation of the FM index [45, 46] optimized for the DNA alphabet. TAPyR uses a seed-based heuristic that explores the characteristics of pyrosequencing data.
Figure 2.4: Burrows-Wheeler Transform: resolving the first step of query word $Q$ on the top-left side.

Figure 2.5: Burrows-Wheeler Transform: resolving the second step of query word $Q$.

### 2.8 De Novo Assembly

For this task, a brief inspection of existing software names in the literature showed that most solutions have heavy hardware requirements, particularly in the aspect of memory consumption. This is specially true of the assemblers which use de Bruijn graphs. Next is a list of the most revered programs for this type of assembly:

- **SOAPdenovo**: a tool devised for the assembly of very short reads [51], with lengths that make overlap-layout strategies unfeasible. Since there is only a very short sequence overlap between reads of this type, it becomes difficult to distinguish between cases where correct assembly is attained and situations of repetitive sequence overlap. The nature of these reads presents reasons that make it impractical to represent in memory all of the sequence overlap information. SOAPdenovo therefore employs the de Bruijn graph data structure.
• Newbler: an assembler designed with intention to be used with 454 data de novo. The assembler also supports Standard Flowgram Format (SFF) files generated by the Ion Torrent platform. It is an OLC-based assembler which actually makes the use of this technique two-fold. The first application of OLC generates unitigs, which function as preliminary, high-confidence conservative contigs, for seeding the rest of the assembly pipeline. The second run of OLC joins the unitigs into a contig layout based on pair-wise overlaps between unitigs. Unitigs whose suffix and prefix align to different contigs may end up being split (these reads may have been chimera reads or the result of a repeat boundary) [52].

• Velvet: a set of algorithms [53] which makes use of the de Bruijn graphs. As a software package, it is implemented in C and is freely available under the GNU Public License. It is ideal for data sets comprised of very short reads (roughly 25-50 bp) with high coverage. The authors of Velvet, testing the algorithms on short reads and paired-ends information exclusively, were able to achieve values of N50 (a statistic detailed in the glossary of this document) around 50 kb with prokaryotic simulations and of 3 kb for simulated mammalian Bacterial Artificial Chromosome (BAC)s. Applied to real Solexa data sets without read pairs, contigs of approximately 8 kb were obtained for prokaryotes and 2 kb for a mammalian BAC.

• MetaVelvet: this tool's authors remark that most pipelines use single-genome assemblers with carefully-chosen parameters. It is an extension [43] of Velvet for metagenomic assembly, relying in the use of de Bruijn graphs. It first builds such a graph from the mixed short reads of the sample. Then, MetaVelvet applies the concept of decomposing the graph into sub-graphs and building a scaffold based on each sub-graph, as an isolate species genome.

• Meta-IDBA: a de novo metagenome assembler [54], designed to solve the metagenome assembly problem caused by polymorphisms in similar species in metagenomic environments.
Chapter 3

MetaGen-FRAME Architecture

As a result of the first months of work, a draft of the top-level composition of the framework’s architecture was proposed. The development process was then continually exposed to an influx of information, providing insight on the set of candidate programs and their integration as framework modules. As a result, both the framework’s structure and its implementation were under constant review in an iterative fashion. This section is therefore divided in three subsections. The first showcases the final iteration of the framework’s architecture. The second subsection is dedicated to specificities and additional properties of the integrated bioinformatics task modules (further technological questions not limited to the architecture of the framework are covered in section 4).

3.1 Final Revision

The project was initially steered towards facilitating the visualization of data results and custom configuration of the framework. Consequently, there was a very important phase associated with selecting software tools that have the best leaning to this purpose. Due to time constraints, a formal ranking system with a concrete set of values for comparing the candidate software tools was not defined. Instead, they were analyzed with respect to some more common functionalities for the purpose of identifying the most widespread input/output formats. The careful consideration of the software was performed not only for each bioinformatics task but also for the software serving as infrastructure. An infrastructure software supporting the proper functioning of other programs and serving as the backbone of the flow of information between the targeted bioinformatics tasks was desirable. Such a flow was made possible by implementing the skeleton of MetaGen-FRAME using the Workflow Management System Taverna, an open-source tool suite created to aid in the design and execution of scientific experiments (described with greater depth in section 4).

The final top-level representation of the framework’s bioinformatics tasks is depicted in Figure 3.7. In that illustration, the tasks Data Quality Control and Alignment were ultimately implemented with the tools previously mentioned as potential candidates for those roles. Respectively, that first makes use
of NGS QC Toolkit (initially considered as the strongest candidate) and the later employs the now-implemented parallel version of TAPyR, dubbed P-TAPyR. Herein are clarified the remaining bioinformatics tasks (visible in both the preliminary and updated depiction of the framework), part of the second stage of execution of the framework, and the choice of tool made for each:

- **Taxonomic Analysis** - MetaPhlAn: for this task, MetaPhlAn\(^1\) (clade-specific marker genes against which reads are mapped) was elected over its only main contender, which was Genometa (traditional database set of reference sequences). Genometa was considered superior to MEGAN as a contender to MetaPhlAn because MEGAN’s strict use of BLAST makes it much more computationally heavier than Genometa with its Bowtie-based approach to alignment [15]. The discarded candidate had the disadvantage of being unable to operate without the interaction of a user, something that is prohibitive in the desired larger scenario of automatically executing a set of steps in the framework (at least such functionality was amiss in Genometa’s documentation). Furthermore, although Genometa attempts to assign a taxonomic label to every sequence, which could yield higher-resolution results (as opposed to MetaPhlAn and its small marker library which allows for more scalability but without attempting to label every single sequence), it relies on a database older than that of MetaPhlAn (Genometa’s database was created in 2010 while MetaPhlAn’s was in mid 2012 [55]). Another reason for choosing MetaPhlAn is the fact that it produces results in a formatted text file with a layout simpler than that of other programs.

- **Functional Annotation** - Blast2GO: this software was the one chosen for the task at hand, picked over other candidates such as MG-RAST. It’s built in a modular way, providing both a functional Graphical User Interface (GUI)-based version and a terminal-based one. Despite the connotation of the terminal version being for advanced users (as per the authors’ remarks on their website), it is is able to produce the same sort of results that the graphical version does (with the same user-friendliness), making it a very desirable software to integrate in MetaGen-FRAME for automatic Functional Annotation. Similarly to the choice of MetaPhlAn for Taxonomic Analysis, the software Blast2GO produces results in a user-friendly way. It produces a set of images which are then referenced in a report in an HTML page, a format which Taverna supports as a choice for presenting results by launching it on the default system browser by request. Another decisive factor is associated with the way Blast2GO defines access to resources such as BLAST. The program executes BLAST on the set of sequences given to it as input, but it is also capable of importing the results of an external BLAST execution in XML format (like the one the latest NCBI BLAST C++ version [56] is capable of producing), which opens the door for parallel BLAST execution (covered in section 4). Such a possibility is of extreme relevance, because the program (blastx) is very slow (and heavy on memory consumption): it took approximately two and a half days for 3769 sequences with average length of 1145 nucleotides to run the program divided into eight parallel blastx processes against a local\(^2\) copy of the NCBI non-redundant database\(^3\) of protein sequences, reaching a peak memory usage of 110 GB and an average of 13.5 GB per process.

\(^1\)Its results were published in the *Nature Methods* science methodology journal, which had an impact factor of 23.565 for the year 2012 according to Thomson Reuters.

\(^2\)The computing facility where this and other programs of the framework run is described in section 3.1

\(^3\)Containing entries from GenPept, Swissprot, PIR, PDF, PDB and NCBI RefSeq. Please visit ftp://ftp.ncbi.nlm.nih.gov/blast/db/README for more information regarding the most common databases.
This flexibility is a huge advantage over other tools which have a statically-configured BLAST integration, which allow neither parallel execution and its gains neither local database copies to avoid access to the NCBI servers through the Internet. MG-RAST’s lack of a proper web service access to its functionality was the main deterrent of its use in the framework. Additionally, MG-RAST job submissions take a considerable amount of time (several days for the single complete genome *P. falciparum 3D7 PlasmoDB rel 7.0*) to complete, which is due to the fact that its computing resources are shared between all job requests. There is a way to achieve higher execution priority in MG-RAST, but such possibility requires that the submitted data be published, which prohibits the platform’s optimal usage when confidentiality is required. Regarding the GOblet candidate, its functionality is similar to Blast2GO, however, it’s not as resource-rich as the later (e.g. in terms of databases used), which contributes negatively to its desirability. Blast2GO was therefore the chosen program for this task.

- **De Novo Branch:**

  - **De Novo Assembly - MetaVelvet:** for this task, various options were encountered: MetaVelvet, Newbler and SOAPdenovo, among others. SOAPdenovo was disregarded as it is admittedly a single-genome *de novo* assembler. MetaVelvet’s authors benchmarked [43] their software (using both simulated and real metagenomic data sets) against the metagenomic assembler Meta-IDBA and the single-genome assemblers SOAPdenovo and Velvet (the assembler that was extended into MetaVelvet). In their tests, four data sets with different levels of taxonomic resolution were used: ‘order’, ‘family’, ‘genus’ and ‘species’ (mentioned in order of decreasing taxonomic level). With a varying level of resolution, the amount of *k*-mer subsequences shared between genomes of a given resolution also varies. As a consequence, the lower the taxonomic level, the higher the percentage of *k*-mer subsequences shared, which directly translates to harder input sequencing reads separation and de Bruijn graph decomposition. In the results shown by the authors of MetaVelvet, their software assembled the metagenomic read data with values of the N50 (defined in the glossary) statistic greater than those of other assemblers. Longer N50 scores benefit the identification of protein-coding genes, which was demonstrated by the fact that MetaVelvet managed to identify more protein-coding genes at all taxonomic levels than the other assemblers. Because of the details mentioned above, it was decided that MetaVelvet would be used for this task.

  - **Gene Structure Prediction - BG7:** the last task to be implemented in the framework. BG7 is a genome annotation system designed with an exclusive focus of prokaryotes in mind. The authors of BG7 identify the structure of classical annotation systems as being composed on two separate phases: ORF prediction (efficient for when the underlying sequencing technology of the data has minimal errors) and then functional annotation. Although with varying characteristics, NGS sequencing technologies produce indels, which are detrimental to the first phase of ORF prediction. They claim that any gene lost in the first phase cannot be recovered in later phases of annotation. BG7 answers these problems with a paradigm integrating gene finding and annotation in a single step supported by massive analysis of similarity with reference proteins. This software, performing automated annotation, was able to produce annotations quite similar to NCBI annotations which were done by a large community of researchers, which was another good point in favor of BG7.
CHAPTER 3. METAGEN-FRAME ARCHITECTURE

3.2 Additional Properties

As previously stated, the interconnections between software packages make use of the Taverna Workflow Management System. Taverna is interfaced with bioinformatics software tools using Python. To enable the replacement of specific bioinformatics task modules without touching the rest of the framework, it was necessary to define and choose a set of internal standard formats for the inputs and outputs of several bioinformatics tasks. In the following paragraphs, the bridging is made between the biological composition of the framework and its concrete implementation in software. Namely, the information flowing between bioinformatics tasks will be clarified.

The list of descriptions that follow are very important, as they should be consulted by any third party wishing to replace any software tool implemented by default in the framework. The descriptions attempt to be as independent as possible of certain technical peculiarities consequent of using Taverna (although including specific Taverna-supported tasks which are needed, such as Remote WS):

- In a high-level perspective, MetaGen-FRAME expects a set of arguments, which are the path to a file with sequences, in FASTA or FASTQ format and a threshold value with which to guide the bioinformatics task of taxonomy (and concretely with the use of Taverna, a string which represents the file type of the sequence file format).
• **Data Quality Control:** the software used for this task must expect input for this task as a sequence file in FASTA/FASTQ format. The output produced is expected to be a cleaned sequence file in FASTQ format. The integrated tool for this task is mentioned in section 2.3.

• **Taxonomic Analysis:** for this task the software must expect an input file of type FASTA/FASTQ and the output expected of it is a tab-separated values file\(^1\) where the first line is the set of taxonomic ranks shown in Fig. 2.1 in increasing order of specificity (From the kingdom rank all the way to the species rank, separated by tabs) and ending with an identity column. Each line that follows the first represents an identified taxon, hopefully a rank as specific as possible. These lines following the first are taxonomic rank names separated by tabs (and spacing inside taxon names should be represented by the plus character: +) and the last element of each line must be a value which is the percentage of the total amount of the sample that corresponds to this line's taxon. To highlight this explanation, Fig. 3.8 shows this result format of taxonomy applied to the data obtained by executing MetaGen-FRAME on a *P. falciparum* strain.

• **Remote WS:** this task accepts an input file with the format Taxonomic Analysis has for its output. It invokes several web services to query (at the time of writing) the NCBI servers in order to retrieve one genome (nucleotide sequences) for each strain (in NCBI) closest to the rank names obtained in the previous task. The output of Remote WS is divided in a direct output and an indirect output. Its indirect output is manifested by the fact that for each strain genome retrieved, a FASTA file with the corresponding sequence (or sequences in the case of incomplete projects found on NCBI) is stored locally on the machine executing the Taverna component of MetaGen-FRAME. The direct output is a list of these local files. Thus, it is a simple text file where each line is the absolute system path of each genome file retrieved. A sample of this output format is presented in Fig. 3.9, for the same strain mentioned in the previous task.

• **Alignment:** as an input, this task expects the simple text file path list produced as output in the previous task. Again, the outputs produced are a set of indirect and direct files. The indirect group contains, for each file path of a reference FASTA file: the resulting consensus-generated contig sequences in a FASTA file; a second FASTA file containing the sequences which did not align against anything. As a consequence, the total size of the indirect output file set should be the number of file paths in the input file received times two. The direct group of outputs consists of two simple text files, one having a list of paths to the generated unaligned sequence files, while the other lists the paths to the contig sequence files. These simple text files have a structure similar to that of Fig. 3.9

• **Functional Annotation:** the expected input for this task comes from the previous one, as it is the text file with a list of paths to the contig sequence FASTA files. Currently, the output of this task is considered a direct output of MetaGen-FRAME and its format is specified on the basis of user-specific criteria\(^2\). In the current implementation, the bioinformatics task of Functional Annotation produces a top-level HTML file which presents the results for each contig sequence FASTA file (each individual result is itself an HTML report in the case of the default tool for this task: Blast2GO).


\(^2\)This apparently-confusing use of a configurable format is further clarified in section 4.
• **De Novo Branch** consists of a sequence of two bioinformatics tasks, depicted in Fig. 3.7:

- **De Novo Assembly**: a single input is expected, namely the simple text file containing a list of paths to the unaligned sequence files generated during the procedure of **Alignment**. It produces two sets of outputs: one is a set of contig sequence files generated through *de novo* assembly and the other is, as previous tasks, a simple text file containing a list of paths to these generated assembly contigs.

- **Gene Structure Prediction**: this task receives as input the set of assembly contig files generated in the previous task, as well as the simple text file listing those same files in the local machine. The output of this task, alike that of **Functional Annotation**, will depend on user-specific criteria and is directly connected to the framework’s execution outputs.

---

Table 3.2: An overview of the taxonomy hierarchy. Kingdom, Domain, Phylum, Class, Order, Family, Genus, Species are displayed for the taxon ‘*Escherichia coli*’.

<table>
<thead>
<tr>
<th>Kingdom</th>
<th>Domain</th>
<th>Phylum</th>
<th>Class</th>
<th>Order</th>
<th>Family</th>
<th>Genus</th>
<th>Species</th>
<th>Identity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bacteria</td>
<td>Proteobacteria</td>
<td>Betaproteobacteria</td>
<td>Burkholderiales</td>
<td>Caulobacteraceae</td>
<td>Candidatus-Sinderia</td>
<td>Candidatus-Sinderia</td>
<td>76.0379</td>
<td></td>
</tr>
<tr>
<td>Bacteria</td>
<td>Proteobacteria</td>
<td>Gammaproteobacteria</td>
<td>Enterobacteriales</td>
<td>Enterobacteriaceae</td>
<td>Candidatus-Carnobacteriaceae</td>
<td>Candidatus-Carnobacteriaceae</td>
<td>13.9379</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.8: Example of the internal taxonomy standard to be output by the **Taxonomic Analysis** bioinformatics task. In this case, through the use of the taxonomic threshold input parameter of the framework, it was specified that only the two highest percentage identity detected ranks present would be shown.

```
/tmp/usecase5633555704218675473dir/307069503_ref.fasta
/tmp/usecase5633555704218675473dir/527135639_ref.fasta
```

Figure 3.9: Example of the internal strain reference path list standard to be output by the **Remote WS** task. The FASTA file names are of the form *X*_ref.fasta, where *X* is the NCBI GI number of the strain’s genome.

The flow of information between these tasks is supported by a combination of Taverna and Python functionality, allowing for an abstraction of the underlying software tools in use. This is further explained in section 4.
Chapter 4

Framework Implementation

In a previous section of this document, it was stated that a technical decision would be mandatory in order to implement the connection between software tools and the flow of generated data. To this end, the choice was ultimately made to implement MetaGen-FRAME using a two-level layer of software abstraction for the infrastructure underlying the bioinformatics tasks (and their selected tools). This is illustrated in Fig. 4.10. To enable the configuration of bioinformatics tasks, a separation was enforced between the concept of a bioinformatics task used in MetaGen-FRAME and the underlying software tool in use. This design decision manifests the necessity of providing auxiliary tools to convert between the input of a bioinformatics task and the software input, as well as from software to data inputs.

Figure 4.10: This scheme illustrates the way software tools are executed. The bioinformatics task composition is represented in Taverna (through the configuration file Skeleton.t2flow). Every concrete software tool implemented interfaces with the Taverna-based workflow (L1 in the figure) through the use of Python (L2 in the figure) scripts and a set of configuration files to be present in the local machine.
output to task output. This has the benefit of isolating the modifications required on the structure of the framework to a specific point, keeping the rest of the working bioinformatics tasks intact. Figure 4.11 exemplifies this sequence of steps.

Figure 4.11: The illustration serves the purpose of highlighting the approach of separating software tools from the bioinformatics concept they aim to satisfy.

4.1 Taverna - Workflow Management System: Major Highlights

In the initial stages of development, Taverna was recommended for the infrastructure of the framework due to its data visualization capabilities. These are desirable, as they ease the set of steps to be taken in order to view the results of execution. During the development of MetaGen-FRAME, confidence was placed on the idea that using Taverna and its GUI would be beneficial to users which don’t necessarily have extensive training in computer engineering and are not particularly computer-savvy. The overall view of the framework’s skeleton represented as a Taverna-based workflow is is shown in Fig. 4.13.

Taverna stores the results of its executions in directories it creates in the operating system’s location for temporary files of the file system. A program that is invoked by Taverna (in the context of Taverna’s terminology, a local tool service) will have its current working directory set to be one of such temporary directories (as illustrated in Fig. 4.12). Such behavior is problematic for the Taverna layer when invoking functionality from the Python layer, as Taverna does not have (to our knowledge) means by which to refer to the path of the framework unless the full path is directly introduced in the Skeleton.t2flow by directly configuring the workflow in Taverna’s interface. If such behavior was accepted, users would have to perform this configuration in Taverna’s GUI, which besides being cumbersome, would be potentially confusing, as different operating systems have different details, such as variation of the symbol used to represent directory separators in paths. To avoid this, instead of simply starting Taverna and opening the workflow built for the framework, one must invoke a Python run.py script which launches a Taverna instance with a special variable in the instance’s environment context. This variable’s value is the path to
MetaGen-FRAME. The framework's workflow has a service that on initiation of a new execution, checks for this variable's existence in order to pass along its value to services which invoke programs in the framework's directories. With this, all that the user needs to do to use the framework is launch the *Python* script `run.py` which exists in the root directory of the framework. As a last detail, that script assumes that in the top-level directory of the framework, there exists a file called `taverna_path.ini`, which, as the name implies, contains the path to Taverna in the file system. This file may be configured through the *Java*-based GUI application `metagenomic.framework.ist.utl.pt.ConfigTaverna`, which has basic functionality for browsing a file path and saving it (an example of it in use is shown in Fig. 4.14).

![Figure 4.12: Example: generated set of directories during an execution of the framework in Taverna.](image)

The functionality of the *Python* layer relies on its main entry point, which consists of the `config.py` script. Whenever Taverna is to perform one of the major (previously highlighted) bioinformatics tasks, the task’s corresponding tool service invokes `config.py` with appropriate parameters. This *Python* script then parses the arguments and chooses the appropriate course of action with the support of configuration files. The framework's directory contains specific directories associated with each major bioinformatics task (QC, Taxonomy, Alignment, Functional_Annotation, De_Novo_Assembly and its follow-up, the Gene_Prediction task). These subdirectories contain, in their turn, the instance of a template configuration file named `task_candidates.ini`, which is used solely for the purpose of binding a given specific software to each bioinformatics task.

Another important feature of Taverna is the web service functionality it has to access the possibilities offered by the NCBI. Mild local processing tasks (mainly for parsing the outputs of a web service to generate the input data for the next web service) are required in order to chain together a sequence of web service calls, starting from a list of taxonomy items and arriving at the situation where reference genome sequences are locally-available as FASTA files (detailed in the input/output description of *Remote WS* in section 3.2). The *Remote WS* task in Taverna is configured as a sequence of web services and illustrated in Fig. 4.15, with the executed web services being the following, in order:
Figure 4.13: Overall view of the Taverna-based workflow which defines MetaGen-FRAME.

- Submit_ESearch: receives a set of formatted NCBI search terms (which were obtained by proper formatting of the Taxonomic Analysis task output) corresponding to the identified target taxons. This service then receives a set of NCBI identifiers associated with the results found in the online databases.

- run_eLink_MS: receives the set of identifiers of organisms and database entities associated with the previous query terms. It then queries the NCBI for nucleotide database sequence identifiers related to these previous identifiers.

- run_eSummary_MS: uses the nucleotide identifiers obtained in the previous web service to query the NCBI for information about them and their summaries. The Parse_ Summaries beanshell script then looks into the obtained summaries for keywords that relate it with taxons obtained in the Taxonomic Analysis task and also containing the expression "complete genome". If a given identifier summary satisfies these criteria, it is added to a set of valid identifiers, to be output to the last web service, which will fetch the reference genome sequence nucleotides.

- Run_eFetch_MS: this web service gets the reference genome sequences associated to the identifiers (GIs) obtained through the previous web services. It then passes its output to a Python script called store_references.py which parses the results and finally stores the sequences as FASTA files, naming each file so it contains the GI of the reference it corresponds to. The purple
and green squares, respectively, perform input configuration (the web service formats are based on XML) and the actual web service invocation.

Figure 4.14: The Java-based application used to configure the `taverna_path.ini` file which is used to store a valid path to Taverna.

When Taverna displays a given bioinformatics task’s results (through its respective implementation of Tool Service’s output ports and the workflow’s output ports), there is a drop-down list which lets the user select how to interpret the results. To better separate program-specific output files and textual output and status messages, it was defined that each bioinformatics task would produce two major output files: one log file containing the output of the program’s `stdout` and another one, always called `view`. With this nomenclature and Taverna’s options for data display, a user may use the drop-down list to specify that a program’s `view` file should be interpreted as an image, (e. g. `.png`) triggering its internal display, or that the result’s format should be interpreted, for example, as an HTML page, which will trigger the launch of that page on the system’s default web browser. This is the type of flexibility encountered in Taverna which contributes to the potential of configuring a specific task in whatever way it may be desired, without imposing that it be bound to a given output type whatsoever (requiring only the nomenclature described earlier).

### 4.2 Python Layer

From an engineering point-of-view, the choice of using Python was made due to technical aspects that needed to be addressed. There were several functionalities that needed to be implemented and did not demand high-performance computing features of compiled languages such as C/C++. Using such languages would also contribute to an additional overhead in setting up MetaGen- FRAME, as it would require compilation of the modules implemented in those languages. Python was also employed as part of an effort to increase the portability of MetaGen- FRAME. A common version of it is available in most Linux distributions that are employed in research facilities (typically, the 2.7 series of the language, and its configuration is straightforward in Linux, OS X and Windows, including both 32-bit and 64-bit architectures).
CHAPTER 4. FRAMEWORK IMPLEMENTATION

Figure 4.15: The configuration of the web service composition used in the framework to obtain reference genomes from the online resources of the NCBI. The cyan squares represent constant values to be input to each processing task in this scheme. The brown ones are simple beanshell scripts used to convert data formats in order to bridge NCBI queries.

4.3 Repeated Sequence Removal

Since we decided (and to facilitate the chaining with the Data Quality Control task) to implement a repeated sequence removal functionality, it made sense to use the dict data structure of Python, which has an internal representation based on hash tables. This data structure contains slots which are conceptually equivalent to $\langle$Hash, Key, Value$\rangle$ tuples. Repetition removal is implemented in the collapse_sequence.py file, consisting of building a dictionary by progressively-inserting sequences as keys. The dict structure of Python, when used to perform an insertion, computes a Hash of the Key of a $\langle$Key, Value$\rangle$ pair that is to be inserted (in the context of this task, Value is always a sequence of the data set, more concretely, a Python list containing a sequence’s name, nucleotides and quality values). If Hash is found to already exist (with one or more copies), it means an internal collision has happened and, as a consequence, the Key fields of the repeated slots are compared (for the hypothetical scenario where different Key objects may generate the same Hash). If there is a match of both Hash and Key, then the insertion method of dict terminates, as it assumes the slot was already present.
This functionality is exactly what makes it desirable for the operation of collapsing repeated sequences: simply inserting all sequences into a dictionary (using each sequence's nucleotides as key) will result in a dictionary containing no repetitions. Once this is achieved, all that is necessary is to output the contents of the dictionary into a sequence file.

The choosing of this method for repeated sequence elimination was influenced mostly by time constraints. This module accounted for a very small footprint of the duration of this project. As can be observed from the excerpt A, the concept applied here is similar to that of the Rabin-Karp algorithm, which is also based on hashing. The underlying technical aspects required to implement Rabin-Karp would constitute a much more complex implementation than the one showcased in this document (again, a considerable factor considering time constraints). This was deemed true, considering in particular the necessary implementation of a hash table structure to avoid using Python's (or another medium of execution's) dict as a hash table. Furthermore, a direct application of Rabin-Karp would not suffice, as it would be necessary to use the algorithm to first obtain the locations of exact sequence matches, and only then would we be able to perform empty string replacements in order to remove the matches (which consist of repetitions). Python's internal implementation of dict is based on C and is optimized for dictionaries with String keys.

4.4 Configuration Files

There are two types of local configuration files required in order for the framework to function properly. One is the .metaframe configuration file to be present in the user's home directory on the machine where Taverna is running (through the use of Python, this feature is trivially supported on operating systems such as Windows 7, OS X and Linux). A small segment example of this configuration file is shown in Fig. 4.16, where an entry for an RSA key is also present. The second type is a template format used to specify a program chosen for a bioinformatics task. Inside the framework's directory, each bioinformatics task shown in Fig. 3.7 has a corresponding directory, minus the Remote WS task. Every single one of these directories contains a task_candidates.ini file which obeys this template format. For the task of Taxonomy, Fig. 4.17 exemplifies a valid configuration of the template. While these template-based configuration files may be manually edited, it is intended that they are edited and updated through the Python script config.py, which has subroutines dedicated not only to accessing the configuration files to determine what software to invoke, but also to changing these files based on user interaction. Although this interaction is supported by execution on a terminal, the script is modularized enough so that it may be incorporated in a GUI application to edit the files in a more user-friendly way.

4.5 Remote Task Execution

Throughout the framework's development, several miscellaneous utility scripts and tools were produced. Although some of these were produced to cater to specific needs, there is confidence that users of the framework may find some of them useful for their goals when configuring MetaGen-FRAME. Some of
the tasks performed with the default software tools included in the framework are very computationally-heavy. An example of this is the blastx step executed in the context of the task of Functional Annotation with Blast2GO. To achieve decent performance (for example, reducing the time taken to perform a blastx computation from six days to less than one for a set with around fifteen hundred contig sequences) figures when executing this step, it was necessary to explore the potential of special hardware (such as the server available to Instituto de Engenharia de Sistemas e Computadores: Investigação e Desenvolvimento em Lisboa (INESC-ID)’s KDBIO team\(^1\) or Instituto Superior Técnico’s technical resource network, Rede das Novas Licenciaturas (RNL), and the cluster infrastructure it provides\(^2\)) and the execution scenarios it allows.

For the purpose of remote execution, a portion of development time was dedicated to exploring and obtaining information on candidate software solutions to enable this feature. One candidate was Taverna’s support for remote workflows via the Taverna Server version, which allows one to configure a dedicated server for remote workflow execution, with built-in mechanisms for securing data. To access the functionality offered by a running Taverna Server, the user would have to send a Taverna workflow document file (file format t2flow, which is the format used for the implemented framework workflow: Skeleton.t2flow) using the Hypertext Transfer Protocol (HTTP) protocol’s request method POST, with the document possibly wrapped in XML. Through similar methods, the triggered remote execution would have an associated identification number which would be sent to the client afterwards. Later on, by usage of HTTP methods, the client would use the retrieved identification to access the outputs of the remote workflow. This procedure to remotely-invoke workflows with Taverna Server is what deterred its use, as this approach was deemed to be insufficiently user-friendly in light of some of the objectives mentioned in this document. Despite this, Taverna Server could become an interesting candidate in the future, with the upcoming web-based interface of Taverna Player\(^3\), which will enable executing existing workflows through the Taverna Server, presumably eliminating the current undesirable lack of a user-friendly GUI for workflow execution.

Considering that bioinformatics task software was being integrated in the workflow as Taverna Tool Service invocations with Python-based layer interfaces in between, it was decided that the distinction between local and remote task execution (and therefore its consequential support) would be supported at the level of the Python layer. This was designed in such a way that users would be allowed to decide which bioinformatics task software would run locally and which would run remotely through the use of a configuration file that centralizes details for both remote server(s) as well as programs (this is the previously-mentioned .metaframe configuration file). Having made this decision, a module named remote.py was developed and included in the Misc directory of MetaGen-FRAME, in order to ease the access of these configured properties from within the Python scripts which operate as interfaces between Taverna and targeted concrete software tools. The aspect of the Python layer which handles the communication between clients and servers (making use of the paramiko\(^4\) Python module which implements the SSH2 version of the SSH protocol for secure connections to remote machines) is performed with an API developed for this purpose. Before the implementation of this module, the remote behavior

\(^1\)More information available at http://www.inesc-id.pt/intranet/laboratoriogrupo/view/group_generalinfo.php?CC=II10

\(^2\)For additional information, please consult https://www.rnl.ist.utl.pt/servicos/cluster/

\(^3\)Taverna’s official website stated it was planned to be released at the end of 2013: http://www.taverna.org.uk/developers/work-in-progress/taverna-player/.

\(^4\)More information available at http://www.lag.net/paramiko/
of three tools was implemented directly in Python. Only after these three implementations were finished and compared side-by-side was the common sequence of actions of remote execution identified. This made it possible to implement this functionality as a module and to refactor the Python implementations so that all use this module as a common dependency. A consequence of this was that the remote behavior was defined in a uniform fashion. Additionally, users wishing to replace a given bioinformatics task program with another one which includes remote execution may employ this module coupled with the usage of configuration files. There is confidence that the default scripts with remote functionality will serve as good-enough examples for clients and users to quickly learn how to use this utility to meet their own ends.

One detail which must be noted is that for remote execution to take place, the targeted server must possess a determined script or program and also a specific directory structure, both being defined in the client machine’s .metaframe file, always under a [Program] section. For the remote implementations which MetaGen-FRAME supports by default, functionality is always divided across two modules: a Python one for specific program execution (always with the same exec prefix: exec_tapyr.py, exec_blast.py, exec_b2g.py, exec_metavelvet.py and exec_bg7.py, all of them located in each one’s respective bioinformatics task sub-directory) and the other (this one takes up the prefix of submit, totaling five scripts as well: submit_tapyr.sh, submit_blast.sh, submit_b2g.sh and then a final composition of submit_metavelvet.sh and submit_bg7.sh, all of them located in the Misc directory of the framework) to be executed on the remote server to invoke a remote copy of the previous module. When setting up the framework for remote execution of any of the default tools, clients will have to reconfigure the submission scripts, as they were configured for explicit running with the KDBIO machinery, which includes a gateway server and a processing server (with a common home directory and the processing server with exclusive access to additional scratch space). Depending on the program to execute remotely and the targeted server, the user must configure one of the script files prefixed with submit in the according way (regarding directory structure and user credentials). The communication between client and server using SSH rely on the user generating an RSA key on the machine where Taverna execution is to take place, and adding it to the list of authorized keys on the server side.

### 4.6 Computing Infrastructure

To employ the potentials of parallel computing for some of the tasks belonging to MetaGen-FRAME’s workflow, some candidate infrastructures were considered. They are described in the following paragraphs, detailing why the final infrastructure of KDBIO was chosen, as well as pros and cons of the considered candidates.

#### 4.6.1 Cluster: Rede das Novas Licenciaturas (RNL)

The first option that was looked into was the experimental cluster infrastructure made available by RNL. Depending on daily activity, it has an amount of available computing cores which may vary between 50
and 240, with each processor being a quad-core with 8 GB of RAM available. This infrastructure does not have any asymmetry in terms of hardware and it offers processing possibilities based on HTCondor\textsuperscript{1}, MPI\textsuperscript{2} and an experimental Hadoop\textsuperscript{3} framework. As an available university facility, it was interesting on account of the prospects of high degrees of parallel computing it supported. Alas, its usage was compromised after many sessions of trial-and-error. Initially, it was decided that the use of Condor to support parallel execution would be discarded, as the installed version lacks the capability of restarting single-node jobs that for some reason failed. On the contrary, Hadoop supports this recovery capability, which made it interesting. Unfortunately, its configuration was in an experimental state (carrying some setup problems) when attempts were made to use it, becoming another candidate crossed off the list. Lastly, the MPI candidate was discarded as it would require time-consuming modifications to some of the programs used, namely, NCBI’s BLAST implementation. Discarding these three options deterred from the usage of the RNL infrastructure. This could be interpreted negatively, as the set of candidate infrastructures grew thinner, but it proved to be a benefit in the long run, as the RNL infrastructure became unusable in the last months of this project due to exhaustive maintenance operations which interrupted its normal functioning, which made its use impossible. One final note must be made regarding the feasibility of executing programs such as blastx on the RNL machinery: they require values of memory greater than the available 8 GB of RAM per computer node (averaging 15 GB of RAM per process on the KDBIO server), so the scope of usable programs on a cluster like that of RNL would be somewhat limited.

4.6.2 Server: Knowledge Discovery and Bioinformatics (KDBIO)

The KDBIO group of INESC-ID, the elected computing infrastructure, has a set of servers available to it. The framework communicates with the target server indirectly by first communicating with a gateway server, where certain programs will be run to launch jobs on the target server. The communication procedure during remote execution is therefore a sequence of two steps supported by SSH. In figure 4.18, the server illustrated on the right side exemplifies the case of using MetaGen-FRAME with this infrastructure. Despite the fact that the available KDBIO gateway and target computational server have a shared user area (after logging into one of the machines, file operations in the user area are synced between all machines of that internal network), communication and file transfer between them is performed through Secure Copy (SCP). If this was not done, it would imply MetaGen-FRAME assumes the machines in a remote LAN have synchronized user directories, which would restrict the range of systems capable of running the framework. The lowest requirement is therefore that the remote machines, with existing gateway or not, are able to communicate between each other using SSH. Enabling the usage of this infrastructure was a project in itself, as either some software was missing or the desired versions or editions were amiss. Furthermore, only regular user access was available, so many programs’ default installation mechanisms (often using system folders) were off-limits. The following are some of the most important packages that were configured and enabled in the computational

\textsuperscript{1}Distributed computing software which allows professionals to increase their computing throughput. More information available at \url{http://research.cs.wisc.edu/htcondor/}.

\textsuperscript{2}The Message-Passing Interface standard, which provides tools to define properties such as process topologies and management, among many others. Check \url{http://www.open-mpi.org/} for information on the implementation available at RNL.

\textsuperscript{3}A software library designed to allow for the distributed processing of large data sets across clusters of computers, by means of simple programming models. For additional insight, please visit \url{http://hadoop.apache.org/}. 
server (this also counts the overhead of learning the setup procedures’ intricacies, which were aplenty):

- MySQL Server 14.14 Distrib 5.6.12 (x86_64): configuring and installing this software required researching online tutorials in order to ascertain the process to follow. After the configurations were done, databases with specific properties for Blast2GO were created.

- Oracle Java SE 7 (64-bit): certain software packages recommended or required at least version 6 of Oracle’s Java. As administrative access was not available and contacting the staff responsible for the infrastructure to install it was not an option, it was downloaded to a local user directory. Minor configurations were then made to make sure the location of Oracle’s software was automatically added to the user program path upon logging in.

- BG7: the inclusion of this tool required some tweaking of the program's skeleton, written in bash. Essentially, there was poor documentation to be found regarding the installation procedure. Furthermore, the official version appears to contain quite an amount of errors, many of which were only found by carefully inspecting the entrails of the main script bg7. For example, this program has some intermediate BLAST steps which produce XML results. However, it was later (through the authors’ online forums) identified that for BG7, the BLAST+ 2.2.25 version is recommended. Not wanting to make the set of software packages in the framework more heterogeneous, a decision was made to employ the usage of the most recent version, BLAST+ 2.2.28. Apparently, BG7 checks if a BLAST execution was valid by checking if the last line corresponds to a specific XML tag produced by NCBI BLAST. With BLAST+ 2.2.28, this tag is produced, but an extra empty line is added at the end of the result file. Not only did this result in BG7's execution failure, it also caused the program to exit silently. This was frustrating and consumed a considerable amount of time, as only close inspection allowed this problem to be identified (let alone realize that execution did not go smoothly in the first place). This was not an isolated case, as other similar problems were encountered. Lastly, another ludicrous error worthy of mention is the way the main script of BG7 handles information: it attempts to access some files inside BG7’s directory structure, but BG7’s access to its own file structure has errors on the internal reference file paths. Despite these facts, BG7 is still a worthy reference in the literature when it comes to Gene Structure Prediction.

- BLAST+ 2.2.28 x64 (NCBI C++ Toolkit): installing this program required looking at the online information and tutorials the NCBI has available. This documentation was relevant to learning the proper invocation and parameters to use with BLAST (and its automated invocation via remote scripts). The software comes prepackaged with a configuration to access NCBI’s remote data bases. Since only a subset of them were needed and the space they require is moderate (counting a total of around 45 GB), they were downloaded to the computing server via some of the official scripts that come in the package. Finally, it was ensured that parallel execution requires a simple breakup of the original input file into fragments which are processed using independent instances. An ordered concatenation of the fragment results in XML format results in a file identical to what would be produced with a single process, allowing for the benefit of lower execution time.

- Blast2GO: this software makes use of various resources and data bases. It performs an initial stage of BLAST, but it is also capable of accepting the output of a previous execution. This is one of the major factors in favor of having the ability to execute the algorithm in a parallel fashion with
local data bases: execution time is reduced with increasing degrees of parallelism, and the local
data bases reduce the bottleneck of having to contact the NCBI servers when aligning.

4.6.3 Additional Utilities

To support the framework's internal programs, tools were developed to tackle certain minor tasks, some of which common to more than one program. Most of them may be found in the Misc directory of MetaGen-FRAME, and we are hopeful that they may constitute a toolbox to aid users in unrelated but also important needs. Examples of these utility scripts include conversions between many sequence file formats, some statistic output functionality and the scripts used in remote program execution, which more savvy users may which to use outside the framework's bulk.
CHAPTER 4. FRAMEWORK IMPLEMENTATION

Figure 4.16: A short segment highlighting the possible contents of the .metaframe configuration file. As shown, it can contain two categories of sections, with one type dedicated to defining remote servers to be targeted during execution and the other serving the purpose of defining the execution mode and other specific properties of programs. The first two sections displayed in the figure show the typical properties to be defined for a desired remote server, such as the RSA key (used when communicating with the remote server using the SSH2 variant of the SSH protocol. Additional detail given in section 4.5). The last two sections are for program configurations: the first one specifies that MetaPhlAn is to be run locally and to return taxonomic results in decreasing value of identity; the second one configures remote parallel TAPyR execution.
CHAPTER 4. FRAMEWORK IMPLEMENTATION

Figure 4.17: Example of the contents of the configuration file for MetaGen-FRAME’s bioinformatics task of Taxonomy. As visible in the image, this template format always begins with a header section with two variables, one defining the amount of programs that have been added to the framework and the other showing the concrete program selected for execution in a workflow. The value of `chosenprogram` must be the name of one of the sections that appear in the following lines of the document.

```
[Header]
numbervofprograms = 2
chosenprogram = MetaPhlAn

[MetaPhlAn]
path = Taxonomy/MetaPhlAn/invoke.py
command = python

[MEGAN4]
path = Taxonomy\MEGAN\MEGAN_unix_4_70_4.sh
command = ./
```

Figure 4.18: Illustration of a hypothetical execution scenario for MetaGen-FRAME. It may be configured so that different bioinformatics task software executes on separate remote machines, which may be directly-accessible over the Internet, or may require an indirect connection through a gateway and further into a private network.
Chapter 5

Evaluation and Results

The programs used for bioinformatics tasks were executed on a computational infrastructure available to the KDBIO group. It has an Intel® Xeon® E7-4830 with 32 cores (4 Central Processing Unit (CPUs) with 8 Cores @ 2.13 GHz) and 256 GB RAM @1066 MHz running Suse Linux Enterprise Server 11. The degree of parallelism running any program related to the framework never surpassed the amount of physical cores, in order to attempt to minimize the influence of hyperthreading.

5.1 Parallel TAPyR

As stated previously, a parallel implementation for TAPyR was proposed and successfully achieved. Its performance was analyzed on the computing infrastructure initially described in this section. The data sets analyzed were a subset of some used in the benchmarks of the original TAPyR and also an additional metagenome. We now present the results obtained whilst performing alignments of this metagenome against an index reference composed of *Pseudomonas fluorescens* (SBW25 complete genome) and *Sulfurospirillum deleyianum* (DSM 6946, complete genome), totalling 9 Mbp approximately. This metagenome is interesting because it contains gut bacteria from a conch organism capable of performing degradation of sea toxins. The metagenome has 1.37 millions of reads, 516 million base pairs with an average of 376 base pairs per read. Figure 5.19 shows the obtained speedups with this metagenome and on Figure 5.20, the total time taken to perform alignments is shown. Active Slaves represents the number of physical CPU cores employed in parallel processing. The benchmarks were performed with an increasing amount of cores used (in intervals of five).

The red line represents execution time taking in account the overhead of merging fragment results into a final consolidated file, while the blue line does not consider that. We denote that the difference is relatively small, so this operation is mostly negligible, specially for our targeted application on large volume data. As the volume of processed information increases, the temporal cost of the merge operation constitutes a progressively-smaller amount of total execution time. As a consequence, greater data set sizes would be necessary in order to ensure scalability in the context of the gains of parallel computing and the impact of joining results together.

Additional benchmark results are provided in Annex B, as well as the poster presented in the COST
Workshop.

Figure 5.19: Measurements of metagenomic conch (against indexed *Pseudomonas fluorescens* and *Sulfurospirillum deleyianum*) data set speedups.

### 5.2 Collective Analysis

In this subsection we present a set of tables and graphics which showcase preliminary results obtained on a set of metagenomes. They are important due to being associated with marine environments with natural leanings for bio-degradation. These data sets were obtained through our liaison and collaborative work performed with the LNCC. Table 5.1 shows some important characteristics of the data sets used, as well as the two most abundant detected organisms for each data set. The identified OTUs were obtained from NCBI queries:

- **Mycobacterium sp.** JDM601 RefSeq Genome, identified in the data sets F6AK87O01 and also F6AK87O02.

- **Frankia sp.** CcI3 RefSeq Genome, identified in the data sets F6AK87O01 and F6AK87O02.

- **Thioalkalivibrio sp.** K90mix RefSeq Genome, identified in the data set F6AK87O03.

- **Desulfovibrio desulfuricans** ND132 RefSeq Genome, identified in the data set F6AK87O03.

- **Flavobacterium sp.** SCGC AAA160-P02 RefSeq Genome, identified exclusively in the data set G33QMDA02.

- **Candidatus Pelagibacter ubique** HTCC1062 RefSeq Genome, identified in G33QMDA02 and also G33QMDA03.
Figure 5.20: Timing of the metagenomic conch (against indexed *Pseudomonas fluorescens* and *Sulfurospirillum deleyianum*) data set execution.

- *Cellulophaga lytica* DSM 7489, identified in the data set G3QM'DA03.

The most represented OTUs in the samples were used as references for P-TAPyR as individual indexes. For each sample, we aligned its sequences (post-filtering stage) against its corresponding indexes. The results of these alignments are presented in Table 5.2. The amount of valid alignments obtained, as well as the counts of the subsequently generated contigs, were considerably low, when compared to the number of sequences present in the original samples. The unaligned sequences obtained during Alignment were submitted to *de novo* reconstruction using the MetaVelvet assembler. Although the usage of statistical methods and de Bruijn graphs in this software led to a greater number of contigs, they were still too few and also not as long as it would be desirable. These results were relevant as they served the purpose of validating and providing additional insight on the methodologies used by researchers at LNCC. Through these processes, we were able to identify an excessive representation of the *Candidatus Pelagibacter ubique* organism in some samples, implying sample contamination and that researchers will need to sequence greater quantities of data. Additionally, we present Figure 5.21, which showcases a phylogenetic map, as well as Figure 5.22 which shows the distribution of identified OTUs on the different data sets.

These results obtained through cooperation efforts serve to highlight the importance and potential of the tools integrated in MetaGen-FRAME. They are interesting applications not only for the performance and biological acuity displayed in their respective publications, but also for their functionalities for result visualization. A direct example of this is Figure 5.21, which was generated by MetaPhlAn.
### Table 5.1: Taxonomic Analysis: MetaPhlAn (defaults). Statistics for the taxonomy of the listed metagenomes. For each data set, the two identified OTUs with the highest identity are shown. Further analysis was performed on these top references.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Top-2 Strains</th>
<th>Length (bp)</th>
<th>Identity(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>F6AK87001</td>
<td><em>Mycobacterium sp.</em></td>
<td>4 643 668</td>
<td>17.24</td>
</tr>
<tr>
<td></td>
<td><em>Frankia sp.</em></td>
<td>5 433 628</td>
<td>16.29</td>
</tr>
<tr>
<td>F6AK87002</td>
<td><em>Mycobacterium sp.</em></td>
<td>4 643 668</td>
<td>18.70</td>
</tr>
<tr>
<td></td>
<td><em>Frankia sp.</em></td>
<td>5 433 628</td>
<td>14.60</td>
</tr>
<tr>
<td>F6AK87003</td>
<td><em>Thioalkalivibrio sp.</em></td>
<td>2 744 800</td>
<td>20.13</td>
</tr>
<tr>
<td></td>
<td><em>Desulfovibrio desulfuricans</em></td>
<td>3 858 580</td>
<td>14.87</td>
</tr>
<tr>
<td>G33QMDA02</td>
<td><em>Flavobacterium sp.</em></td>
<td>2 544 614</td>
<td>36.46</td>
</tr>
<tr>
<td></td>
<td><em>Candidatus Pelagibacter ubique</em></td>
<td>1 308 759</td>
<td>29.17</td>
</tr>
<tr>
<td>G33QMDA03</td>
<td><em>Candidatus Pelagibacter ubique</em></td>
<td>1 308 759</td>
<td>44.84</td>
</tr>
<tr>
<td></td>
<td><em>Cellulophaga lytica</em></td>
<td>3 765 936</td>
<td>18.34</td>
</tr>
</tbody>
</table>

### Table 5.2: Alignment: P-TAPyR. This table has statistics for the resequencing-based assembly with the previous top references.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Top-2 Strains</th>
<th>Coverage</th>
<th>#Reads</th>
<th>Avg. Length (bp)</th>
<th>#Contigs</th>
</tr>
</thead>
<tbody>
<tr>
<td>F6AK87001</td>
<td><em>Mycobacterium sp.</em></td>
<td>2.32</td>
<td>307</td>
<td>513</td>
<td>210</td>
</tr>
<tr>
<td></td>
<td><em>Frankia sp.</em></td>
<td>1.23</td>
<td>255</td>
<td>607</td>
<td>110</td>
</tr>
<tr>
<td>F6AK87002</td>
<td><em>Mycobacterium sp.</em></td>
<td>3.72</td>
<td>572</td>
<td>560</td>
<td>309</td>
</tr>
<tr>
<td></td>
<td><em>Frankia sp.</em></td>
<td>1.60</td>
<td>376</td>
<td>632</td>
<td>137</td>
</tr>
<tr>
<td>F6AK87003</td>
<td><em>Thioalkalivibrio sp.</em></td>
<td>1.14</td>
<td>123</td>
<td>537</td>
<td>58</td>
</tr>
<tr>
<td></td>
<td><em>Desulfovibrio desulfuricans</em></td>
<td>0.32</td>
<td>50</td>
<td>567</td>
<td>22</td>
</tr>
<tr>
<td>G33QMDA02</td>
<td><em>Flavobacterium sp.</em></td>
<td>0.79</td>
<td>84</td>
<td>343</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td><em>Candidatus Pelagibacter ubique</em></td>
<td>13.99</td>
<td>988</td>
<td>357</td>
<td>512</td>
</tr>
<tr>
<td>G33QMDA03</td>
<td><em>Candidatus Pelagibacter ubique</em></td>
<td>21.77</td>
<td>1940</td>
<td>463</td>
<td>614</td>
</tr>
<tr>
<td></td>
<td><em>Cellulophaga lytica</em></td>
<td>0.82</td>
<td>238</td>
<td>630</td>
<td>49</td>
</tr>
</tbody>
</table>

### Table 5.3: De Novo Assembly: MetaVelvet (defaults). Contig count and average sequence lengths after processing sets of unaligned sequences using MetaVelvet, which is based on the decomposition of de Bruijn graphs.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>#Contigs</th>
<th>Avg. Length (bp)</th>
</tr>
</thead>
<tbody>
<tr>
<td>F6AK87001</td>
<td>5183</td>
<td>326.40</td>
</tr>
<tr>
<td></td>
<td>4575</td>
<td>324.73</td>
</tr>
<tr>
<td>F6AK87002</td>
<td>1678</td>
<td>323.13</td>
</tr>
<tr>
<td></td>
<td>1638</td>
<td>329.25</td>
</tr>
<tr>
<td>F6AK87003</td>
<td>653</td>
<td>307.32</td>
</tr>
<tr>
<td></td>
<td>678</td>
<td>303.26</td>
</tr>
<tr>
<td>G33QMDA02</td>
<td>18881</td>
<td>275.76</td>
</tr>
<tr>
<td></td>
<td>18832</td>
<td>274.48</td>
</tr>
<tr>
<td>G33QMDA03</td>
<td>8182</td>
<td>308.98</td>
</tr>
<tr>
<td></td>
<td>8764</td>
<td>307.01</td>
</tr>
</tbody>
</table>
5.3 Further Study: *E. coli* Strain

To also illustrate the application of MetaGen-FRAME on data sets known to be composed with sequences of a single strain, herein we display results of Functional Annotation on an *E. coli* strain which was mentioned in the BG7 software publication [57]. This strain’s formal nomenclature in its concrete sequences’ names is *Escherichia coli* O104:H4 str. TY-2482. The framework, for this execution, was configured to retrieve and process only the OTU with the highest identity percentage out of all the results obtained during the task of Taxonomic Analysis. This led to the Remote WS operation retrieving the genome strain *Escherichia coli* 55989 chromosome, complete genome from the NCBI servers. After aligning the BG7 data set against this retrieved strain, a set of contigs were generated. These were then passed on to Blast2GO to produce information on the data set’s composition and main functions. To corroborate the fact that an *E. coli* strain was aligned against a set of fragment sequences from another strain of the same organism, we show Figure 5.23, which contains a BLAST-based species distribution. As expected, the presence of *Escherichia coli* is greater than other OTUs detected by Blast2GO. Figures 5.24, 5.25 and 5.26 highlight the main properties pertaining, respectively, biological processes, cellular components and molecular functions. The information that the software produces for these categories is especially important, as it condenses sets of terms relevant to the main functionalities present in the sample, helping to corroborate beliefs about potential reactions that occur within the sample (metagenome) and the way its organisms may influence their environment.
Figure 5.21: A phylogenetic tree we produced during collaboration efforts between KDBIO and LNCC. It shows a grouped analysis for the metagenomes previously mentioned. The purple area $A$ on the top-left side refers to the genus *Sphingobacteriaceae unclassified* while the cyan $B$ area on the bottom is associated to the genus *Candidatus Pelagibacter*. 
Figure 5.22: A depiction of OTU distribution through the analyzed metagenomes. It is particularly relevant to note the high presence (denoted by the yellow coloring) of the *Candidatus Pelagibacter ubique*, important for some of its capabilities for recycling dissolved organic carbon (dissolved organic carbon plays an important role in the global carbon cycle and is also relevant in the transport of metals in aquatic systems).
Figure 5.23: Species distribution for the contigs related to the valid alignments obtained with the different strains of *E. coli*. The longest cyan block at the top illustrates the dominance of this organism in terms of percentage of BLAST hits.
Figure 5.24: Counts of Gene Ontology biological processes and the amount of sequences associated to each of them. Although not the most present biological process term, it should be noted the considerable presence of phosphorylation-related terms (fourth red rectangle from the top). Phosphorylation is relevant for its ability to activate and deactivate many protein enzymes, altering their activity and function as a result.
CHAPTER 5. EVALUATION AND RESULTS

Figure 5.25: Distribution of Gene Ontology terms related to cellular components.
Figure 5.26: Sequence counts for several Gene Ontology terms, obtained with Blast2GO. Particularly important is the presence of ATP binding terms, which are responsible for the molecular function of enzyme and coenzyme regulation (second red rectangle from the top).
Chapter 6

Conclusions and Future Work

In this document, metagenomics and the benefits that arise from innovations in this field were described. A historical background was provided, explaining how it became what it is today and why certain tasks in the study of metagenomic samples are of the highest importance. With metagenomics being a complex field, an effort was made to bring attention to its most relevant tasks throughout this document. Focus was put on presenting high-level descriptions of the most known pipelines and methods in the literature, while at the same time not excessively deepening the level of detail. As mentioned earlier in this document, there is a scarcity of platforms that attempt to integrate all of the different metagenomic task perspectives in unison. Consequently, it is not trivial to produce an illustrative definition of the generic components that existing metagenomic components have. This is due to the fact that they tend to specialize in one purpose or another, which has an impact on the components that are present. However, as a result of the survey undertaken, there is confidence that the means proposed to produce a metagenomic framework adequately consider the experience and configuration of existing platforms. Choosing and deciding an infrastructure for the framework (Taverna was chosen) was an important part of this project. Implementing the software infrastructure for remote execution and communication between modules was also a task deserving of attention. Determining how to employ Python and some of its features (including open-source third-party modules) to allow for secure communication (including code execution and file transfer) in a reusable way was also an important and necessary part of development.

This project resulted in the creation of MetaGen-FRAME, a flexible package with mechanisms for the configuration of specific bioinformatics task softwares and computing infrastructure selection. The proposed modification of the TAPyR sequence aligner was also successfully achieved and integrated in the framework. This parallel implementation of TAPyR, P-TAPyR, spanned a poster presentation in Barcelona and a technical article [4] presented in the EuroMPI 2013 conference in Madrid. It was chosen as one of the best articles presented in the conference, leading to the authors being invited to further extend it for publication in the Parallel Computing international journal.
6.1 Future Work

The goal of a future project would be to improve and add novel functionality to MetaGen-FRAME in order to produce a more polished product, targeting professionals who may lack strong backgrounds in computer engineering and informatics, from both academia and the corporate world. The aim is to create a metagenomic suite of state-of-the-art software tools with robust functionality to harness the parallel computing potential provided by some of the hardware infrastructures that have become more common in recent years (spanning technologies such as clusters, multi-core machines and general-purpose Graphics Processing Units (GPUs)), such as the Amazon Elastic Compute Cloud, to give an example. This project would intend to produce high-quality work which bridges the needs of the field of Biology (focused on metagenomics) and the solutions to said problems using techniques and methods from the field of Computer Engineering and Information Systems. During development, there would be an aim to collaborate with biology and bioinformatics professionals, considering them as end-users, in order to ascertain the major biological outputs and functionality that are desired as the research fields evolve. This is true both in terms of the semantic quality of the results as well as the way they are presented. It is intended that, through close collaboration with these professionals, the most desirable means of usage will be determined in order to enable an easy and intuitive wielding of MetaGen-FRAME.

Striving towards such a vision, some of the goals taking priority may be defined. It would be highly interesting to include some layer of meta data production in the framework which makes use of the Resource Description Framework (RDF). Since metagenomics and related scientific fields have a lot of interconnected information, being able to create a more rich network between concepts, data and terms is a very coveted capability. This would facilitate reasoning and analysis of scientific case studies, by heightening researchers’ abilities of tracing back information and the processes that led to concrete conclusions and results. This would also be a helpful property for information when it comes to visualization capabilities and traceability over the Internet (this one is especially relevant, given the distributed nature of biological resources all over the globe). To terminate, we feel it is important to mention that there are ongoing developments for Taverna. Specifically, a Taverna Player program is being produced. It will allow the visualization of remote workflow executions through the web browser. It is officially set for release by the end of 2013. It would be interesting to experiment with remote workflow execution using Taverna Server then inspecting and transferring back results via the Taverna Player’s main mechanisms, possibly removing the need for some of the complexity in the framework.
Bibliography


Appendix A

Code Excerpts

Excerpt of `collapse_sequences.py`:

```python
import sys
import os
import string
import itertools
import copy

# Script entry point.

if not len(sys.argv) == 4:
    script_name = __file__[__file__.rfind(os.sep) + 1:]
    print 'Usage:
    $ python ' + script_name + '
    <FASTQ_file>
    <output_file_path>
    <output_file_name>'
    sys.exit(-1)

fastq_path = os.path.abspath(sys.argv[1])
output_directory = os.path.abspath(sys.argv[2])
target_file = sys.argv[3]
target_path = os.path.join(output_directory, target_file)

print 'Target collapsed FASTQ file is ' + fastq_path
sequences = {}
original_seq_count = 0

try:
    with open(fastq_path, 'r') as seq_file:
        is_nt = False
        is_qual = False
```
for line in seq_file.readlines():
    line = line.strip()
    if is_nt:
        nt = line
        is_nt = False
    elif is_qual:
        qual = line
        is_qual = False
    sequences[nt] = [seq_name, nt, qual]
    if line.startswith('@'):
        original_seq_count = original_seq_count + 1
        seq_name = line
        is_nt = True
    elif line == '+':
        is_qual = True

with open(target_path, 'w') as fastq_file:
    for seq in sequences.values():
        fastq_file.write(seq[0] + '
')
        fastq_file.write(seq[1] + '
+
')
        fastq_file.write(seq[2] + '
')

    removed_count = str(original_seq_count - len(sequences))
    print '> Produced collapsed FASTQ file ' + target_path
    print '> A total of ' + removed_count + ' sequences were removed.'
except IOError, e:
    sys.stderr.write('ERROR: %s\n' % str(e))
    sys.exit(-1)
except ValueError, e:
    sys.stderr.write('ERROR: %s\n' % str(e))
    sys.exit(-1)
sys.exit()
Appendix B

P-TAPyR Material

B.1 Benchmarks

This section of Appendix B contains three pairs of figures, each pair detailing the speedup and execution time for the *Caenorhabditis elegans* (Figures B.27 and B.28), *Drosophila pseudoobscura* (Figures B.29 and B.30) and *Plasmodium falciparum* (Figures B.31 and B.32). *Caenorhabditis elegans* was benchmarked with the inclusion of the result merging operation to demonstrate that if a smaller data set such as this one is used, the merge operation accounts for a greater proportion of the total execution time, justifying our initial claims that the potential gains of parallelizing TAPyR would manifest on larger volumes of data, which was verified by the conch metagenome benchmark.
Figure B.27: Measurements of *Caenorhabditis elegans* index data set speedups.

Figure B.28: Timing of the *Caenorhabditis elegans* index data set execution.
Figure B.29: Measurements of *Drosophila pseudoobscura* index data set speedups.

Figure B.30: Timing of the *Drosophila pseudoobscura* index data set execution.
Figure B.31: Measurements of *Plasmodium falciparum* index data set speedups.

Figure B.32: Timing of the *Plasmodium falciparum* index data set execution.
Abstract

Background: In the context of resequencing projects, the efficient and accurate mapping of reads to a reference genome is a critical problem. One instance of this problem is the alignment of pyrosequencing reads produced by the GS FLX (454) system against a reference sequence. TAPyR (Tool for the Alignment of Pyrosequencing Reads) [1] implements a methodology to efficiently solve this problem, which proved to yield results of a quality (both in terms of content and execution speed) higher than those of mainstream applications. With the goal of further improving this platform’s results, the query and reference sequence access procedures in the original version have been made parallel [2].

Results: Through the use of multi-threading, this new version produces considerable reductions in the processing time of queries, scaling with the amount of hardware-supported threads (not accounting for hyper-threading) available.

Conclusions: We present the modifications made to this software tool to allow for parallel querying of reads against an indexed reference which scales proportionally to the amount of available physical cores.

Context

Although initially developed having the GS FLX (454) technology in mind, TAPyR performs equally well on a range of platforms that can return their sequencing reads in the FASTA, AXXTO or SFF formats, including Illumina, Ion Torrent and Pacific Biosciences technologies. As the availability of machines with multi-core processing capabilities is spreading, it makes sense to adapt the versatile tool TAPyR to harness the power of parallel computing, by decomposing the mapping procedure in concurrent tasks. This has become ever more important with the advent of NGS technologies and the large volumes of data produced in the field of metagenomics.

Performance tests were executed on a 32-core rack with 256 GB RAM. The following graph shows the measured speedups and execution times for increasing degrees of parallelism for a metagenomic conch reference data set. As illustrated, for large data sets such as the one employed, performance scaled practically linearly with the number of parallel hardware-supported tasks executed.

References


Acknowledgements

The authors were supported by FCT by national funds through FCT Foundation para a Ciência e a Tecnologia, under projects PEst-OE/EEI/UI0021/2011 and TAGIS PTDC/EIA/112083/2009.

Figure 1 – Overview of the parallel version of TAPyR

Figure 2 – Time comparison of the serial execution and different parallel configurations