Computational Methods and Applications for Real-Time Identification of Species and Pathogens from Raw Read Sequencing Data.

Egyetemi doktori (PhD) értekezés

Ramin Karimi

Témavezető: Dr. András Hajdu

DEBRECENI EGYETEM

Természettudományi Doktori Tanács

Informatikai Tudományok Doktori Iskola

Debrecen, 2017

\

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Hungary, Debrecen, June 2017

Ramin Karimi

This is to certify that the thesis entitled "Computational Methods and Applications for Real-Time Identification of Species and Pathogens from Raw Read Sequencing Data." submitted by Ramin Karimi to University of Debrecen for the award of the degree of Doctor of Philosophy is a bona fide record of the research work carried out by him under my supervision and guidance. The content of the thesis, in full or parts have not been submitted to any other Institute or University for the award of any other degree or diploma.

Hungary, Debrecen, June 2017

Dr. András Hajdu

Computational Methods and Applications for Real-Time Identification of Species and Pathogens from Raw Read Sequencing Data.

Értekezés a doktori (Ph.D.) fokozat megszerzése érdekében
a informatika tudományágban

Írta: Ramin Karimi okleveles informatikus

Készült a Debreceni Egyetem Informatikai Tudományok doktori iskolája
(Diszkrét matematika, adatfeldolgozás és vizualizáció programja) keretében

Témavezető: Dr. András Hajdu

A doktori szigorlati bizottság:

elnök: Prof. Halász Gábor

tagok: Dr. Péter Balázs

Dr. Antal Bálint

A doktori szigorlat időpontja: 2015. November 19.

Az értekezés bírálói:

Dr. ....

Dr. ....

Dr. ....

A bírálóbizottság:

elnök: Dr. ....

tagok: Dr. ....

Dr. ....

Dr. ....

Dr. ....

Az értekezés védésének időpontja: **2017**

Table of Contents

List of Figures

List of Tables

Acknowledgments

First and foremost I would like to express my sincere gratitude to my advisor Dr. András Hajdu for the continuous support of my Ph.D. study and related research, for your patience, motivation, and immense knowledge. Your guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D. study. You have set an example of excellence as a researcher, mentor, and instructor.

Besides my advisor, I would like to thank my thesis committee members, Professor János Sztrik, Dr. Tamás Bérczes, and Dr. Gábor Szũcs, for all your guidance, insightful comments, and encouragement through this process. Your discussion, ideas, and feedback have been absolutely invaluable.

I would like to dedicate this thesis to my lovely wife Samira Sajjadian for her endless love, understanding, tolerance, and sacrifices. Words cannot express my gratitude for her patience, continual source of support with strong encouragement at every stage, and cope with living far away from our family throughout the journey of the research. She is not only my lovely wife but also my best friend who gives me wise and timely advice when I need it.

I am deeply indebted to my family for their love, support, and encouragement. Without them, this thesis would never have been written. Words cannot express how grateful I am to my parents, my parents-in-law, my brothers, my brothers-in-law, my sister, and my sister-in-law for all of the supports, and sacrifices that they have made on my behalf.

My sincere thanks also go to Professor Ladjel Bellatreche, Professor Patrick Girard, and Dr. Brigitte Vannier, who provided me an opportunity to join their team at Poitiers University in France, and for their precious support during a year, I was there as an exchange member.

Special thanks also to Dr. Edéné Rutkovszky, Dr. Zsuzsanna Bokor, Dr. Marianna Zichar for their kind help and consideration.

Special thanks to the Faculty of Informatics, Department of Computer Graphics and Image Processing, and the Doctoral School of Informatics at the University of Debrecen for providing this research opportunity and providing the computational resources for this research.

Special thanks to Mr. József Akai and his family, my dear Hungarian friends, for making me and my wife feel part of their family in the years of our stay in Hungary. We will never forget their kindness, hospitality, and friendship.

Finally, special thanks to all my friends for their support and sharing the happiness and great moments we had during last few years.

Abstract

A very small proportion (often cited as <1%) of the total microbial diversity in nature can be cultivated in the laboratory. The vast majority of them cannot be isolated or are extremely difficult to grow in the laboratory. In recent years, impressive progress has been made in the field of bioinformatics by technological advances of genome sequencing to perform whole genome sequencing in thousands of individuals.

The emerging field of metagenomics provides a series of technical innovations for culture-independent scrutiny of microbial communities in the environment. It is a large-scale sequencing of the entire community, sampled directly from its natural environment. It provides new opportunities for gaining access to previously hidden phylogenetic, functional, metabolic, and ecological diversity of organisms and their community structure. While these technologies are constantly continued to offer increases in throughput, the time and cost of DNA sequencing continue to fall. Therefore, sequencing technologies are becoming applicable as a routine tool for diagnostic and public health microbiology. However, the complexity of the analysis and high-costs of the computational resources has encountered many challenges and obstacles to achieving this goal.

One of the major challenges for metagenomics studies is the accurate identification of organisms present in complex environments. Although, a wide variety of assembling and alignment-based algorithms, software and computational analysis workflows have been subsequently developed, computational approaches for alignment-based identification of complex communities, without very extensive sequencing coverage are inadequate for even the most abundant members.

In this research, we have proposed an alignment-free method and its appropriate pipelines and software for Real-time identification of species and strains from raw read sequencing data. The method tries to shortcut identification into a quick and accurate process in environmental and clinical sequencing samples, using parallel and distributed computing on commodity hardware for enhancing the applicability of the analysis as a routine process in the entire research community.

Chapter 1

Introduction

1.1 Motivation and Overview

The number of bacterial cells (10^{14}) inhabiting in an average healthy adult human body is estimated tenfold more than human cells (10^{13}) [1][2] and the number of existing microbial species in the world is estimated at 10^{7} to 10^{9} [3][4][5]; therefore, they play a pivotal role not only in Human life but also the whole life on Earth. Over 99% of microorganisms in our planet are not cultivable in vitro or require a long and difficult cultivation period [6][7][8]; thus, how can we discover these unseen occupiers of our body and our planet? What is the appropriate solution for the rapid identification of these best Friends and the worst enemies?

In 1983, Kary Mullis, a scientist at the Cetus Corporation, conceived of Polymerase Chain Reaction (PCR) as a method to synthesize unlimited copies of a small fragment of target DNA. Over the next two years, a team of Cetus scientists that recognized the impact of PCR on molecular biology could turn the theoretical process into reality [9].

Polymerase Chain Reaction (PCR) is a quick, easy, and inexpensive laboratory technique to amplify a single copy or a few copies of a small fragment of target DNA (the template) to unlimited copies. PCR can synthesize any particular piece of DNA [10]. PCR technique applies a series of 20-40 repeated temperature cycles. Each cycle consists of three stages:

1- Denaturing: heating temperature to separate the double-stranded template DNA into two single strands.
2- Annealing: cooling temperature to enable the DNA primer to attach to the template DNA and serve as a starting point to copy the DNA.
3- Synthesis: raising the temperature and making the new strand of DNA by the Taq polymerase enzyme.

The invention of the PCR technology provided a new era of studying uncultivable microorganisms, but there was still a long way ahead. PCR has some limitations.

1- Amplification inhibitors such as detergents, antibiotics, enzymes, polysaccharides, fats, proteins, and salts can reduce the amplification efficiency [11]

2- PCR is a highly sensitive; any form of contamination of the sample can produce false positives or false negative results [12][13].
3- Another limitation of PCR is the length of the fragment that can be amplified. PCR works well over short stretches of DNA up to about 2 kbp[1].
4- PCR can only be used to identify the presence or absence of a known pathogen or gene [14].
5- PCR cannot be used to identify species in complex communities.

Later with emerging the microarray approaches, identifying uncultivable microorganisms entered into a new stage. The microarray is the combination of a very large set of distinct probes attached to a solid structure (Glass slides). Probes are small fragments of sequences which are complementary to a pathogen-specific gene sequence [15]. According to the reports, microarray techniques bring the possibility of species identification or detecting and diagnosing of various bacterial samples at the same time with main advantages of high throughput, parallelism, miniaturization, speed, and automation.

Microarray has been limited to a small set of functional genes such as 16S rRNA genes, and it is not a suitable approach to investigate the uncultivable majority of the species in the environment [15][16].

With respect to remarkable abilities of PCR and microarray approaches, they are not enough powerful for studying complex environmental and clinical samples which contain hundreds or thousands of different species.

With the advancement of Next-generation sequencing (NGS) which is a combination of massively parallel sequencing technologies besides PCR and microarray techniques, considerable progress has occurred not only in the phylogenetic and functional analysis of microbial communities but also in their affiliated science, significantly. It is a culture-free method that enables analysis of the entire microbial community within a sample. It has the ability to combine many samples in a sequencing run.

1- bp: base pairs; it is a unit of length for the strands of DNA or RNA

The 16S ribosomal RNA (16S rRNA) gene with about 1,500 bp length is part of the bacterial DNA and generally contains nine "hypervariable regions" (V1 – V9) that represent considerable sequence diversity among different bacterial species and can be used for species identification [17].

16S rRNA is the most common standard culture-free approach which is currently used for taxonomic assignments, bacterial identification, and studying bacterial diversity in ecology and clinical microbiology [18][19]. It is also used to design the primers for Polymerase Chain Reaction (PCR) and probes for microarray studies [20]. There are a considerable number of publications about limitations of 16S rRNA gene.

The major limitation is that the copy numbers of 16S rRNA per genome vary from 1 up to 15 or more copies [18][21]. Therefore, the amount of 16S rRNA variants is estimated to be 2.5-fold greater than the number of bacterial species [18][22]. Moreover, 16S sequences of the same species or even the same genome are often different [18].

The ambiguous and incorrect identity of species and also the artificial classification of an organism into more than one species can be led by divergent evolution of rRNA genes [23]. This problem can be solved in cultivable species by cloning rRNA genes from the pure culture of that species to identify the degree of variation [23]. As mentioned above, more than 99% of species are culture-independent, therefore in a complex and mixed community of microorganisms, sequence heterogeneity of 16S rRNA within a single genome can lead overestimation of the microbial diversity [22].

Short reads as the output of the high-throughput sequencing technology are very noisy and partial, with too many missing parts [16]. As an example, in the first phase of the Global Ocean Sampling (GOS) expedition, only 4,125 distinct full length or partial 16S genes were identified from 7.7 million metagenomics reads [24][26]. According to another release of GOS, the total of 142,783 (1.4%) 16S genes from 80 GOS metagenomes were obtained [25][26].

Moreover, most of the reads from recent NGS platforms are too short in length [22], thus de novo assembly is required in order to make longer sequences. It may represent an extra limitation to use 16S rRNA fragments for taxonomic assignments. It can be argued that in the case of reads with the short length, 16S rRNA is more efficient to identify a higher level of taxonomic assignments such as phyla, classes, orders, families, and genera, than species or strains [27]. As an example, *Escherichia coli* is a bacterial species with several strains. Some of these strains have nearly similar 16S rRNA-encoding genes but have very dissimilar functional capabilities [28][29].

The application of next-generation sequencing technologies has provided a set of technical innovations called "Metagenomics" as a culture-free method to study the genetic content of all organisms in a community obtained directly from their natural environment. Debility of 16S rRNA to identify and especially rapid identification of microorganisms in the metagenomics reads is more visible.

One of the key elements of the medicine of the future is to bring the applications of sequencing technologies and metagenomics techniques to the routine medical laboratories. The new generation of sequencing technologies, especially Whole Genome Sequencing (WGS) and Single Cell Sequencing has provided this opportunity, but analyzing the massive amount of data that these machines produce is still expensive and requires a high effort of ultra-specialized personnel and High-performance computing resources.

The key contribution of this research is the development of an alternative fast and cost-effective method to allow identifying species and strains from raw read metagenomics sequencing data, regardless of aforementioned limitations and without further processes such as assembling and alignment. The second contribution of this research has tended to develop the required applications with the automated process in order to enhance the applicability of the analysis as a routine process for the research community, particularly in medical laboratories. The proposed methods in this doctoral research involve software and pipelines along with parallel and distributed computing applications such as Hadoop and Hive in a cluster of low-cost nodes.

The motivation behind this research is developing alignment-free approaches, not only to shortcut identification into a quick and accurate process using parallel and distributed computing on commodity hardware, but also for other purposes in bioinformatics and metagenomics studies such as the accurate estimation of microbial community composition based on metagenomics sequencing data, the alignment and assembly of short reads, and other Next-generation sequencing analysis.

1.2 Description of this Thesis Outline

This thesis is organized as follow.

Chapter 2 is intended to provide a brief summary of the basic biological concepts and methodology used in this research. Chapter 3 gives a background of the relative computer science concepts and methodology used in this research.

The aim of Chapter 4 is to outline the research approach, design and methods used in this research to find the best solutions for the following three primary questions that are the major computational challenges of high-throughput sequencing and metagenomics data analytics.

1. What is the proper alignment-free method for rapidly identifying the species and strains from raw read sequencing data?
2. What is the proper method of finding DNA signatures from genome databases?
3. What is the proper method to use less computational resources?

The initial section of this chapter proposes an alternative alignment-free method for real-time identification of species and pathogens from raw read sequencing data. The next section provides an overview of the state-of-the-art methods and discusses the challenges and limitations of these methods in comparison with our proposed method of using DNA signature as a solution for the first question.

The rest of the chapter describes a novel method of searching DNA signatures among target databases. In addition, we addressed the resources of the data used in this research and explained how to prepare the data for use in the proposed method.

The goal of Chapter 5 is to address the following questions:

1. What is the proper method for searching DNA signatures among the reads and matching with their related species?
2. What is the proper method to use less computational resources?

The chapter starts with the review of the more popular existing methods for short reads classification and discusses the challenges and limitations of the methods in comparison with our proposed methods of reads classification.

We have developed two methods to answer the questions. The methods that are presented in this chapter explained two different ways of matching the DNA signatures with the reads in metagenomics sequencing reads. The first method uses Bitmap Indexes and NoSQL and the second method uses a novel pipeline named SRIdent. Hadoop and Hive are used for both methods. In addition, we addressed the resources of the data used in this research and explained how to prepare the data for use in the proposed methods. Chapter 6 elaborates the results and discussion of the 3 methods explained in Chapters 4 and 5.

Chapter 7 as the concluding chapter is intended to summarize the main contributions of the research presented in this doctoral thesis. The methods and analysis developed to achieve the proposed objectives are presented in a much-summarized way and the main conclusions derived from the results are presented. Finally, the future research lines that give continuity to the presented work are listed.

Chapter 2

Basic Biological Concepts and Related Challenges

2.1 Next-Generation Sequencing (NGS)

The first DNA sequences were obtained in the early 1970s. Frederick Sanger, the pioneer of developing a rapid DNA sequencing published a reliable and reproducible method (chain termination) for DNA sequencing in 1977 [30]. A decade later, several new methods for DNA sequencing were developed and "first-generation" sequencing automated instruments were invented. The appearance of next-generation sequencing (NGS) technologies is indebted to the invention of massively parallel sequencing techniques in 2000 [31].

The recent technologies have revolutionized sequencing capabilities. They allow us to sequence DNA and RNA much more quickly and cheaply than the previous sequencing methods. As an example, the first human genome, finished in 2003[32][33], required 13 years to sequence and cost nearly 3 billion dollars. In contrast, new sequencing technologies can sequence over 45 human genomes in a single day for approximately $1000 each (Illumina documents that is addressed in Figure 2.1 caption). Figure 2.1 shows a dramatic increase in the volume and concurrent decreasing the cost of sequencing data since 2000.

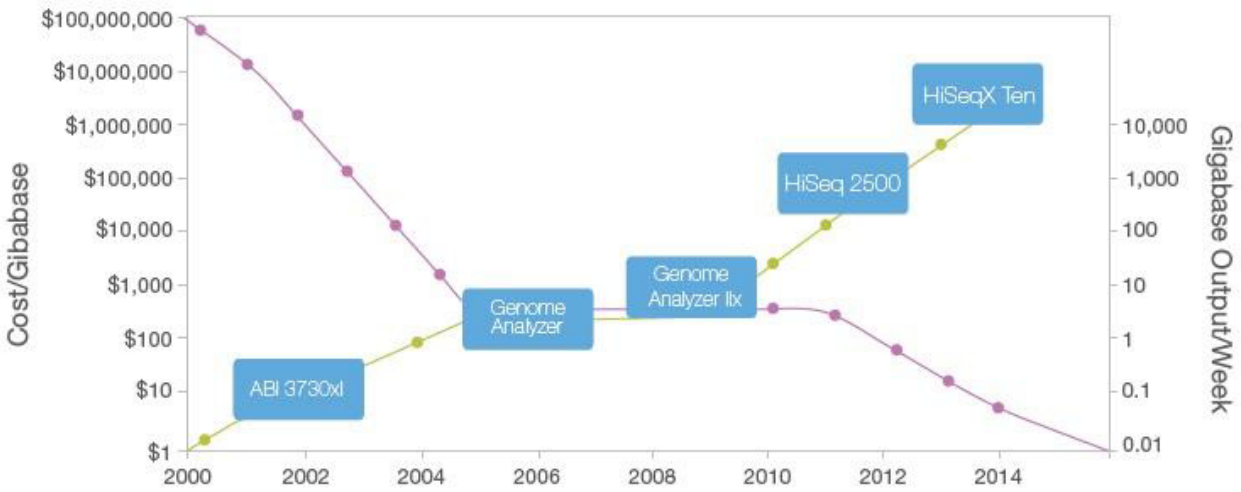

Figure 2.1. Sequencing Cost and Data Output Since 2000. The Y-axes on both sides of the graph is logarithmic.

Source: https://www.illumina.com/content/dam/illumina-marketing/documents/products/illumina_sequencing_introduction.pdf

Next generation sequencing technology offers a wide range of options for sequencing, such as 454 Life Sciences (Roche), Illumina, Ion Torrent (Life Technologies), Applied Biosystems (Life Technologies), and Pacific Biosciences among others. Each platform offers different coverage and reads length and the cost per base of sequencing is likely to become more affordable with the rapid advances in this field. Along with the ability to generate community sequence data, the price of DNA sequencing continues to fall. It allows the microbial community to be investigated at a much greater scale and detail than before.

Today's sequencing technologies generate an enormous amount of sequencing data (1.8 terabases) in a single run. Ever-increasing and complexity of generated sequences have large implications for the analysis of this data. Therefore, developing fast and cost-effective means and applications to accelerate sequencing analysis is necessarily required. The first question after obtaining raw sequencing data is “who's there?”

2.2 Short Reads

Reads generated by sequencing technology are short fragments of a longer DNA molecule present in the sequencing sample. The length of a short-read is normally less than 1,000 base pairs [34].

The sequencing read length depends on sequencing technology and chemistry methods used. The major limitation of Next-generation sequencing (NGS) methods is the short length of reads [35].

Repetitive sequences limit the capability of the short reads to assemble complex regions of the genome [36], and it requires de novo assembly before most genome analysis. When the reads are longer, there is more overlap between reads and more overlap with the reference genome.

Recent progress in long-read DNA sequencing, so called the Third-generation sequencing allows direct sequencing of single DNA molecules to produce longer reads than second generation sequencing [37].

The term "Raw sequencing reads" that is used in the title and several pages of this dissertation refers to the primary output of sequencing technologies (short reads) without any pre-processing.

2.3 Metagenomics

The increase of the sequencing capacities from the emergence of NGS technologies has opened a new way not only to taxonomic analysis but also to obtain a complete catalog of genes that an organism can express at any point in its life cycle. Hence, the NGS has had a great impact in many fields of the current modern biology allowing the simultaneous study of several microorganisms of interest.

Metagenomics, a high-throughput culture-independent technique has provided the ability to investigate the entire community of microorganisms of an environment with analyzing their genetic content obtained directly from their natural residence. The method does not have to worry about problems such as culture-negative, gram-negative, gram-positive, and so on. This promising ability has led to growing the integration of different fields of science, along with sequencing technology and computational approaches to investigate the mysteries of the hidden world of microbes and to reveal the secrets of these invisible and unknown inhabitants of our planet. It allows scientists to gain a more comprehensive picture of the diversity and function of microbial communities.

The appearance of metagenomics techniques, as we know it today was first reported by Stein et al., who revealed the presence of two unknown genes in archaea with sequencing 40-kb archaeal genome fragment in a metagenomics library [38]. The term "metagenomics" was first used by Jo Handelsman in 1998 [39]. Since then, many metagenomics projects have

been organized from ecology, environmental sciences, human health (e.g., the human gut microbiome), and chemical industry [40].

According to the definition of the GOLD database (Genomes Online Database), there are three types of ecosystems for obtaining Biosamples: Environmental ecosystems (e.g. water samples taken from deep ocean), host-associated ecosystems (e.g. samples taken from human, animals, and plants), and engineered ecosystems (e.g. Bioreactors).

The first step in a metagenomic study is to obtain the sample. Long DNA molecules extracted from the sample are broken into smaller pieces by special fragmentation and cloning techniques. Then, these small pieces are fed into the sequencer for determining the order of nucleotides in short fragments of DNA [41]. Sequencing output for a metagenomics sample is enormous data sets containing the short reads of hundreds to thousands of known and unknown organisms. The DNA extracted should be representative of all cells present in the sample and sufficient amounts of high-quality nucleic acids must be obtained for subsequent library production and sequencing [41] [42]. Figure 2.2 details the steps involved in a typical sequence-based metagenomics project.

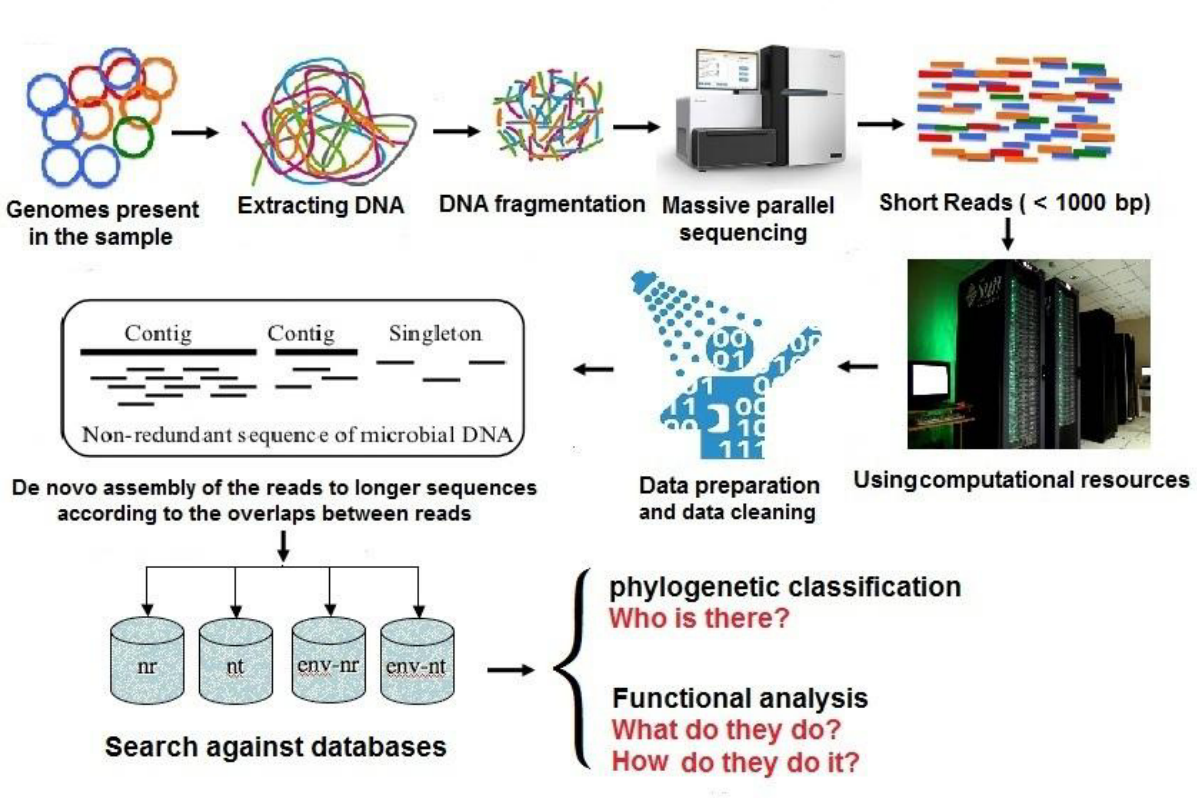

Figure 2.2. A typical metagenomics project.

There are two basic types of metagenomics studies: function based metagenomics and sequence-based metagenomics. Functional analysis of metagenomics data provides valuable insight into the kind of biological functions that are performed by organisms in a given environment. It involves searching for a particular function or activity. Sequence-based metagenomics involves sequencing and analysis of DNA obtained from the samples. It can be used to assemble genomes, phylogeny reconstruction and taxonomic assignments, identify genes, and compare organisms of different communities.

The term metagenomics can also be classified into two distinct methods: shotgun metagenomics and targeted metagenomics.

Shotgun metagenomics also named whole metagenomics usually refers to extracting and sequencing the entire genomic content of a sample from the complex community. This sequencing approach provides an accurate and deep understanding of both taxonomic and

functional diversity of microbial communities. Due to the complexity and the size of the data, shotgun metagenomics is still very expensive and the informatics data analysis is a challenging issue [43].

Targeted metagenomics, [44][45] or amplicon-based metagenomics is an approach for constructing gene collections which are useful for studying the composition of gene clusters in microbial communities and the adaptive evolution of enzymes toward environmental changes. In compare with shotgun metagenomics, the amplification of the gene regions before sequencing reduces the amount of data in the targeted metagenomics. Since, sequencing and analyzing the data are easier, faster, and cheaper, it can be integrated into the routine processes of laboratories.

2.3.1 Challenges of Metagenomics Approaches

Metagenomics as a newbie and immature field of science involves with too many challenges from sampling to sequence assembly, data annotation, and functional analysis. Metagenome samples come from complex communities of microorganisms, sometimes containing more than 10,000 species [46]. Due to the diversity of the microbial communities, most genomes are not completely represented by short reads.

Managing, mining, and analyzing of the massive amount of metagenomics data poses great challenges and it requires the use of specialized bioinformatics tools. The sequence-only method is comparatively less time-consuming than the alternative, which is the construction of metagenomics libraries and subsequent function and or sequence-based screening to identify gene products encoded by the target microbial partners [47].

Having efficient implementations to facilitate the analysis is urgently required in both biological and computational parts of any metagenomics project.

Currently, the traditional culture-based and biochemical testing techniques are the majority of diagnostics assays in clinical laboratories. Along with the fact that these techniques are time-

consuming and labor intensive, also failing the isolation of disease-causing organism is unavoidable [48]. The dramatic drop in the price of sequencing is going to be a great opportunity in clinical laboratories to apply culture-independent metagenomics techniques for clinical diagnostics assays [49][50].

Well-established standard molecular biology protocols for sequencing of the metagenomics samples are available to most research and clinical laboratories. However, they are often limited by computational resources and methods to analyze these data and it is still far from being standardized [51].

2.4 Sequence-Based Identification

After sequencing of the genetic material, phylogenetic assignment and taxonomic affiliation of the species is the first thing that should be considered so that we can look into the mysterious world of the DNA sequences in order to figure out who is there.

There are several methods of species identification that are commonly used. The most popular methods are identification based on morphological characteristics of the species and sequence-based identification to distinguish species.

The traditional identification methods based on morphological, physiological (nutrient need), and structural (differences in membrane lipids) characteristics are unable to study culture-negative microorganisms, especially in complex microbial communities. For all of them, cultivation was an essential prerequisite, but the current means of cultivation did not allow even a small part of existing microorganisms to be cultivated (for various reasons such as lack of knowledge of the required nutrients, atmospheric conditions, etc.). Therefore, species identification has to shift towards genetic identification regarding the advances in sequencing technology. Sequence-based identification includes two different categories of alignment-based and alignment-free approaches.

2.4.1 Alignment-Based Identification

Since the origin of bioinformatics, sequence analysis has been the forefront of the biological research. The very first approaches for sequence analysis were based on sequence alignment. It is the procedure of comparing two (pairwise alignment) or more (multiple alignments) sequences to find the most significant similarities. There are two common computational approaches for sequence alignment: local alignment and global alignment. [52].

Local alignment searches regions of similarity between substrings of the query sequence and the target sequence. The Smith-Waterman algorithm [53] is the most commonly used for local alignments. Instead of searching the whole sequence, the algorithm compares segments of all possible lengths and optimizes the similarity measure.

Global alignment performs an end to end alignment over the entire length of two or more sequences as the query and the target sequences. The sequences are assumed to be homologous along their entire length. Needleman and Wunsch algorithm [54] is an efficient algorithm to perform the global alignment. Figure 2.3 illustrates examples of local and global alignment.

Figure 2.3. Examples of local and global alignments.

Source: http://www.majordifferences.com/2016/05/difference-between-global-and-local.html#.WVdT-4SGPIU

2.4.1.1. The Limits of Alignment-Based Identification Methods

The applications of sequence alignment are limited to apply for closely related sequences, but when the sequences are divergent, the results cannot be reliable. Alignment-based approaches are computationally complex, expensive, and time-consuming and therefore aligning large-scale sequence data is another limitation [55].

The exponential growth of next-generation sequencing (NGS) data has resulted in the generation of a huge amount of data. This causes several challenges on alignment-based algorithms.

Raw reads as the output of NGS technology are short, partial, and large in volume with lots of errors and missing parts [16][41]. About 50% of the reads generated in one run cannot be perfectly mapped to the reference sequences [56]. Since the assembly of sequence data is based on the overlapping of the reads, assembly of the less diverse members of a community may require additional sequencing [57]. Alignment and assembly methods are inadequate when two reads from the same gene do not overlap [58][59]. If reads do overlap, there is no guarantee that they are from the same genomes [60][61].

These limitations and challenges cause incapability of alignment-based approaches or they make the mapping process, assembly, annotation, and the alignment-based identification as a prolonged, complex, and expensive process. This inability becomes more visible when the data comes from metagenomics sequencing, which is the sequencing of multiple species at the same time. Sequence reads come from different regions of various genomes and their alignment may not be possible.

2.4.2 Alignment-Free Identification

Since alignment-free methods could overcome various difficulties of traditional sequence alignment approaches, they are increasingly used in NGS sequence analysis, such as searching sequence similarity, clustering, classification of sequences, and more recently in

phylogeny reconstruction and taxonomic assignments [62][63]. Moreover, they are much faster than alignment-based methods that mostly run in linear time.

One of the most popular alignment-free methods is based on k-mer/word frequency. In this method small fragments of sequence called k-mers are used to search the frequency of all possible k-mers (words) in the entire sequence or in a database and then use statistical approaches and distance measurement based on these frequency vectors. "k" is a fixed length of the oligonucleotide to represent the sequence fragment [64][65][66].

There are some other alignment-free approaches:

- Methods based on substrings that search similarity and differences of substrings in a pair of sequences.
- Methods based on information theory that includes global and local characterization of DNA, RNA and proteins and region classification
- Methods based on graphical representation that uses for sequence comparison and characterization

In this research, k-mers/words frequency is considered to propose the methods of identification.

2.5 k-mers or Words Frequency

In computational genomics, counting k-mers (words of size k) refers to search the occurrence of all the possible substrings of length k in genome sequence data. It is an essential component of many alignment-free methods, including genome and transcriptome de novo assembly, sequencing coverage estimation, error correction of sequence reads, and metagenomics sequencing analysis. It can be used as a first stage analysis before alignment, assembly, and an early identification of the species in metagenomics samples. Comparing the frequency of k-mers in metagenomics sequences is computationally easier than sequence alignment and it is an important alignment-free method of sequence analysis [67][68].

Using k-mer spectra as a reference-free technique for analyzing next-generation sequencing data can reveal lower sensitivity to the depth of coverage, errors rate, the probability of reads contamination, the size of sequenced data, and estimating the level of repetitive elements of the sequence data. The key factor of using k-mers is directly analyzing of raw reads without the need for assembly.

2.6 De Bruijn Graph

It is an efficient way to represent all the possible k-mers (subsequences of length k) of a sequence. These types of graphs are important because of their usefulness in the reconstruction of genomic sequences and are used in most of the applications of de novo assembly of the short reads. The n-dimensional Bruijn graph of 'm' characters is a directed graph representing overlaps between sequences of characters. The graph has m^n vertices consisting of all possible k-mers of the given sequence of characters present in the reads sequence. For example, given the alphabet comprising A, T, G and C, there are 4^3 = 64 nucleotides of length k=3 [69][70].

In a string with length S, The amount of all possible k-mers is {S – k + 1}. As an example in Figure 2.4 the length of the sequence is S = 17, therefore the amount of all the possibilities of k=7 are:

S – k + 1 = 17 – 7 + 1 = 11

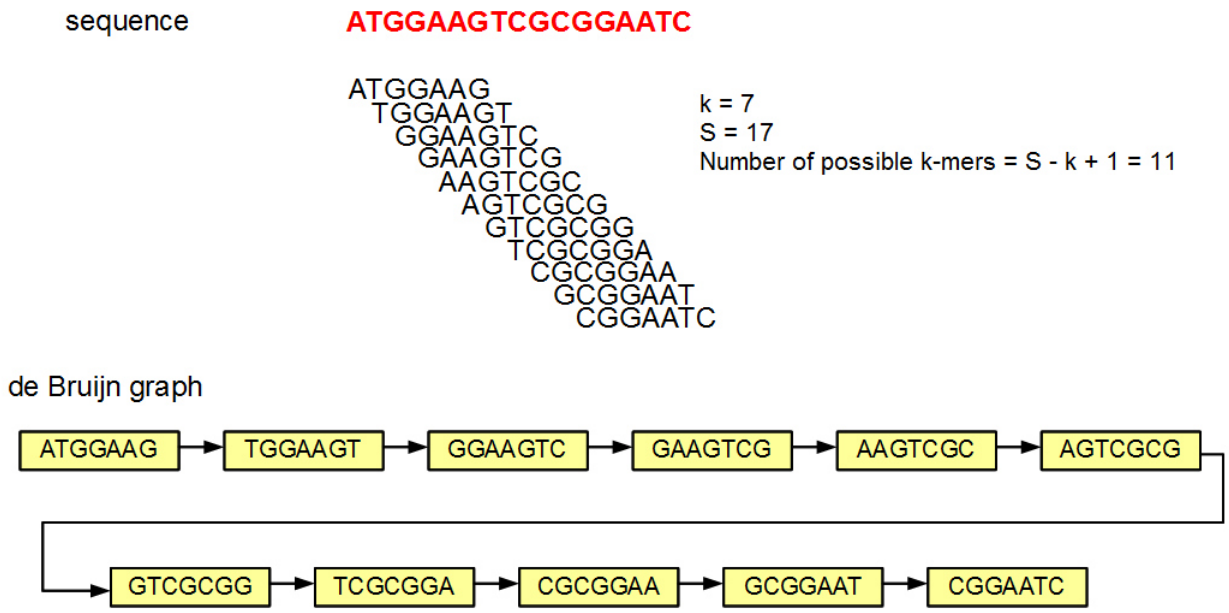

Figure 2.4. An example of de Bruijn graph.

Source: http://www.homolog.us/Tutorials/index.php?p=2.1&s=1

In our proposed pipelines explained in Chapters 4 and 5 of this thesis, there are two software called GkmerG and RkmerG that have been designed for generating all the possibilities of k-mers of the genomes and short reads. The method of generating k-mers of the two software is inspired from de Bruijn graph.

2.7 DNA Signature

DNA signature is a short k-mer oligonucleotide fragment with an arbitrary length k, which is unique or specific to a particular group of species selected from a target genome database. There are two categories of unique and common signatures according to the purpose of usage. The presence of a unique DNA signature in any volume of sequences and genetic materials represents the existence of the corresponding species[71][72][73]. Therefore, signature discovery is the action of finding specific fragments of the genome in a database. Any pipeline, application, or algorithm that is designed for DNA signature discovery has to

detect an entire database or multiple databases recursively. The procedure varies according to the purpose of using DNA signatures.

There exist methods for detecting signatures with different lengths of nucleotide for each species using k-mer frequencies and pattern comparison approaches. The number of DNA signatures for each species can differ from a few to millions of signatures. It depends on the number of species in the target database, the size of the genome, the length of the signature, and the similarity's degree of the genome with its relatives (strains) [71].

Using DNA signatures in the isolated sample studies and Polymerase Chain Reaction (PCR) base detection is easy to perform, because of few number of targets. But in the metagenomics studies, it is much more complicated. Taking into account the number of signatures, short reads and organisms in the metagenomics samples, it is obvious that we are facing massive data sets. Using ordinary hardware and software tools is impossible since it takes a long time regardless of any failure during the process.

DNA signature can help the identification process to be easier, cheaper, and faster. Using DNA signature can illuminate the dark part of sequencing to see who is there rapidly. It can be used as an application of early identification of the species from raw read sequencing data in metagenomics and next-generation sequencing (NGS) analysis. Unlike the alignment-based methods, the DNA signature identification method does not depend on the type of sequencing technology, the level of sequence coverage and the amount of sequencing required to obtain complete coverage. It is not also a serious matter if reads are short or long. According to the literature, DNA signature is mostly used to design primers for Polymerase Chain Reaction (PCR) assays and probes for DNA microarray assays [71]. Despite the rapidly increasing of completed genomes and with respect to the importance of DNA signature for real-time identification of species in metagenomics and clinical diagnostic assays, the term DNA signature has not been well considered in this area.

Despite the impact of the 16S rRNA on the microbial taxonomy, it is particularly useful for taxa above the rank of species. Because of sequence similarities, they are not sufficient to define bacterial species and strains [74].

There are two types of unique and common DNA signature. The unique DNA signature is a k-mer that occurs once in the target database and plays the role of fingerprint for microbial species. The common DNA signature is a k-mer that occurs among multiple species or strains and it can be collected by searching the shared regions through pairwise alignments between the input genomes. It is used for the design of microarray and PCR-based pathogen diagnostic assays [71].

2.7.1 Advantages of Using DNA Signature

A major advantage of the use of DNA signatures is the gene-independent and alignment-free nature of this approach [75]. As nucleotide signatures are generally pervasive across genomes [76][77], the requirement for the presence of conserved genes or motifs typically used for identification and classification of sequences is circumvented. DNA signature is an important method for classifying all genome sequence fragments independently of reference databases [77].

The GC (Guanine and Cytosine) percentage of the genome fragments is widely used as a measure of nucleotide composition [77]. Although the GC% is relatively constant within species, it varies widely between species. Therefore, GC% is suitable for tracing the origins of genome fragments within the species [78]. They are not suitable for assigning genome fragments of metagenomics sequences. As another advantage of DNA signature is it's higher effectively than GC% for assigning of nucleotide composition [79].

DNA signatures are well suited to the analysis of the sequences with lacking a robust estimate of phylogenetic relationships and to the analysis of complex sequences such as metagenomics sequences that alignment-based methods often perform poorly [77][80][81][82][83][84].

Sequence analysis of high-throughput technologies with DNA signature is easier than the other methods. The amount of DNA signatures and their specificity increase with adding the length of signature. It causes a wide flexibility of using the method [71].

Relative abundance estimation of microorganisms from the sequence reads in diverse communities, and searching for genetic variations that are known as genetic polymorphisms in eukaryotes are another potentially advantages of DNA signatures. Regarding a large number of DNA signatures in different species and the possibility to choose arbitrary lengths of them for identification, this approach is suitable, not only for PCR and microarray-based assays but also has great potential for next-generation sequencing analysis.

The flexibility to choose targeted and non-targeted databases and an arbitrary length of signatures are other advantages that allow reducing the cost of sequencing by performing lower-coverage sequencing. Since the length of signatures is short, the size of the reads does not a serious matter.

Detecting horizontally transferred DNA [79][85][86], reconstruct phylogenetic relationships [87], and infer lifestyles of bacteriophage [88] are among other advantages mentioned in the literature.

2.7.2 Disadvantages of Using DNA Signature

Although, it facilitates the analysis of sequence reads, searching and finding DNA signature itself in targeted databases is computationally intensive. Especial resources and applications are required. While adding the length of signature increases the specificity and the amount of DNA signatures, it increases also the difficulties and the use of more computational resources. Any change or update in the completed genome databases that are used as the target for finding DNA signature can change the amount of DNA signatures of a species in the database [71].

Chapter 3

Basic Computational Concepts

During the last decades, molecular biology has been revolutionized not only the development of increasingly rapid DNA sequencing techniques but also the computer technologies used to store, manipulate, distribute, and analyze huge amounts of biological information. It can be said that bioinformatics is the application of computer technology for managing and analyzing biological data. Bioinformatics is an area of interdisciplinary research that acts as a bridge between the biological sciences and computational sciences. The objective of the bioinformatics is to discover the important information that is hidden within huge volumes of data and get a clear view of the biology of organisms. Data mining that is the process of extracting significant information from large databases plays a significant role in bioinformatics. Data Mining is an interdisciplinary sub-field of computer science, involving methods at the intersection of statistics, machine learning, artificial intelligence, database technology, data visualization, pattern mining, and pattern recognition [89]. It includes the analysis of large data sets (e.g. genomics data sets) and the search for relationships among variables, through computationally intensive methods.

Mining such large high-dimensional and complex datasets in bioinformatics need intensive computational resources and applications. Parallel and distributed systems and computational methods have a pivotal role to overcome the challenges in bioinformatics analysis.

3.1 Distributed Systems

The development of distributed systems began around 1980 with the advent of Local Area Networks (LANs.). Distributed systems consist of a (possibly very large) network of stand-alone computers with different system software that communicate with each other, using a distribution middleware as illustrated in figure 3.1. Another definition is as follows: "A distributed system consists of a collection of computers that do not Share neither their physical memories nor their clocks" [90]. On the other hand, each computer of a distributed system has its own processor, local memory, and physical clock. In order to set up the

distributed system, different network technologies are used as well. An important motivation for the development of distributed systems is the sharing of resources [91].

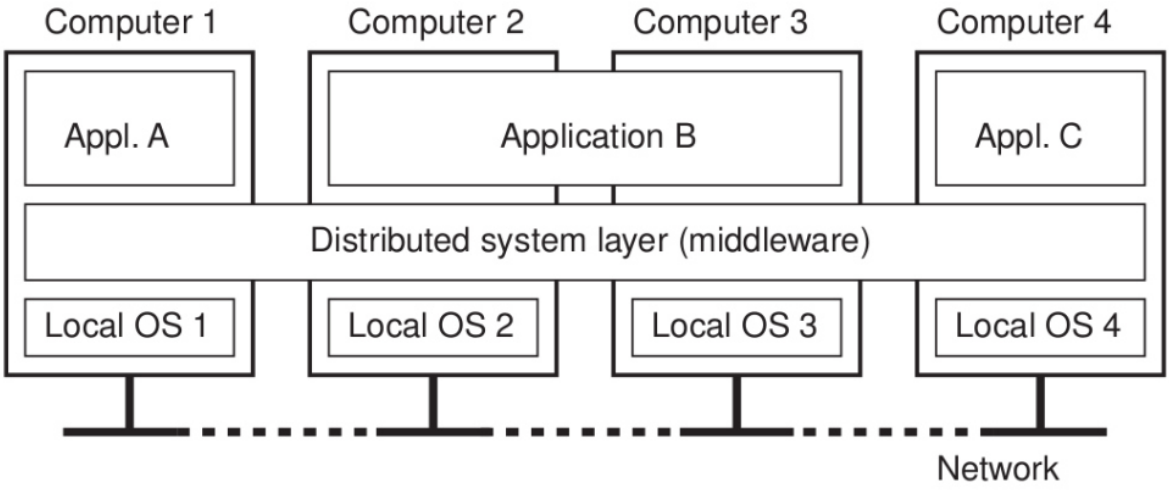

Figure 3.1. A distributed system organized as middleware. The middleware layer extends over multiple machines.

Source: http://www.ejbtutorial.com/distributed-systems/introduction-to-distributed-systems

The goals of distributed systems are making resources available, distribution transparency, openness (interact with services from other open systems, regardless of the underlying environment), fault-tolerant (In case of failure of one or more components, a fault-tolerant design allows a system to continue its function properly at a reduced level, rather than failing completely.), and scalability. A distributed system is more scalable than a single computer since the performance can easily increase by adding additional computers [92]. From the user's point of view, distributed system appears as a single coherent system, regardless of where and when interaction takes place and the communication components are mostly hidden from the users.

Distributed systems are the backbone of many applications such as World Wide Web, industrial control applications, ATM (cash machine) distributed databases, network computing, global positioning systems, air-traffic control, enterprise computing, office automation, and big data management and analytics. With the advent of mobile

communication networks and mobile systems, distributed systems are experiencing an intensive development in recent years.

3.2 Types of Distributed Systems:

3.2.1 Distributed Computing Systems

Distributed computing is designed to solve massive computing problems using a large number of computers organized in clusters that are embedded in a distributed telecommunications infrastructure. In distributed computing, components of a software system are shared among multiple computers to solve large problems with splitting them to the smaller problems in order to improve the efficiency and performance [93]. Grid, Cluster, and Cloud computing are the most common types of distributed computing systems.

3.2.2 Distributed Information Systems

An information system (IS) is a tool for gathering and communicating information for the purpose of meeting the needs of its users. An IS supports enterprise activities by providing the information it needs or by automating the activities associated with the activities. It includes all the resources available for gathering, managing, using, and distributing information within the organization. Typical applications for information systems can be found in almost all commercial and many technical fields; for example reservation systems, banking systems, telephone companies, and engineering applications [94]. They have become an important factor in successful economic development and competitiveness of the businesses, both in the domestic and foreign markets [95].

3.2.3 Distributed Pervasive /Embedded Systems

Distributed pervasive systems try to bring the flexibility of information technology to entire aspect of daily life. Pervasive computing is a growing trend in connection with the Internet of

Things. More and more microprocessors are being implemented in everyday objects so they can communicate information. Thus, computing is everywhere and ubiquitous. Devices for pervasive computing are always connected and permanently available. This technology has focused on wireless technologies, advanced electronic devices, and the Internet. The goal of the researchers in the area of pervasive computing is to create intelligent products that are connected to the Internet and their generated data are easily available [96][97].

3.3 Parallel and Distributed Computing

Parallel computing is a form of processing that allows many computing devices to work simultaneously to solve a problem [98][99].

Traditionally, parallelism has been used in supercomputing centers to solve large problems, but in recent years, distributed systems have become increasingly important because of improvements in computer networks and the spread of multi-core processors. Clusters, Clouds, and Grids made possible the availability of distributed computing platforms at low cost. It is no longer exclusive to specialized environments and high-performance computing centers [100].

Efficient development of distributed and parallel applications is of crucial importance, because most of the systems currently being developed are distributed and parallel.

Many classification schemes have been proposed, but the most popular is the taxonomy of Flynn proposed by Michael J. Flynn [101], which is based on the way the data flow and instructions are organized. The central idea is based on the analysis of the flow of instructions and data, which can be simple or multiple, giving rise to the appearance of 4 types of machines. This classification is based on the number of flows of instructions and simultaneous data that can be handled by the system during the execution of a program. A flow of instruction is a sequence of instructions transmitted from a control unit to one or more processors [102].

The possible categories based on Flynn classifications are as follows:

3.3.1 Single Instruction Stream, Single Data Stream (SISD)

This represents the classic Von-Neumann machine, in which a single program is executed using a set of data specific to it. It is composed of a central memory where the data and programs are stored and a processor. It is shown in figure 3.2. This Platform can only give a kind of virtual parallelism through the paradigm of multitasking, in which the processor time is shared between different programs. Thus, more than parallelism, what this platform supports is a type of concurrency [103][104].

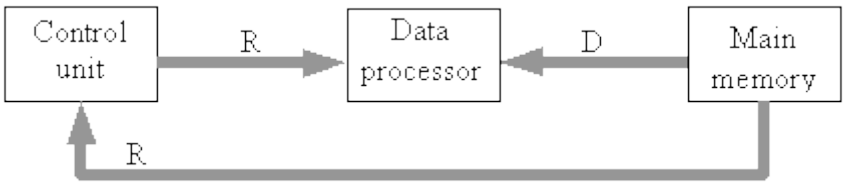

Figure 3.2. SISD Architecture. A single computer that has one control unit, one processor unit, and one memory unit.

Source: https://edux.pjwstk.edu.pl/mat/264/lec/main121.html

3.3.2 Single Instruction Stream, Multiple Data Stream (SIMD)

It is a class of parallel computers in which the array of processing elements executes the same operation on multiple data points at the same time. Processor arrays are typical examples of this kind of architecture. In these architectures, a controller receives and decodes sequences of instructions to execute, and then send them to multiple processors slaves. The processors are connected through a network. The data can be processed in a memory space which is common to all processors or in a memory space of each unit. Figure 3.3 illustrates SIMD architecture. All processors work with perfect synchronization. In this class, each processor is simply an arithmetic-logic unit and has a single unit of control. All processors execute each received operation at the same time, whereby each of them executes the same instruction on different data [105].

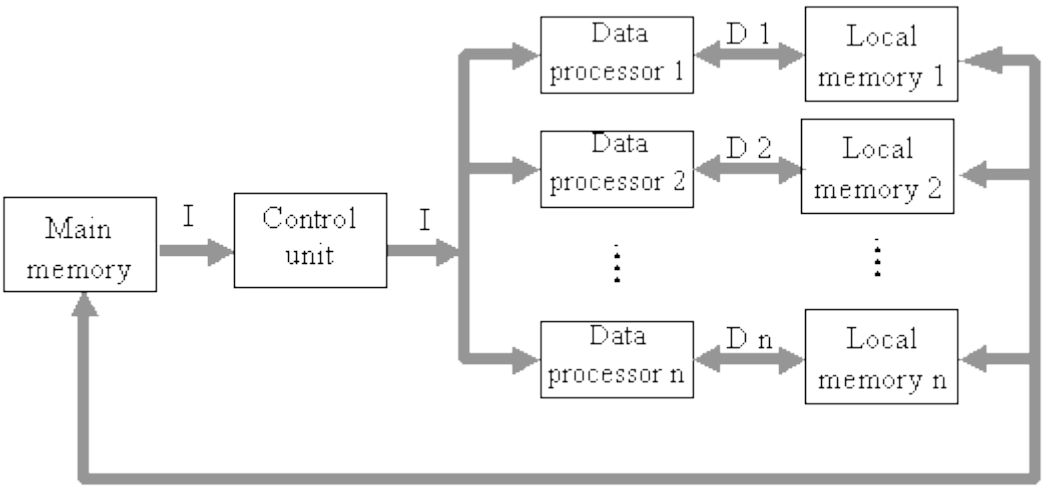

Figure 3.3. SIMD Architecture. Performing an operation on multiple data.
Source: https://edux.pjwstk.edu.pl/mat/264/lec/main121.html

This type of computing platform has been developed due to the large number of scientific and engineering applications that are well adapted to this class such as image processing, particle simulation, finite elements, molecular systems, etc. In this class there is a simple sequence of operations, operating at the same time on a set of data.

3.3.3 Multiple Instructions Stream, Single Data Stream (MISD)

It is a system with multiple instructions that operate on a single data stream [106]. All processors receive instructions individually from their own control unit and operate on a single data flow in accordance with the instructions they have received from their respective control units. These processors operate simultaneously. This architecture is known as a systolic array or array of processors for pipeline execution of special algorithms. An MISD computer is similar to a segmented superstructure, in which each processor performs a part of the operation on the data flow. MISS computers contain multiple control units, multiple processing units, and a shared memory unit, as it is shown in Figure 3.4.

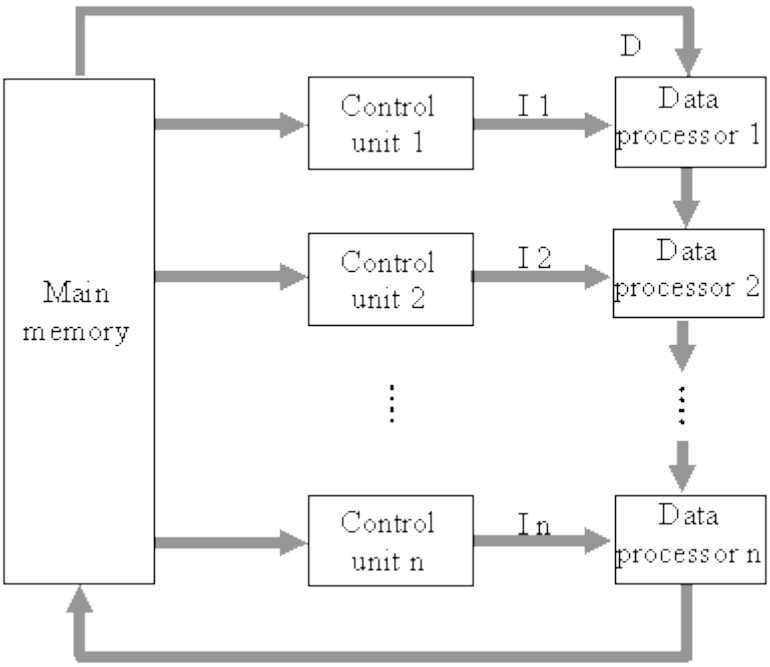

Figure 3.4. MISD Architecture. Performing multiple operations on a single data flow (assigned to processors from 1 to n).

Source: https://edux.pjwstk.edu.pl/mat/264/lec/main121.html

3.3.4 Multiple Instructions Stream, Multiple Data Stream (MIMD)

It is the most general model of parallelism and because of its flexibility, a wide variety of parallelism can be exploited. The basic idea is that multiple heterogeneous tasks can be executed on multiple data at the same time and each processor operates independently with occasional synchronizations with others [107]. It is composed of a set of processing elements where each one performs a task independent or not, with respect to the other processors. The form of programming usually used is a concurrent type, in which multiple tasks can be executed simultaneously. Many multiprocessor systems and multi-computer systems fall into this category. The MIMDs can be classified into two types of distributed memory and shared memory [108]. Figure 3.5 is a type of MIMD system with shared memory.

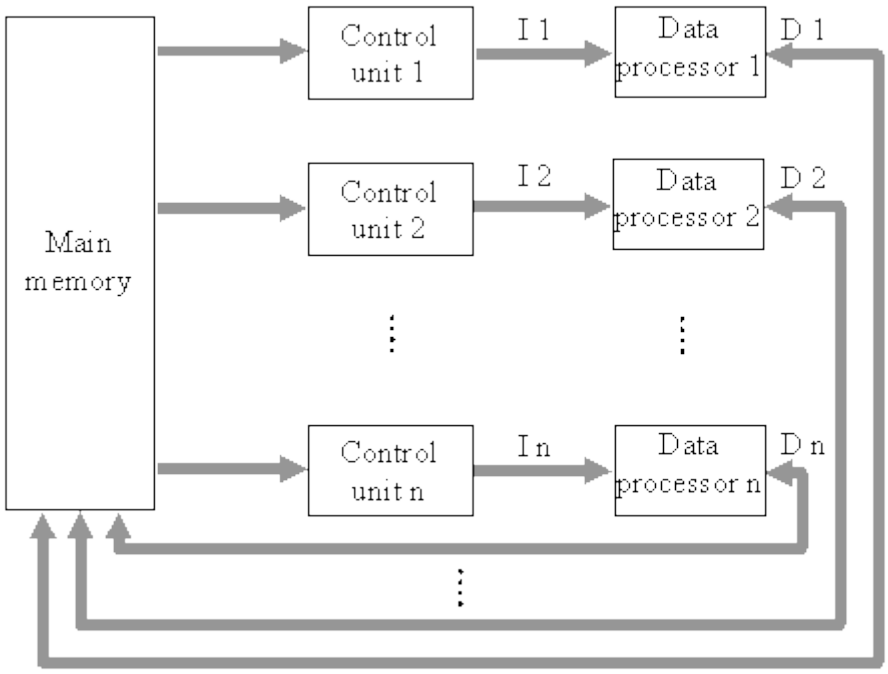

Figure 3.5. MIMD architecture with a shared memory (assigned to processors from 1 to n). Source: https://edux.pjwstk.edu.pl/mat/264/lec/main121.html

3.4 Parallel Programming

Parallel computing is a programming technique in which many instructions are executed simultaneously. It is based on the principle that large problems can be divided into smaller parts that can be solved concurrently and in parallel. It focuses on the distribution of data in parallel among the different computational nodes. Parallelization often leads to perform similar sequences of operations (not necessarily identical) or functions that are performed on the elements of a large data structure [109][110].

The computational needs of many applications require the development of efficient and secure software for multiprocessor platforms. Since the rise of multi-core processors and computer networks has increased; parallel programming is required to use parallel and/or distributed systems efficiently. Parallel software programs are more difficult to write than sequential ones [111].

Hadoop, a parallel and distributed processing infrastructure and MapReduce programming models are typical examples of parallel and distributed computing.

3.5 The Apache Hadoop

The dramatic increase in the amount of data in various fields of science revealed the inadequacy of existing ordinary computers for big data analytics. It has prompted the developers to compose tools and applications using parallel and distributed computing that could be applicable on commodity hardware. Big Data defines the set of new technologies and business applications for the management of massive data in large structured and unstructured databases. The Apache Hadoop project has been designed as an open source and Java-based software framework for parallel and distributed computing on large datasets using commodity hardware. Hadoop allows running simple programming models on large structured and unstructured datasets across an arbitrary number of nodes in a cluster. From the user's point of view, it looks like a single computer [112][113][114].

A Hadoop cluster has a single master and several slave nodes (Data-nodes) that are connected to each other through Secure Shell. It can run on a single node or multi-node cluster with thousands of nodes. Hadoop separates and distributes the files that contain the data automatically, as well as splitting the job into smaller tasks and executing them in a distributed way and recovering from failures automatically and transparently to the user. Although, the function of Hadoop is defined as a distributed system with multi-computer nodes, the ability of Apache Hadoop to use MapReduce for parallel processing of large data sets is an extra power to let even a single-node run program on the large data sets, larger than its memory and CPU capacity. Hadoop handles any types of data from structured, unstructured, text files, log files, images, audio files, communications records, etc. The Hadoop core has two primary components: Hadoop Distributed File System (HDFS) and MapReduce.

3.5.1 Hadoop Distributed File System (HDFS)

HDFS [115] is the data storage part of Hadoop. It is designed to store and high-throughput access of large datasets across multi-nodes of clusters. HDFS breaks down the data into small block-sized chunks, which are stored as independent units. HDFS provides input data storage for the MapReduce framework. It is a highly fault-tolerant distributed file system designed for use on low-cost hardware.

HDFS has three main components [41][116]:

1. **Name-Node**: HDFS has master/slave architecture. The cluster consists of a single Name-Node; it is the master of the file system. It is responsible for managing the blocks in Data-Nodes and maintains the metadata and indexes of the blocks, but not the data itself.
2. **Data-Nodes**: They are the workhorses of the file system. Name-Node breaks down data into block-sized chunks, which are stored as independent units in Data-Nod.
3. **Secondary Name-Node**: It keeps a copy of the merged namespace image, which can be used in case of any failure for the Name-Node.

3.5.2 MapReduce

MapReduce [117] is a programming model for parallel and distributed processing. MapReduce works by breaking the processing into two phases: the map phase and the reduce phase. The Map phase processes a set of data in parallel and returns it as an intermediate result and then the Reduce phase reduces it to a smaller set of data. Each Map and Reduce works independently. The process is illustrated in Figure 3.6. In fact, MapReduce decreases the input large amount of raw data into useful data [118][71].

In a Hadoop cluster, the input data is in the form of a file or directory and is stored in the Hadoop file system (HDFS). The MapReduce paradigm sends the computational process to the data-node where the data will be processed. When a MapReduce process is launched, the

tasks distributed among the different servers in the cluster. The input file is passed to the map function line by line. The mapper processes the data and creates several small pieces of data. The reduce stage processes the intermediate values coming from the map to produce the final output of MapReduce. The Hadoop framework itself manages the sending and receiving of data between nodes. Most of the computation happens at data-nodes (where the data is placed) in order to minimize network traffic. Once all the data has been processed, the user receives the result of the cluster. The two main components of MapReduce are:

1. **JobTracker:** As the Master of the system, it is responsible for managing the map and reducing tasks.
2. **TaskTracker:** As the slave, it receives the mapper and reducer task from JobTracker and returns the results to the JobTracker after execution.

Hadoop is highly faulted tolerant. In order to prevent any failure in the process, HDFS creates multiple copies of data through the blocks, 3 copies by default. Name-Node can detect any failure in Data-Nodes or blocks and JobTracker also can detect any failure of TaskTrackers and will replace them [41].

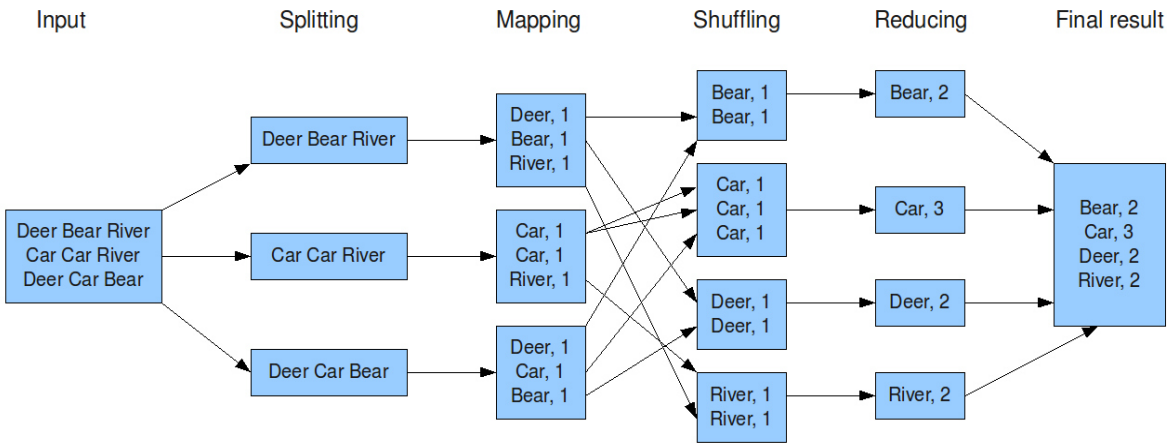

Figure 3.6. An example of the overall MapReduce WordCount process.

The original image was made by Trifork.

Source: https://cs.calvin.edu/courses/cs/374/exercises/12/lab/

3.5.3 Hadoop YARN (Yet Another Resource Negotiator)

As another component of the second-generation Hadoop 2 released by the Apache software foundation, YARN (Yet Another Resource Negotiator) was added in order to upgrade scheduling, resource management, and execution in Hadoop. YARN is a cluster management technology. It consists of a central resource manager and a manager for each node, which handles controlling a single node [119][120].

YARN has two main components shown in Figure 3.7: a central resource manager that monitor how applications use the Hadoop system resources and several node managers (one for each node) that monitor the processing operations of individual nodes in the cluster. Separating HDFS from MapReduce with YARN makes the Hadoop environment more suitable for operational applications that cannot wait for batch jobs to be completed [121].

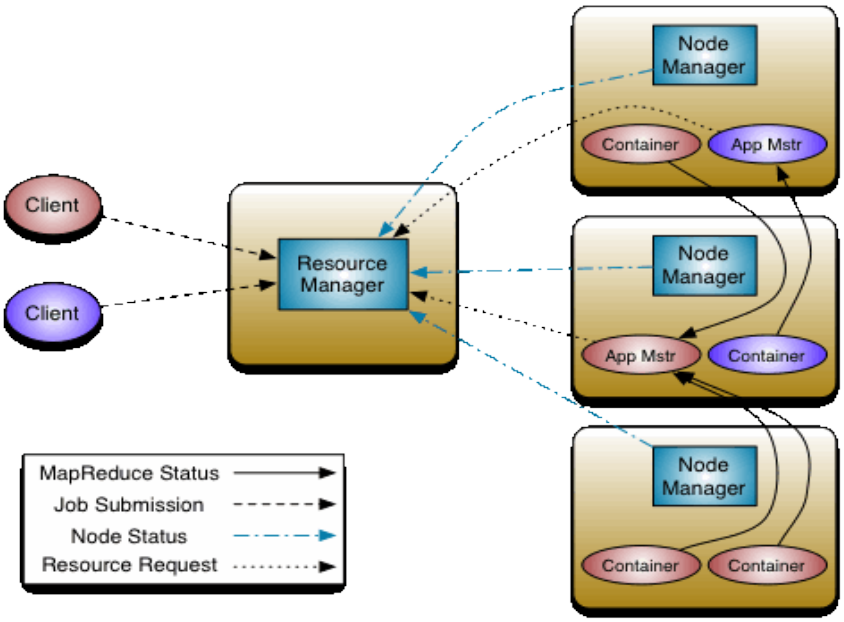

Figure 3.7. YARN architecture.
Source: https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html

3.5.4 Hive

The Apache Hive [122][123][124] is a data warehouse infrastructure on the top of the Hadoop MapReduce framework. It is designed to query a large dataset that is stored in the HDFS using an SQL-like language called HiveQL. Hive is designed to write once and read several times. Real-time queries and row-level update are not possible. Facebook Data Infrastructure Team started to create Hive in January 2007 to bring the familiar concepts of tables, columns, partitions, and a subset of SQL to the unstructured world of Hadoop and it was open sourced in August 2008 [125]. Traditional relational databases require the data to be in a structured format, while Hive can handle both structured and unstructured information. It lets the user process large datasets with relatively little effort and in a reasonably short time. Hive can handle querying on billions of rows in a table or multiple tables.

3.5.5 NoSQL

"NoSQL" Stands for Not Only SQL. The term "NoSQL" was used by Carlo Strozzi for the first time in 1998 [126]. It is a non-relational database. One of the aspects of NoSQL is its ability to handle database analytics of big data sets in parallel and distributed platforms like Hadoop on commodity hardware. Hive and Hbase are types of NoSQL applications on top of the Apache Hadoop file system. NoSQL databases can handle unstructured data such as text files, log files, email, social media, and multimedia. Horizontal scaling is one of the most important features of NoSQL databases and allows us to add more nodes to our distributed system. Vertical scaling only allows increasing the power of existing machine [41][127][128].

Chapter 4

Proposed Method of DNA Signature Discovery

The aim of this chapter is to outline the research approach, design, and methods used in this study to find the best solutions for the following three primary questions that are the major computational challenges of high-throughput sequencing and metagenomics data analytics.

1. What is the proper alignment-free method for rapidly identifying the species and strains from raw read sequencing data?
2. What is the proper method of finding DNA signatures from genome databases?
3. What is the proper method to use less computational resources?

The initial section of this chapter proposes an alternative alignment-free method for real-time identification of species and pathogens from raw read sequencing data. The next section provides an overview of the state-of-the-art methods and discusses the challenges and limitations of these methods in comparison with our proposed method of using DNA signature as a solution for the first question.

The rest of the chapter describes a novel method of searching DNA signatures among targeted databases and non-targeted databases. In addition, we addressed the resources of the data used in this research and explained how to prepare the data for using in the proposed method.

4.1 Proposed Alternative Method of Identification

In Chapters 1 and 2, we discussed the importance of real-time identification of species and pathogens from raw read sequencing data, particularly the metagenomics data, which comes from sequencing of very diverse and complex microbial communities. Since the traditional identification of species based on morphological, physiological, and structural characteristics are unable to identify culture-negative species, identification and taxonomy of organisms would no longer depend on morphological characteristics (visible characters), but only on DNA sequences. We discussed the two common types of sequence-based identification methods, the alignment-based and the alignment-free methods. The inability of alignment-

based approaches for rapid identification purposes caused a necessity for shifting into the alignment-free approaches as an alternative method.

Among the alignment-free approaches, the most popular options are to use marker genes such as the 16S rRNA gene. For many years, 16S rRNA was used as the primary tool for taxonomic assignment of bacterial species. 16S rRNA enables identification of a few numbers of targeted species using PCR or targeted metagenomics techniques. Although the progress of the next-generation sequencing has greatly increased the depth of sequencing coverage, the length of the reads is very short. The shortness of the reads, along with other limitations caused by sequencing technologies (mentioned in Chapters 1 and 2), reveals the fact that 16S rRNA genes are inefficient agents for many of real-time analysis of short reads generated from high-throughput sequencing tools; De novo assembly is required to obtain longer sequences from the reads.

The exponential increase in the number of completed genomes in last two decades represents a growth of more than 100-fold for each decade. There were only two completed bacterial genome sequences in 1995 [129]. A decade later, the number of completed genomes increased to 300 and only two metagenomics projects were published in 2006 [130]. Based on NCBI[1] genome database, the number of sequenced bacterial genomes increased to more than 30,000 and thousands of metagenomics projects were published according to the GOLD[2] database by the end of 2014 [129].

The completion of numerous whole genome projects and availability of completed genome databases in recent years has provided a novel opportunity, in order to find alternative taxonomic solutions than using 16S rRNA gene. Due to the complex sequences from diverse communities of microbes, 16S rRNA gene comparison has to be replaced by more comprehensive approaches.

1- The National Center for Biotechnology Information
2- Genomes Online Database

As defined in Chapter 2, DNA signature is a short k-mer oligonucleotide fragment with an arbitrary length k which is unique or specific to a particular group of species selected from a target genome database. There are two categories of unique and common signatures according to the purpose of usage. The presence of a unique DNA signature in any volume of sequences and genetic materials represents the existence of the corresponding species. Therefore, signature discovery is the action of finding specific fragments of the genome in a database [131].

Any pipeline, application or algorithm which is designed for DNA signature discovery, has to detect an entire database or multiple databases recursively. The procedure varies according to the purpose of using DNA signatures.

It was previously used to design primers and probes for PCR and microarray assays. However, the completion of tens of thousands of bacterial genomes in recent years has provided the opportunity to use DNA signature for rigorous analysis of metagenomics sequences.

The primary goal of this research is proposing an alternative fast and cost-effective method to allow a rapid identifying the species and the strains from raw read metagenomics sequencing data, regardless of aforementioned limitations. In this research, we have proposed to use DNA signature as a tool to facilitate and speed up the taxonomic assignment of microorganisms in high-throughput sequencing analysis and metagenomics.

4.2 Overview of the Related State-of-the-Art Methods

Several tools and algorithms of DNA signature discovery have been proposed in the literature in order to facilitate the design of microbial and pathogen-based diagnostic assays; notable instances are discussed next.

TOFI [1132] is designed to identify DNA fingerprints of a single genome as suitable probes for microarray-based diagnostic assays. It utilizes the whole genome of the pathogen instead of the special gene (such as 16s rRNA) or special regions of the genome for designing probes

[133]. In order to design DNA microarray probes, TOFI reduces the solution space by discarding DNA sequences that are common to the target sequence and one or more phylogenetically close sequences. Then, each extracted DNA microarray probe is compared with all DNA sequences from the chosen reference database [132].

TOPSI [134] is a pipeline for real-time PCR signature discovery. TOPSI detects common signatures among multiple strains of bacterial genomes by collecting the shared regions through pairwise alignments between the input genomes. It is an extended version of TOFI [134][135].

Insignia [136] provides unique signatures that can be used to design primers for PCR and probes for microarray assays. It has two main components: the web interface and the computational pipeline. The computational pipeline uses grid computing and an algorithm to perform pairwise alignment of every pair of target genomes and background genomes for their comparison. Insignia provides signatures which are unique to the background genome database (BLAST database) [137]. In fact, when a user adjusts the desired options in the Insignia web interface, a query runs on the database that contains the results of DNA signature discovery which has already been provided.

TOFI, TOPSI and Insignia use the open source software MUMmer [138] that implements a suffix tree-based algorithm for comparing genomic sequences [132][134][136]. It is a package for the alignment of very large DNA and amino acid sequences. Furthermore, these three pipelines use BLAST for the evaluation of signatures regarding specificity.

CaSSiS [139] is an algorithm for detecting signatures with maximal group coverage within a user-defined specificity range for designing primers and probes. It provides signatures for single or group organisms in hierarchically-clustered sequence datasets. This algorithm calculates the Hamming-distance between a signature candidate and its matched targets. CaSSiS uses the rRNA sequences provided by the SILVA database to create a signature collection for designing primers and probes.

The Consecutive Multiple Discovery (CMD) algorithm [140] is an iterative method including the parallel and incremental signature discovery (PISD) method as a kernel routine to discover implicit DNA signatures. PISD is a combination of the Hamming-distance-based algorithm, the IMUS approach [141] and Zheng's method (UO) [142] in terms of using the corresponding incremental and parallel computing techniques. PISD uses a mismatch tolerance and previously discovered signatures of specific lengths as candidates to find shorter signatures instead of scanning the whole database. CMD and PISD can find unique signatures for single sequences, but cannot search for signatures that are specific for groups [141]; they are designed to find signatures of sequences from Expressed Sequence Tags (ESTs) [1] databases.

The internal-memory-based unique signature discovery algorithm IMUS [141] is an improvement of Zheng's method [142] which is based on the Hamming-distance for detecting unique signatures. IMUS tries to discard similar substrings of a sequence in order to obtain the DNA signatures as unique fragments. PIMUS [143] is the improved version of IMUS. Both algorithms load the complete DNA database into the main memory to find unique signatures in ESTs datasets.

DDCSD (Distributed Divide-and- Conquer-based Signature Discovery) [144] applies a divide-and-conquer strategy for detecting DNA signatures. When the dataset is large and cannot be loaded into the memory all at once, the algorithm splits it into smaller segments which parts are loaded and processed one by one. The discovery node and the discovery routine are the main components of this algorithm. When the size of the dataset is larger than the available memory, the discovery routine splits the dataset into multiple parts which are processed one at a time by the discovery nodes. This algorithm is based on searching for similarities and mismatches in the patterns. Similarly to CMD, PISD, IMUS, and PIMUS, this algorithm is designed to search ESTs datasets, but it can process larger databases such as the human whole-genome ESTs database, as well.

1- Short fragments of mRNA sequences obtained by single sequencing of randomly selected cDNA clones. ESTs are mostly used to either identify gene transcripts or as an alternative cheap method of gene discovery and gene sequence determination

Jeliyfish [145] is an algorithm to count the k-mers in parallel. This algorithm implements lock-free hash table optimization for counting k-mers up to 31 bases in length.

There are other approaches to find signatures or probe sequences, such as PROBESEL [146], OligoArray [147], OligoWiz [148], YODA [149], PRIMROSE [150], and ARB-ProbeDesign [151]. All of them are limited to one selected target or single sequence in each run, thus, they are not applicable for large datasets [139]. Table 4.1 contains a comparison of the more popular DNA signature discovery methods.

Table 4.1. A comparison of signature discovery algorithms according to the data format, computational resources, and ability to process single or multiple sequences.

Name	Data Format	Adopted platform according to the publication	BLAST specificity	Ability for single sequence	Ability for multiple sequences
TOFI	FASTA	64 x 1.5 GHz Itanium 2 processors with 64 GB of shared memory	✓	✓	✕
TOPSI	FASTA	98-cores Linux cluster	✓	✓	✓
Insignia	FASTA	192-node Linux cluster	✓	✓	✓
CaSSiS	rRNA	Intel Core i7 CPU (4 cores, 2.67 GHz) with 24 GB of RAM	✕	✓	✓
CMD and PISD	ESTs	Dell PowerEdge R900 server with two Intel Xeon E7430 2.13 GHz quad-core CPUs, 12 GB RAM	✕	✓	✕
IMUS	ESTs	Intel 2.93GHz CPU	✕	✓	✕
PIMUS	ESTs	Intel Core i7 870 2.93GHz quad-core CPU and 16 GB RAM	✕	✓	✕
DDCSD	ESTs	A Master Node: Intel Core i7 CPU 870 at 2.93 GHz and 16 GB RAM, 10 Slave nodes: Intel Core i7 CPU 3770 K at 3.50 GHz and 32 GB of RAM for each one	✕	✓	✓

In practice, despite the respected efforts of above mentioned and other methods, there are still a number of limitations for DNA signature discovery.

Since most existing methods of DNA signature discovery require significant computational resources, they are not applicable to the entire research community. Due to the size of genome databases, the large amount of RAM and CPU capacity requirements and long execution times are major limitations of most of the above methods that are based on pattern

comparison and pairwise alignment of the genomes. The determination of the mismatch tolerance level as a discovery condition also influences the results.

In some cases, it is necessary to load the whole dataset into the main memory for searching for unique or common signatures. When the size of the data exceeds the available memory, the execution will fail. For instance, in IMUS, PIMUS, and Zheng's method, the entire database has to be loaded into the memory [144]. Thus, for such sequential algorithms like IMUS, increasing the number of CPU cores does not increase the discovery efficiency of the algorithm [143]. Another limitation for most of the above methods is the lack of efficiency to find both unique and common signatures simultaneously. Most of them are capable of finding only DNA signatures of a single genome. Another limitation of some of these methods is the lack of the possibility to select an arbitrary length (k) for the signatures.

The additional challenge as another major limitation for DNA signature discovery methods is the lack of option in the choice of target and non-target genome databases. TOFI, TOPSI, and Insignia use BLAST databases (such as nt and nr databases) as the background or non-target genomes for specificity evaluation of signatures and there is no option for the user to choose other target and non-target genome databases. As an example, in the Insignia web interface, the user receives a quick response without special requirements on local computational resources. However, this privilege comes with the restriction that there is no option to use other sequences as the target and background genomes because they are part of the Insignia database [71][144]. With the advancements in the sequencing technologies and the increasing number of completed genomes, whole genome shotgun sequences, and draft genomes, it is obvious that some of these signatures will not be unique later using BLAST specificity evaluation. This issue is a challenge not only for DNA signatures but also for all the sequence-based identification methods.

Geographical distribution and diversity of the species, ecological and chemical status, host and environmental factors, isolation or complexity of the samples, and many other factors can have a great impact on the selection of target and non-target genome databases for DNA signature discovery. When the absence of a considerable number of species in the sample is

evident, it seems quite questionable that we eliminate a large number of useful DNA signatures through their assessment and specificity evaluation against the entire background sequence databases such as BLAST. For instance, when we sure that in the sample there is nothing from Zebrafish, Muse, Chimpanzee, Black cottonwood, Macaca Fascicularis, etc. we do not need to check the uniqueness of our DNA signatures against their genome. Otherwise, we will lose a significant number of signatures.

Expressed sequence tags (ESTs) are short fragments of mRNA sequences obtained by single sequencing of randomly selected cDNA clones. ESTs are mostly used to either identify gene transcripts or as an alternative cheap method of gene discovery and gene sequence determination [152].

IMUS, PIMUS, CMD, PISD, and DDCSD are designed to scan ESTs sequences for the unique signatures. However, the ESTs represent only fragments of genes, not complete coding sequences [153]; therefore, many signatures are missed.

4.3 Proposed Method of DNA Signature Discovery

To overcome the aforementioned challenges of DNA signature discovery, we proposed our method as a powerful pipeline. The pipeline HTSFinder (High-Throughput Signature Finder) has been designed in order to enhance the usability of DNA signatures for massively high-throughput sequencing analysis.

The pipeline HTSFinder has significant advantages compared with the DNA signature discovery pipelines and algorithms described above.

- First, HTSFinder is capable of detecting all unique, common, and maximal group coverage signatures of the entire database or multiple databases simultaneously.
- Second, it becomes possible to select target and non-target genome databases, based on user requirements. For instance, we have the ability to use both forward and reverse-complement genome sequences of a database for detecting DNA signatures.

- Third, the pipeline can consider either a cluster of low-cost computer nodes that are commonly available in research facilities or a high-performance computer (HPC).
- Finally, the flexibility of the different phases of the pipeline makes it suitable for other bioinformatics and metagenomics studies such as Next-Generation Sequencing (NGS) analysis.

HTSFinder is very efficient and powerful with high accuracy for both unique and group-specific signatures without discarding even a single signature from the database, except those ones that contain IUPAC (International Union of Pure and Applied Chemistry) nucleotide codes such as K, M, N, R, S, W, Y, etc. Our GkmerG (Genome k-mer Generator) component will remove any k-mer containing IUPAC nucleotide codes after generating the k-mers. In this pipeline, there is nothing to worry about the mismatch tolerance and complexity of comparison and pairwise alignment search methods.

4.3.1 Description of the HTSFinder Pipeline

HTSFinder consists of three computational phases as shown in Figure 4.1. This pipeline generates all the possibilities of k-mers for every genome individually and then determines their frequency in the entire database. Finally, DNA signatures of every species or strain are obtained in the database or multiple databases that have been involved in the pipeline. HTSFinder implements the parallel and distributed computational tool Hadoop for the second and third phases.

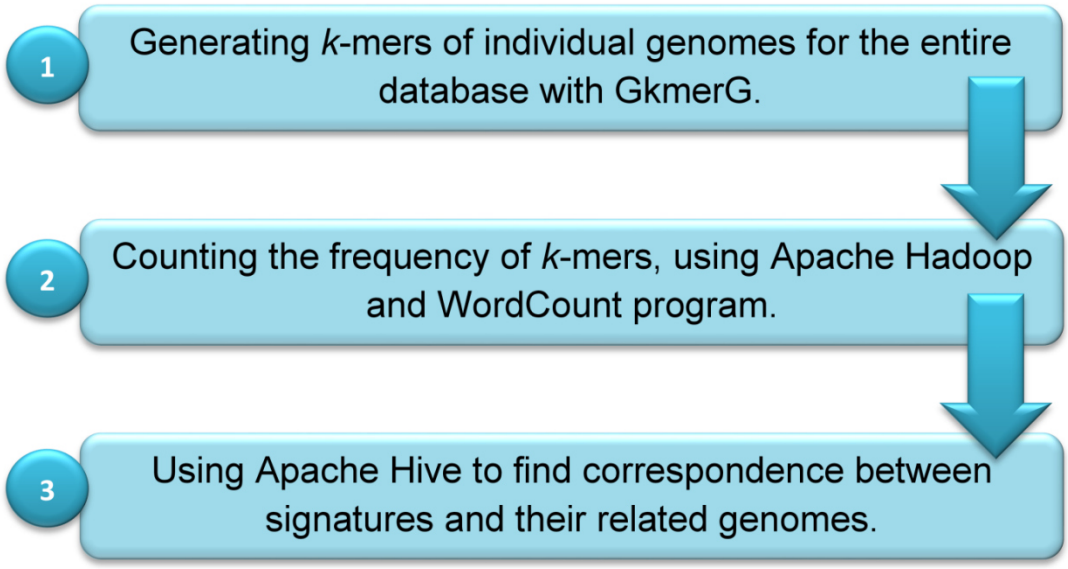

Figure 4.1. The three main phases of HTSFinder for detecting DNA signatures. We can repeat the second phase with the obtained results if required.

Source: **Karimi R. et al:** Journal of Evolutionary bioinformatics 2016.

4.3.2 Data Preparation

The first phase of the pipeline is carried out by GkmerG that is designed to obtain all the possibilities of k-mers of genome sequences with FASTA format[1] (*.fna or *.fa). The method of generating k-mers in this software is inspired from De Bruijn graph[2].

This software tool removes the remarks of the genome and splits it to the specific length k. Then, it eliminates the k-mers that contain IUPAC nucleotide codes and every subsequence of length less than k which has remained from the end of the sequence after splitting. Figure 4.2 illustrates the split of the genome by GkmerG. Concatenating the files, sorting k-mers and removing all duplicates except one are the last steps of GkmerG. For the species with multiple chromosomes and some bacterial genomes that are comprised of multiple chromosomes [154] and plasmids, GkmerG concatenates them into a single file before sorting at the end of the first phase.

1- It is a text-based format for representing nucleotide sequences using single-letter codes.
2- Explained in Chapter 2.

GkmerG copies the original database into another directory as the reference database with appending a number to the beginning of every species name in it, to simplify future data management. Once we get the output of the first phase for a database, we can keep it forever. In the case of any update in the database, we need only to repeat this phase for the updated or new genomes, not for the whole database. The output of GkmerG is the input for the second phase in the pipeline described in the next section.

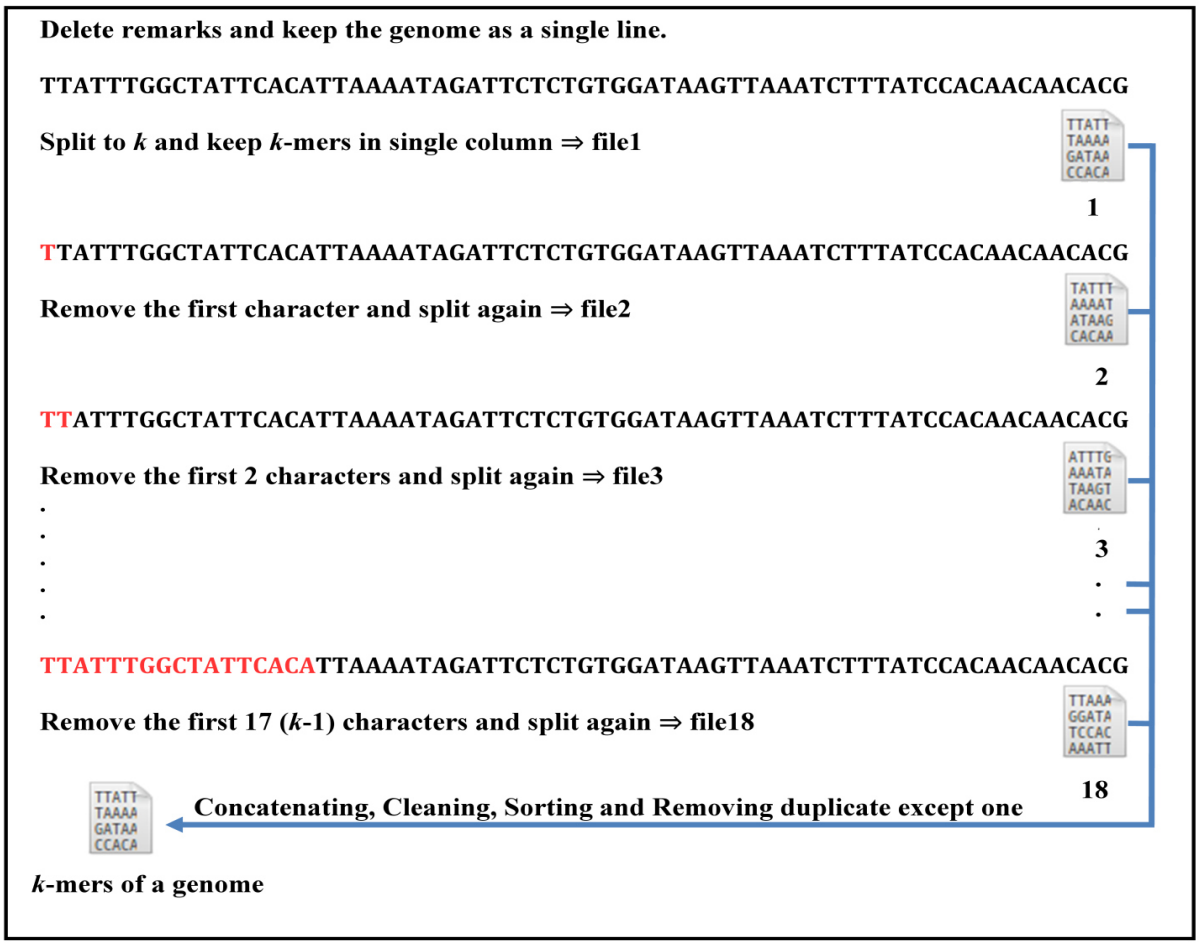

Figure 4.2. Splitting of the genome by GkmerG for k = 18 to get all the possibilities of 18-mers. Generating k-mers for a single genome with GkmerG includes purgation, splitting, concatenation, cleaning, sorting, and removing duplicate except one. The output of GkmerG is a file containing k-mers of a genome in a single column. The labels above the file numbers in this figure represent the beginning of four k-mers in the head of files.

Source: **Karimi R. et al:** Journal of Evolutionary bioinformatics 2016.

In this research, we used the Hadoop framework and WordCount program to calculate the frequency of k-mers in very large genome datasets. In Hadoop 1.2.1 and earlier releases, the JAR (Java Archive) file of WordCount is also included.

4.3.3 Pipeline Implementation

In the second phase of the pipeline, we copy all the output files of the first phase to the HDFS (Hadoop Distributed Filesystem) and run the WordCount program.

The result of this phase is a large file containing sorted and a non-duplicate list of k-mers obtained from the files generated in the first phase in one column and another column containing the frequencies of k-mers among genomes of the database. A k-mer with a frequency value 1 means that this k-mer is a unique substring, so appeared only in one of the species in the database. These occurrences are primarily what we are looking for as unique DNA signatures. The value in front of a k-mer indicates the number of genomes (species) that contain the given k-mer. Table 4.2 shows a portion of the Hadoop and WordCount output. For instance, the 18-mer with frequency=8 in the first row of Table 4.2 means that this 18-mer occurs in 8 genomes among the 2,773 bacterial ones, while the 18-mer in the fourth row is a unique signature in the database. In the second phase of the pipeline, we can extract all unique signatures or group-specific signatures due to the frequency, but we cannot determine the owner of the signatures.

Table 4.2. An example of Hadoop and WordCount results in the second phase.

Signatures or 18-mers	Frequency in the database
AAAAAAAAAAAAAAAGAG	8
AAAAAAAAAAAAAAAGAT	25
AAAAAAAAAAAAAAAGCA	20
AAAAAAAAAAAAAAAGCC	1
AAAAAAAAAAAAAAAGCG	5
AAAAAAAAAAAAAAAGCT	6
AAAAAAAAAAAAAAAGGA	9
AAAAAAAAAAAAAAAGGC	3
AAAAAAAAAAAAAAAGGG	6
AAAAAAAAAAAAAAAGGT	38

Once we execute the second phase for a database we can use the results in the future until the next update of the database. However, as a difference from the updatable first phase, in the case of any update in the database, we have to repeat the second phase for the entire database.

When there are multiple target and non-target databases, it is possible to merge all of them in the pipeline, but as the input grows larger, it requires far more computational resources. As a suggestion, it is better to implement the first and second phases for every database, separately. With respect to the WordCount function that discards repeated k-mers and keeps only one in the output, we can reduce the size of output files and also the execution time. Then, we can merge the output of the second phase for all the databases and repeat the second phase with WordCount in Hadoop, one more time. In this case, we have a shorter process. Moreover, for the future execution, we can select the output files of the second phase as the candidate of their corresponding databases. In this case, we will not need to perform the first phase of the pipeline for the previously processed databases and we can repeat the second phase for target and non-target databases with merging the smaller files. For instance, the output of the first phase for the bacterial genome database resulted in a file with 177.35 GB of 18-mers. However, in the second phase, the size of this file was reduced to 103.03 GB that contained all the candidates of 18-mers in the database without any repeat. We can use this file as the candidate of bacterial genome database for further processing. Figure 4.3 illustrates the process of finding DNA signatures of the target database among non-target databases.

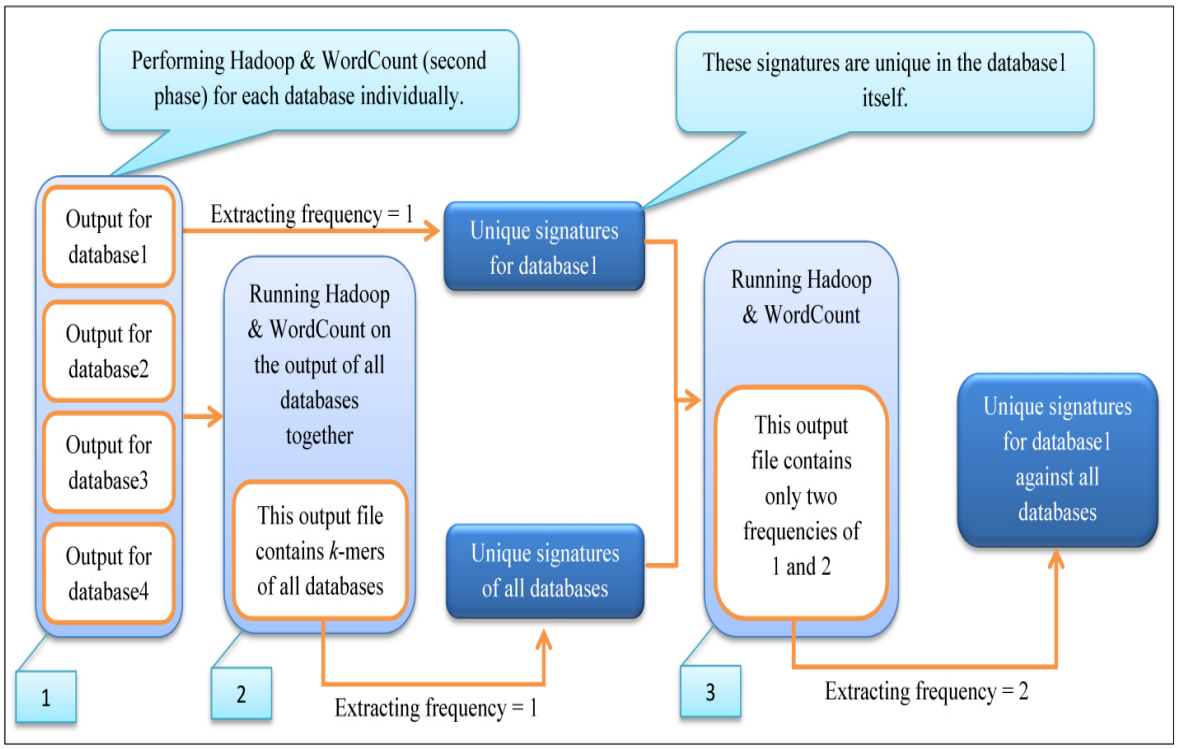

Figure 4.3. The recommended process for detecting unique DNA signatures of a target-database against non-target databases.

Source: **Karimi R. et al:** Journal of Evolutionary bioinformatics 2016.

In step 2 of the figure, the frequency number of k-mers varies from 1 to n, where n is the total number of the databases that are used in the pipeline. Since there are four databases in this figure, the frequency of k-mers in step 2 is from 1 to 4. In step 3, there are two input files with the list of non-repeated k-mers; therefore, the frequency of k-mers in the output is 1 or 2. Hence, k-mers with frequency 2 that are common in both input files are the unique signatures of database 1 against all databases.

The input for the third phase of the pipeline is the output of the first and second phases. The proper steps of the third phase are described in the next section.

The Apache Hive lets the user process large datasets with relatively little effort and in a reasonably short time. This research proves the efficiency of Hive to handle querying on billions of rows in a table or multiple tables. With HiveQL we can extract whatever we need from the results of the second phase of the pipeline. We can extract all the unique signatures

of a specific species in the database or extract group-specific signatures which are common between 2, 3, 4, etc. Due to the flexibility of querying in Hive, there are various ways to create the tables and design the queries in the third phase. Optimization techniques are very effective for reducing the time-consumption and computational resource usage in a Hadoop cluster. Running time, CPU usage, memory usage, and speed of disk reading of the nodes are the subjects of optimization in a Hadoop cluster. Managing the maps according to the size of data and memory is critical. To get a better and faster operation, optimizing the configurations and parameters of Hadoop is also required in order to reduce the data transfer and communication between nodes of the cluster [155]. Moreover, query optimization and designing the tables in Hive for preventing a repetition of queries are very important.

After loading data into the tables created with Hive, we can use queries such as SELECT and JOIN to extract relationships. We should create two tables with Hive: one for the output of the first phase and another one for the complete or a special part of the output of the second phase. With considering the ability of Hive to query very large tables and in order to prevent repeating the queries, we added a column containing the reference number to the files from the first phase. For example file 1 contains 18-mers from the first species in the database, so we inserted a column containing reference index 1 before all 18-mers in this file. Then, we merged all the 2,773 files in a single large one (220.35 GB) with two columns of k-mers and their related reference numbers. The reference number indicates the number that has been appended to the name of species by GkmerG in the first phase.

There are several options to create the table from the output of the second phase: one is to create the table without making any changes in the output. Another one is to break down the output to smaller groups according to the targeted signature. For example, if we are looking for the unique signatures, it would be better to extract only 18-mers with frequency=1. However, if we are looking for a common signature, then it would be better to extract the 18-mers with a specific frequency number such as 2, 3, 4, etc. In order to have a faster and easier implementation with Hive and later steps, we recommend the second option.

4.4 Selected Sequence Databases

4.4.1 Bacterial Genome Database

To prove the efficiency of our proposed method, the bacterial genome database with 2,773 completed genomes in FASTA format (*.fna) were downloaded from the NCBI database. The size of this database is 9.7 GB after decompression.

4.4.2 The Reverse-Complement Bacterial Genome Database

Another database that we used in this research was the Reverse-Complement Bacterial Genome Database. The revcom.pl 1.2 [available at http://code.google.com/p/nash-bioinformatics-codelets/] is a Perl program written by John Nash [Copyright (c) Government of Canada, 2000-2012]. We used this program to provide the Reverse-Complement sequences for the whole bacterial genome database.

4.4.3 Human Genome Database

The whole Human Genome is another database used in this research. The Homo sapiens hs-ref-GRCh38 sequences in FASTA format (*.fa.gz) were downloaded from the NCBI database. The size of the genome was 2.9 GB after decompression.

Chapter 5

Proposed Methods of Short Reads Classification

The goal of this chapter is to address the following questions:

1. What is the proper method for searching DNA signatures among the reads and matching with their related species?
2. What is the proper method to use less computational resources?

We have developed two methods to answer the questions. These methods that are presented in this chapter explained two different ways of matching the DNA signatures with the raw reads in metagenomics sequencing data. The first method uses Bitmap Indexes and NoSQL and the second method uses a novel pipeline named SRIdent. Hadoop and Hive are used for both methods. In addition, we addressed the resources of the data used in this research and explained how to prepare the data for use in the proposed methods.

Notably, we had to check the effectiveness of DNA signature before proposing a method to find it; therefore, we have done these two methods before proposing the HTSFinder pipeline (presented in Chapter 4) and DNA signatures that are used in these methods are obtained from existing databases (Insignia database).

5.1 Overview of the Related State-of-the-Art Methods

Listed below are the three more popular and newer metagenomics reads classifiers that work based on the k-mers frequency search methods and comparing genomic k-mers of the reads to find the matches. These methods are closer to our method for short reads classification with matching the DNA signatures, reads and their related species.

Kaiju [156], is a metagenomics reads classifier which finds the maximum (in-)exact matches on the protein level using the Burrows–Wheeler transform of the protein database. Kaiju translates metagenomics sequencing reads into the six possible reading frames and searches for maximum exact matches of amino acid sequences in a given database of annotated proteins from microbial reference genomes.

Kraken [157] uses an alignment of k-mers to classify small subsets of metagenomics data. Kraken makes an index of all k-mers found in the reference genomes (e.g. bacterial genomes) and matches the k-mers found in the reads to this index and then assigns the read to the taxon with most matching k-mers by following a path from the root in the evolutionary tree.

CLARK [158] can classify metagenomics reads based on reduced sets of k-mers. CLARK builds a large index containing the k-mers of all targets sequences. Then it removes any common k-mers between target sequences. In fact the remaining k-mers are the unique k-mers in the target sequences. Then it searches the matches of the k-mers of the reads with the highest number of indexed k-mers.

These reads classifiers are mostly attempting to speed up the process of matching k-mers with reads and they did not explain too much about the methods of collecting DNA signatures. For most of them, an external tool such as BLAST or MEGABLAST is required. Kaiju, Kraken, and CLARK use BLAST databases (such as nt and nr databases) for their specificity evaluation. Table 5.1 contains some more details on the algorithms and applications described above.

Table 5.1. A comparison of metagenomics reads classifiers according to the data format, computational resources, and ability to process single or multiple sequences.

Name	Data Format	Adopted platform according to the publication	BLAST specificity	Ability for single sequence	Ability for multiple sequences
Kaiju	FASTA	HP Apollo 6000 System ProLiant XL230a Gen9 Server, with two 64-bit Intel Xeon E5-2683 2 GHz CPUs (14 cores each), 128 GB DDR4 memory	✓	✓	✓
Kraken	FASTA	48 AMD Opteron 6172 2.1 GHz CPUs and 252 GB of RAM, running Red Hat Enterprise Linux 5.	✓	✓	✓
CLARK	FASTA	Dell PowerEdge T710 server, dual Intel Xeon X5660 2.8 GHz, 12 cores, 192 GB of RAM	✓	✓	✓

Using expensive computational resources is the other important issue regarding these methods. As an example: CLARK needs a maximum 165 GB of RAM [158] and the construction of Kaiju's index from the protein sequence database needs peak memory usage of 24 GB using 25 threads, Kraken 165 GB, and Clark 152 GB [156].

5.2 The First Proposed Reads Classifier Based on Bitmap Indexes and NoSQL

Using DNA signatures in the isolated sample studies and Polymerase Chain Reaction (PCR) base detection is easy to perform, because of the low number of targets. But in the metagenomics studies, it is much more complicated. Taking into account the number of signatures, short reads, and organisms in the metagenomics samples, it is obvious that we are facing massive data sets. Using ordinary hardware and software tools is impossible since it takes a long time regardless of any failure during the process. The proposed method in this section shows how parallel and distributed computing and Bitmap Indexing technique can solve this problem.

In this method, we use optimization techniques borrowed from database technology, namely bitmap indexes. They are used to speed up searching and matching of billions of DNA signatures in the short reads of thousands of different microorganisms, using commodity High-performance computing, such as Hadoop MapReduce and Hive.

Bitmap Index [159][160] is an efficient way to speed up the queries and improve performance in data warehouse environments, which contain tables with low cardinality columns. As the example given in Table 5.2, we index the values of the column Grade having low cardinality. In this case, our index has the same number of rows and the number of columns is equal to the number of distinct values in column Grade. In Table 5.2, the cardinality of the column Grade is 4 because we have 4 different values in it.

Table 5.2. An example of a bitmap index defined on Grade column.

RID	Name	Nationality	Grade
1	John	FRANCE	B
2	Sara	USA	D
3	Piter	RUSSIA	C
4	David	ENGLAND	A
5	Tania	GERMANY	B
6	Daniel	POLAND	A
7	Tom	CANADA	C
8	Robert	ITALY	C
9	Jain	FRANCE	D

RID	A	B	C	D
1	0	1	0	0
2	0	0	0	1
3	0	0	1	0
4	1	0	0	0
5	0	1	0	0
6	1	0	0	0
7	0	0	1	0
8	0	0	1	0
9	0	0	0	1

We downloaded DNA signatures of 100 bacteria from the insignia database. Insignia (introduced in Chapter 4) is a pipeline to generate unique DNA signatures and it is also a database and web application for obtaining DNA signatures. It detects signatures for designing primers in Polymerase Chain Reaction (PCR) and probes in microarray technologies. The signatures can also be used for real-time identification of species in microbial and viral assays. As we are in the testing process, we just downloaded the signatures with the length of 18 bp. Table 5.3 is an example of *Acholeplasma laidlawii* DNA signatures downloaded from the Insignia database.

Table 5.3. A partial view of unique DNA signatures (18-mers), downloaded from the Insignia database.

Insignia V0.7
Signatures calculated: Thu Mar 6 2014 05:55:14
Reference Organism:
Acholeplasma laidlawii PG-8A
Target Organism(s):
Signatures:

Index	Start	Stop	Strand Sequence
1	14151	14168	TATCAACAGGAGACATGG
2	1283370	1283387	GTACTACGACAACAATCG
3	406523	406540	TTGGTATTGGTTGGGTAG
4	1161523	1161540	TAAGCCACCTTCACCATT
5	631946	631963	GTGACCAAATCAGTGATG
6	1297841	1297858	CTTGGCCATCAAAACCAG
7	587419	587436	AGCAACTGATGCAGATGA
8	457243	457260	TGCTGCACCAGATCTATC
9	60205	60222	TGATGCTGCAGCAGGTGC
10	602186	602203	TCAAGGTTATGGTGGTGT

5.2.1 Short Read Simulator Application

Metasim [161][162] is a short read simulator application for genomics and metagenomics studies. It can be a great help to develop and improve metagenomics tools, and for planning metagenomics projects. Metasim can simulate the short reads of Roches 454 pyrosequencing, Sanger sequencing, and Empirical sequencing technology. Roches 454 pyrosequencing simulation is used for this study. The output of Metasim is a compressed file containing the short reads of a bacterial chromosome or its Plasmids begins with their header information as illustrated in Figure 5.1.

```
>r16.1|SOURCES={GI=11497281,bw,1947919816}|ERRORS={8_1:C,46: ,135_1:T,160_1:A,190_1:G}
|SOURCE_1="Borrelia burgdorferi B31 plasmid cp32-8" (44840ff90be8dcf7b704d6908ca095d559d2949e)

TTTAGGATTCGTACCCGTTTTCTTCTAATTTTTTCCTAGTGTTGTATGAATTTCTTTTAATTTTTTTTGTTTTTC
TTTCATGCAAGATTTTTTTATATTGAATTTTTTTATTAGGGCAATTTCATTTTGTTTTAAGTATATTTATTGCC
TCAATCTTAGTATACTTTATCAATATTTAAATACAAAATAGAAAGGAGCTTCTTCCGTTTTAAAGTTACAAT
TATTGAAATAATTTCTTAGTTGATATTTTTCTATTTCTTTAATCTTTCTTTCTTCTTTTATATTATTTTTATTAC
TTAAACACTCCACTGAATTTACACTACCATTTTTAGAAACATTG
```

Figure 5.1. A partial view of Metasim reads.

Source: ***Karimi R. et al :*** *Lecture Notes in Computer Science, 2014.*

We chose 100 bacterial genomes for simulating the short reads. We divided the bacteria with signatures into two groups of 50. The first group is common in 100 chosen bacterial genomes. There are no any common bacteria from the second group.

Before any implementation, some pre-processing is needed. We need to attach the short reads from all bacterial chromosomes and Plasmids as one file, remove the breaks between lines of the short reads and keep each of them in a single line. From the signatures, we need just the signatures of every bacterium as a single file. We should remove all extra information, in order to have smaller data size and shorter execution time. The pre-processing is done with bash script programming in Linux.

5.2.2 The Use of the Bitmap Index

Bitmap index techniques are used to create the index table by searching the existence of signatures in short reads. '1' represents the existence of the signature in the short reads and '0' represents non-existence. This process is done with Java programming.

As it is shown in Table 5.4, the index table can be created in two ways. The first is to keep every single signature as a column and put '0' and '1' depending on the existence of this signature in short reads. In this case, considering the number of signatures and reads, huge storage is needed.

Table 5.4. An example for our index tables; each column of these tables is kept as a single file.

RID	Reads	b1	b2	b3	b4	b5
1	R1	0	0	0	1	0
2	R2	1	0	0	0	0
3	R3	0	0	0	1	0
4	R4	0	0	0	0	0
5	R5	0	0	1	0	0
6	R6	0	0	0	0	1
7	R7	1	0	0	0	0
8	R8	0	0	0	0	0
9	R9	0	0	0	0	0
10	R10	1	0	0	0	0

RID	Reads	b1						
		s1	s2	s3	s4	s5	s6	s7
1	R1	0	0	0	0	0	0	0
2	R2	0	0	0	0	0	0	1
3	R3	0	0	0	0	0	0	0
4	R4	0	0	0	0	0	0	0
5	R5	0	0	0	0	0	0	0
6	R6	0	0	0	0	0	0	0
7	R7	0	0	0	1	0	0	0
8	R8	0	0	0	0	0	0	0
9	R9	0	0	0	0	0	0	0
10	R10	0	0	1	0	0	0	0

Another way is to keep every bacterium as a column. We store '1' if any signature of the bacteria exists in a short read, '0' if not. In this case, the table is much smaller. The number of columns is equal to the number of bacteria plus two more columns, one for row identification and the other for short reads. The number of rows is equal to the number of short reads. We can easily use Linux command to put all the files together as a single file. As an example, in Table 5.5 we have 6 files. One file contains the reads and their identification numbers and the other five contain '0' and '1' for five bacteria.

Table 5.5. An example of input table for Hive, that is created by merging files to a single file.

1	R1	0	0	0	1	0
2	R2	1	0	0	0	0
3	R3	0	0	0	1	0
4	R4	0	0	0	0	0
5	R5	0	0	1	0	0
6	R6	0	0	0	0	1
7	R7	1	0	0	0	0
8	R8	0	0	0	0	0
9	R9	0	0	0	0	0
10	R10	1	0	0	0	0

Creating the tables and executing the Hive queries are the next step to get the results. It is explained in details in the related paper [41].

5.3 The Second Proposed Reads Classifier Based on SRIdent Pipeline

In the method based on using Bitmap indexes, although the running time of the queries is very short, constructing the index files for each bacterium is time-consuming. This problem becomes more visible when a large number of bacterial species are considered (such as the real metagenomics samples). Therefore, we were motivated by the need to make an extra effort to come up with better and more creative solutions to address this problem and it resulted in the creation of the pipeline, named SRIdent (Short Read Identifier) [163] that is explained in this section.

This pipeline is based on generating k-mers from the short reads and searching the existence of DNA signatures in the Reads k-mers, by using Apache Hive data warehousing. RkmerG (Read k-mers Generator) is a software program presented in this study, for producing k-mers of the short reads, in order to use in the pipeline. The purpose of this study is to identify the species in a sample, directly from the raw reads without assembling and alignment.

5.3.1 Pipeline Description

The SRIdent pipeline consists of two computational stages. Data preparation is the first stage. DNA signatures with specific length (k) of every individual known species and short reads generated by sequencing technologies are two types of data that are used in this pipeline. According to the length of DNA signature that we use in the pipeline, we should produce the same length of k-mers from the short reads. RkmerG (Read k-mer Generator) is designed to generate all the possibilities of k-mers of the short reads with any pre-defined length of k. The method of generating k-mers in this software is inspired from De Bruijn graph.

Figure 5.2 shows the RkmerG function. This software removes the information line of the reads and changes each read to a single line, and then copies each line (read) for k times and splits these k lines as illustrated in figure 5.2. It splits the first line as the complete read to k-mers, then eliminates the characters in red color from the rest of lines and splits again, in order to get all the possibilities of k-mers.

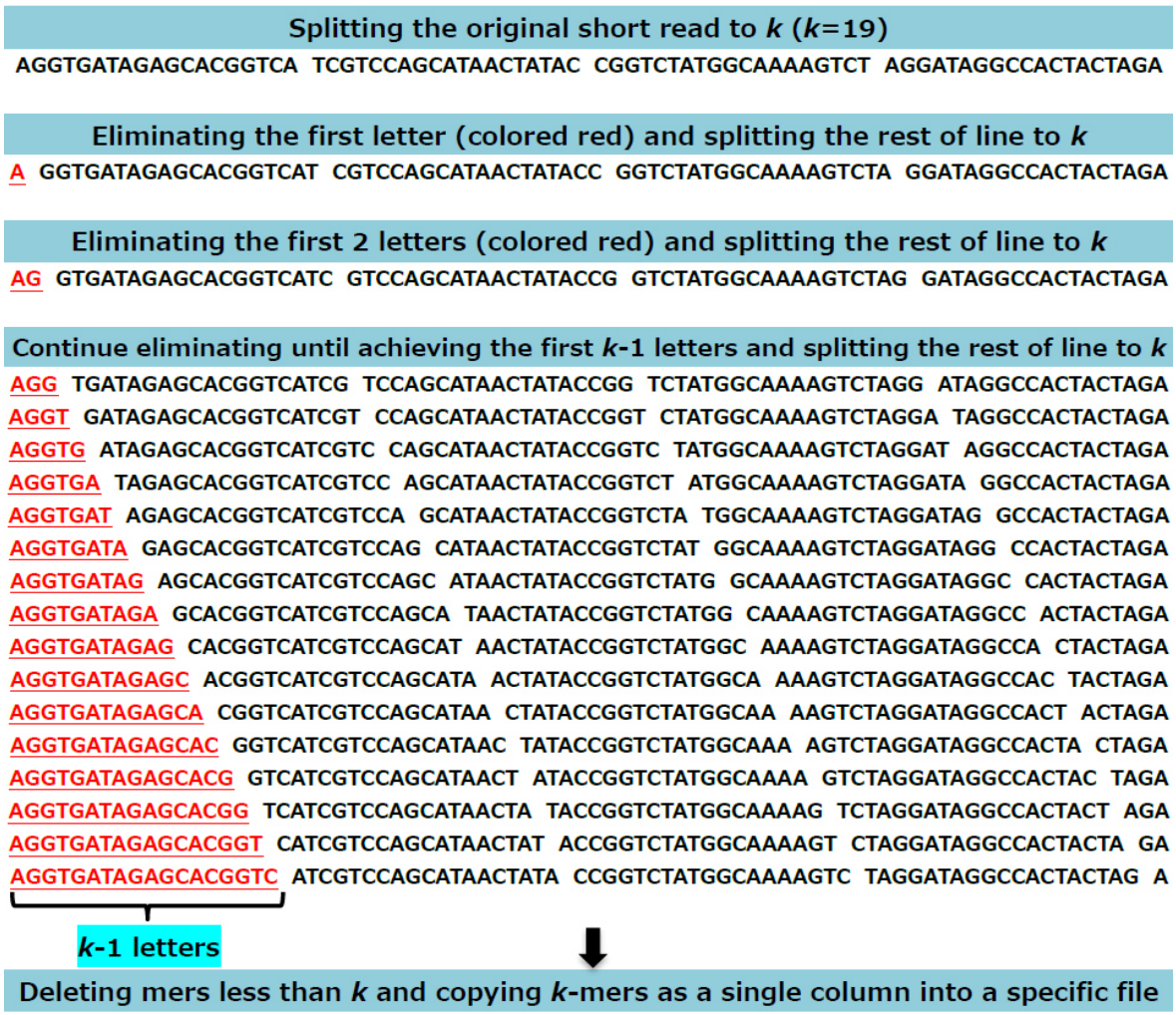

Figure 5.2. An example of RkmerG procedure to get all the possibilities of k-mers (k=19)

Source: ***Karimi R. et al :*** *IEEE Engineering in Medicine and Biology Society, 2015.*

RkmerG generates two files from the original read's file. One is the reference with a line number that considers as the identification number and another is a temporary file for facilitating the process. According to the line number of each read in the reference, RkmerG appends a number to the beginning of each k-mer in a separate column, as the identification

number. This number is equal for all the k-mers of every individual read. The output of this software is a file with 2 columns of id-numbers and k-mers. This file will be loaded into a table that is created with Hive, in the second stage of the pipeline. Figure 5.3 described the algorithm of RkmerG.

```
for r in file
        do
                remove information line of each r
                make each r as a single line
                copy file to reference-file
                add line number to reference-file
                copy file to tmp-file
        done
while read line {1,...,n} in tmp-file
        do
                for i in {1,...,k}
                        do
                                copy line for k times
                                delete i-1 first character of lines
                                split lines to k-mers
                                delete mers<k
                                add reference number to k-mers
                                move k-mers to output
                if end of the tmp-file
                        then
                                exit 0
                fi
        done
done
```

Figure 5.3. RkmerG algorithm. (r=Short read, n=Total number of reads, k=Length of mers, i=each of the lines (reads) that is copied from a single short read for k times, illustrated in figure 5.2.) Source: ***Karimi R. et al :*** *IEEE Engineering in Medicine and Biology Society, 2015.*

Same with the previous method, Metasim was used as the read simulator to simulate the short reads of Roche 454 sequencing technology. DNA signatures of this research were also obtained from the Insignia database. We downloaded DNA signatures of k=19 for 50 bacterial species, chosen randomly from 200 samples. We downloaded also DNA signatures of another 50 bacterial species that were not common to 200 sample bacterial genomes, for

testing the accuracy of the pipeline. In order to prepare the DNA signatures for using in Hive table, we should remove all information from the file that contains signatures of a single species and keep the signatures of a specific length in a column, then append a number as the reference in a separate column, for identifying the owner of these signatures. The reference number can be the alphabetic order of species in the database. All signatures of a species get the same reference number. After appending the reference numbers to the signatures, the result will be a file with 2 columns of reference numbers and signatures for each species. We concatenated all of these files into a single file which will be loaded into the Hive table in the second stage of the pipeline.

Chapter 6

Results and Discussion

This chapter elaborates the results and discussion of the 3 aforementioned methods explained in Chapters 4 and 5.

6.1 The Results of HTSFinder Pipeline

6.1.1 Results for the Bacterial Genome Database

GkmerG has generated 2,773 files with a total size of 177.35 GB containing all the possibilities of 18-mers from individual bacterial genomes in the first phase of the pipeline. After copying the results of the first phase into the HDFS, we ran the WordCount program using Hadoop in order to determine the frequencies of the 18-mers in the 2,773 files. The result of this execution was a 103.03 GB file with two columns. The first column contained the 18-mers or signatures, while the second one contains the frequency number of each 18-mer. Frequency 1 in this file represents the uniqueness of the related signature in the entire database. In other words, an 18-mer with frequency 1 is a unique signature among 2,773 bacterial genomes and an 18-mer with frequency 2 is a common signature which is presented in two genomes among 2,773 bacterial genomes. Table 6.1 represents the quantity of 10 least common and 10 most common 18-mers with their frequencies in the bacterial genome databases. This table shows that 3,552,866,254 of signatures are unique in the database and there is one subsequence (18-mer) that is repeated in 2,125 bacterial genomes.

Table 6.1. A total number of 10 least and 10 most common signatures in the bacterial genome database.

Frequency (Least common)	Number of signatures in the database	Frequency (Most common)	Number of 18-mers in the database
1	3,552,866,254	2040	1
2	689,790,798	2042	1
3	245,109,794	2044	1
4	114,234,398	2074	2
5	68,395,645	2075	1
6	48,107,467	2102	1
7	31,544,271	2112	1
8	26,164,511	2113	2
9	23,650,821	2114	2
10	16,156,541	2125	1

In the third phase, we have created tables in Hive and loaded files from the first and second phases. The table with the reference numbers and k-mers (220.35 GB) and the table with the list of unique signatures (67.5 GB) are used to run the query in Hive in order to specify the species and strains as the owners of the unique signatures. We have repeated the query on the table containing the list of signatures with frequency 2 instead of the unique signature's table to find every pair of species with a common signature. For other frequencies, the same implementation is required.

As shown in Table 6.2, the output of the third phase was a file with two columns containing the following: the signature and the reference number that indicates its corresponding bacterial genome in the reference database created by GkmerG in the first phase.

Table 6.2. An example of the output for the third phase (2 columns in the left side of the table). The reference numbers in this table indicate the numbers appended by GkmerG for easier tracking of data in the pipeline.

Signature	Genome reference number	Name of the bacterial genome that owned the signature
AAAAACGCTCTGATATGA	1059	*Eubacterium_rectale_ATCC_33656_uid59169*
AAAAACGCTCTGCCACCA	1520	*Methanobacterium_SWAN_1_uid67359*
AAAAACGCTCTGGGAATT	705	*Chromohalobacter_salexigens_DSM_3043_uid62921*
AAAAACGCTCTTTTATTT	472	*Campylobacter_hominis_ATCC_BAA_381_uid58981*
AAAAACGCTGAAACGCCT	2649	*Tolumonas_auensis_DSM_9187_uid59395*
AAAAACGCTGAAATCCGC	2013	*Rahnella_Y9602_uid62715*
AAAAACGCTGAATGAAGC	39	*Acinetobacter_ADP1_uid61597*
AAAAACGCTGACAATAAA	1337	*Lactobacillus_brevis_KB290_uid195560*
AAAAACGCTGACCTTCTA	1	*Acaryochloris_marina_MBIC11017_uid58167*
AAAAACGCTGACGGAAGT	2126	*Ruminococcus_albus_7_uid51721*

The following examples are parts of the results obtained by HTSFinder to show the efficiency of this pipeline. No unique DNA signatures with k = 18 were found for 30 of the bacterial genomes in the database.

The number of unique DNA signatures in 475 genomes was less than 10,000. *Chlamydia* as a genus of bacteria with 83 species and strains in the bacterial genome database has the lowest

number of unique DNA signatures of 18-mers. The number of the unique signatures of 18-mers in 75 of them was less than 10,000 and in 57 it was less than 1,000. We have located 13 *Chlamydia* bacteria without unique signatures with k = 18. Top 10 bacterial genomes with the highest number of unique DNA signatures in the bacterial genome database are shown in Figure 6.1.

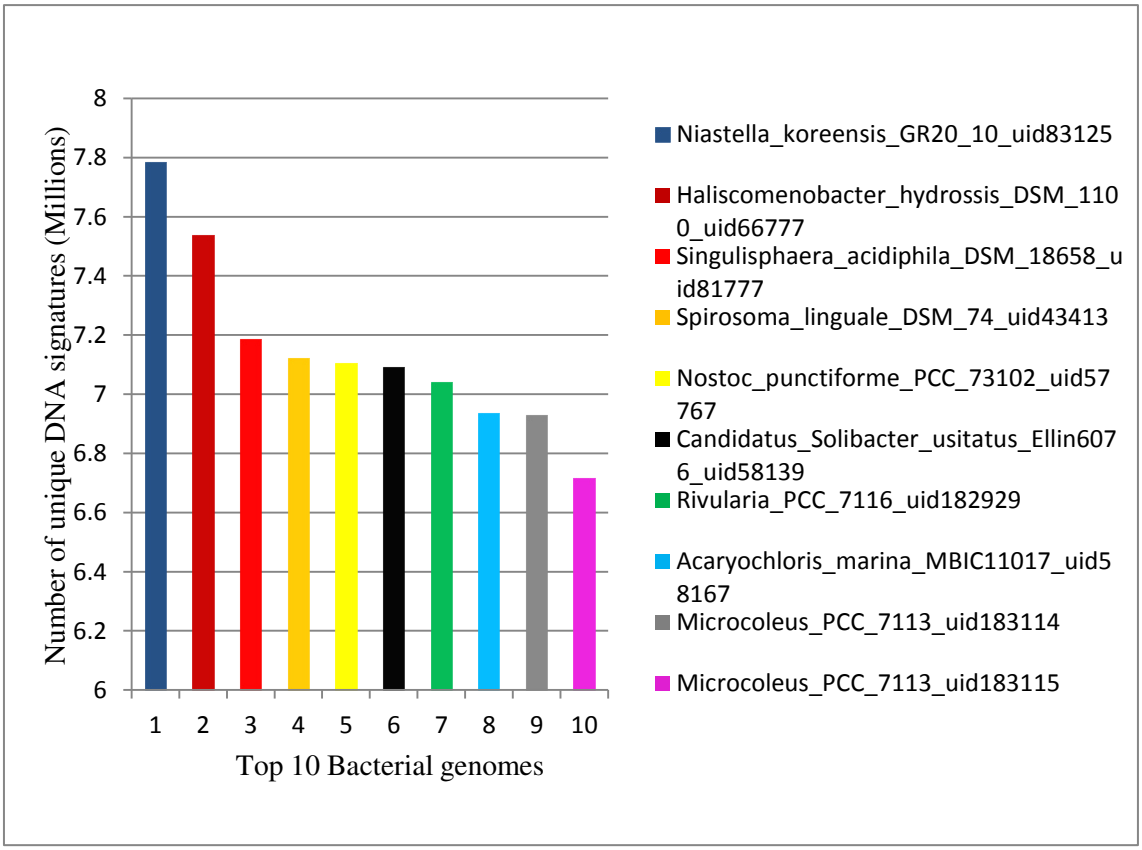

Figure 6.1. Top 10 bacterial genomes with the highest number of unique DNA signatures in the bacterial genome database.

Source: **Karimi R. et al:** Journal of Evolutionary bioinformatics 2016.

Burkholderia mallei and *Burkholderia pseudomallei* are two closely related pathogens that are very difficult cases for PCR assays. These two bacteria are the causative agents of glanders and melioidosis diseases in human and animals [134][164][165]. Due to the phenotypic and genotypic similarity of them, until a few years ago, they were considered to have the same species status. Concerning the literature, only one PCR signature was reported to be unique to *B. mallei* [134][164]. The HTSFinder pipeline could detect a considerable

number of DNA signatures for *B. mallei* and *B. pseudomallei*. Although these signatures are just unique in the bacterial genome database, due to the notable number of signatures listed in Table 6.3, it is evident that under different circumstances it would be better to have an alternative opportunity to define the uniqueness of the DNA signatures and to select the target databases according to the requirements. Moreover, it should be noted that much more DNA signatures could be found by increasing the length of k-mers.

Table 6.3. *B. mallei* and *B. pseudomallei* strains with their number of unique DNA signatures of 18-mers in the bacterial genome database.

The reference number and name of the *Burkholderia* genomes	**Number of unique DNA signatures**
Burkholderia_mallei_ATCC_23344_uid57725	90,278
Burkholderia_mallei_NCTC_10229_uid58383	24,858
Burkholderia_mallei_NCTC_10247_uid58385	19,442
Burkholderia_mallei_SAVP1_uid58387	7,649
Burkholderia_pseudomallei_1026b_uid162511	282,992
Burkholderia_pseudomallei_1106a_uid58515	173,688
Burkholderia_pseudomallei_1710b_uid58391	41,153
Burkholderia_pseudomallei_668_uid58389	218,985
Burkholderia_pseudomallei_BPC006_uid174460	81,768
Burkholderia_pseudomallei_K96243_uid57733	195,711
Burkholderia_pseudomallei_MSHR305_uid213227	320,198
Burkholderia_pseudomallei_MSHR346_uid55259	172,551
Burkholderia_pseudomallei_NCTC_13179_uid226109	382,494

This pipeline detects the common signatures not only among a species and its strains but also in the entire database. Frequencies 2 and 3 have been considered as the samples to prove the efficiency of this pipeline for discovering common DNA signatures within the bacterial genome database.

As an example, the following results were obtained for *Acaryochloris_marina_MBIC11017_uid58167* that is the first bacteria in the database. A total of 689,790,798 signatures of k = 18 with frequency 2 were found in the database, whereas

673,490 of them are shared between *Acaryochloris_marina_MBIC11017_uid58167* and 2,382 other species.

There was not any signature of k = 18 with frequency 2 between *Acaryochloris_marina_MBIC11017_uid58167* and 390 other bacterial genomes. Figure 6.2 presents the highest number of signatures with frequency 2 which are common between *Acaryochloris_marina_MBIC11017_uid58167* and 10 other bacterial genomes in the database.

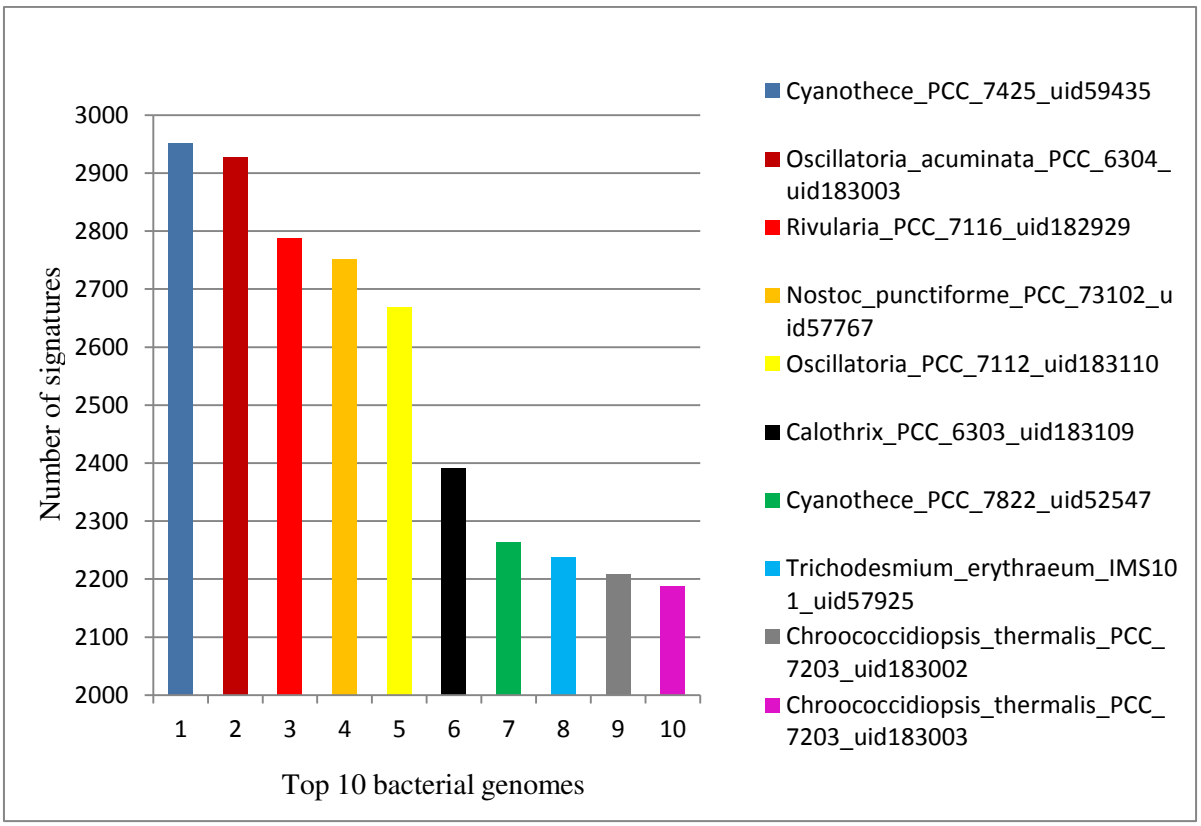

Figure 6.2. Ten bacterial genomes with the highest number of signatures common with *Acaryochloris_marina_MBIC11017_uid58167* in the bacterial genomes database. This is an example of the results for common signatures with frequency 2 obtained by HTSFTSFTSFinder.

Source: **Karimi R. et al:** Journal of Evolutionary bioinformatics 2016.

The results of the executions for signatures with frequencies 2 and 3 showed that most of the signatures are shared among phylogenetically close species of the database. However, there

were also a lot of signatures that belonged to unrelated bacterial genera and families. Table 6.4 shows a partial view of the results for frequencies 2 and 3. From 245,109,794 signatures of frequency 3 in the database, 160,264 of them are shared between *Acaryochloris_marina_MBIC11017_uid58167* and two other species.

Table 6.4. A portion of the results for signatures with frequencies 2 and 3 in the database. Concerning the reference numbers, most of the common signatures are shared among the phylogenetically close genomes. However, the number of common signatures among unrelated species is also notable.

Signatures with frequency=2	Genome reference numbers		Signatures with frequency=3	Genome reference numbers		
AAAAAAAAAAGATAAATA	355	508	AAAAAAAAAAAAATATCG	1709	1708	2677
AAAAAAAAACAGACACAA	2110	2109	AAAAAAAAAAAACAGAAC	1249	1255	1267
AAAAAAAAACAGCATTAA	2209	2214	AAAAAAAAAATAAATACA	2726	2734	542
AAAAAAAAACAGGCTTAC	394	1499	AAAAAAAAAGAAACAAAG	681	678	679
AAAAAAAAACCGCCGAAC	1046	1048	AAAAAAAAAGATGTTAAT	969	2384	247
AAAAAAAAACCGCTTTTA	1879	1265	AAAAAAAAAGCAAAACAA	2223	355	102
AAAAAAAAACGAACAAAC	101	1813	AAAAAAAAAGTAAATGCG	1793	2731	2730
AAAAAAAAACGATTCAGA	2106	2107	AAAAAAAAATAGACAATG	498	500	755
AAAAAAAAACTAATGCTT	349	355	AAAAAAAAATATTCATGC	321	897	560
AAAAAAAAACTAATTCTG	1406	1408	AAAAAAAAATTCAAAATT	567	505	325
AAAAAAAAAGAACCAAAC	544	545	AAAAAAAAATTTAGCGAT	2714	1814	321
AAAAAAAAAGACTGACTC	2696	1066	AAAAAAAAATTTTTATAG	703	402	324
AAAAAAAAAGATGTTGTA	545	544	AAAAAAAACAAGAAGCGC	1426	1427	1428
AAAAAAAAAGGATTCGAA	1428	1427	AAAAAAAACAATTAGCGA	1128	2677	2404
AAAAAAAAATAAAGACTC	345	343	AAAAAAAACAGATAGTGA	2115	1061	1508
AAAAAAAAATAGTGACGA	1686	1693	AAAAAAAACAGCAGCACC	2535	1584	1058

A series of lengths k from 21 to 30 have been considered to evaluate the effect of increasing the length of the signature in the results and to compare the results of the odd and the even number of k. Table 6.5 contains the number of unique DNA signatures for the human genome and three chromosomes **1**, **x**, and **y**, which represent large, medium, and small sequences in the human genome. This table shows that increasing the length of the signature causes increasing the number of unique signatures and there is not a meaningful difference between the odd and the even numbers.

Table 6.5. A number of unique DNA signatures in the human genome and its three chromosomes with different sequence sizes for a series of lengths of k-mers from 21 to 30.

Length of signature	The whole genome (2.9 GB)	Chr1 (222 MB)	Chrx (147 MB)	Chry (19 MB)
$k = 21$	2.24297e+09	176,137,004	109,691,126	10,221,240
$k = 22$	2.28624e+09	179,436,876	112,370,062	10,550,076
$k = 23$	2.31954e+09	181,982,115	114,505,744	10,825,761
$k = 24$	2.34792e+09	184,157,371	116,349,017	11,070,875
$k = 25$	2.37333e+09	186,108,431	117,999,580	11,294,439
$k = 26$	2.39664e+09	187,904,867	119,504,725	11,501,139
$k = 27$	2.41829e+09	189,580,382	120,891,039	11,693,180
$k = 28$	2.43849e+09	191,150,531	122,172,724	11,872,250
$k = 29$	2.45744e+09	192,629,345	123,363,397	12,039,734
$k = 30$	2.47529e+09	194,027,911	124,472,828	12,196,710

To compare the number of unique signatures against the within-species variability and the entire bacterial genome database, *Bacillus* species with 81 strains in the database was selected. The three phases of the pipeline were executed on these strains. Table 6.6 contains five strains of *Bacillus* with the highest number of unique signatures and five others with the lowest number within species and in the entire database. Although *Bacillus_anthracis_A0248_uid59385* and *Bacillus_anthracis_Ames_Ancestor_uid58083* have larger genome size, any unique signature of length 18 could not be found for them because of their high similarity with other *Bacillus* strains.

Table 6.6. A comparison of five *Bacillus* strains with the highest number of unique signatures and five others with the lowest number of signatures of length 18 within species and in the entire database. This table shows that within-species similarity and variability have more influence on the volume of signatures than the remainder of the database.

Name of strains	Within-species	In the entire database	The original genome size
Bacillus_megaterium_WSH_002_uid159841	4,860,315	4,012,591	5,0 MB
Bacillus_infantis_NRRL_B_14911_uid222804	4,712,042	3,932,760	4,8 MB
Bacillus_1NLA3E_uid81841	4,527,694	3,734,930	4,7 MB
Bacillus_cellulosilyticus_DSM_2522_uid43329	4,441,938	3,688,824	4,6 MB
Bacillus_clausii_KSM_K16_uid58237	4,177,156	3,576,848	4,2 MB
Bacillus_subtilis_168_uid57675	248	205	4,1 MB
Bacillus_amyloliquefaciens_CC178_uid226115	247	202	3,8 MB
Bacillus_anthracis_A0248_uid59385	0	0	5,4 MB
Bacillus_anthracis_A2012_uid54101	0	0	284 KB
Bacillus_anthracis__Ames_Ancestor__uid58083	0	0	5,4 MB

6.1.2 Results on Both Forward and Reverse-Complement Sequences of the Bacterial Genome Database.

In the genome databases such as NCBI, only one strand of DNA sequence is provided. However, to design the primers, both forward and reverse-complement sequences should be considered. Moreover, depending on the sequencing technology, generated short reads can be from both strands. Therefore, the ability to obtain DNA signatures of both strands is potentially useful.

For the reverse complement sequences, the size of the output files and the computational times of the first and second phases of the pipeline were the same as in the forward genome implementations. The output of the second phase for forward and reverse-complement genome databases resulted in a file of 103.03 GB for each. We have repeated WordCount on both of the databases one more time to determine the frequencies of k-mers as illustrated in Figure 4.3 (Chapter 4). The final result was a file of 52.53 GB for the forward and the same size for the reverse-complement genome database. On the one hand, it means that the volume

of data containing unique signatures for the forward database decreased from 67.5 to 52.53 GB similarly to the reverse-complement genome database. On the other hand, the overall volume of DNA signatures that we could find for the species in the bacterial genome database increased from 67.5 GB containing signatures for a single strand to 105.06 GB for both strands of DNA.

6.1.3 Results for the Forward and Reverse-Complement Bacterial Genome Database and the Human Genome Database.

We have considered the forward bacterial genome as the target database. We have applied the method that is described in Figure 4.3 (Chapter 4) and found 50.28 GB of k-mers for the target genome database which is unique among the three databases.

Table 6.7 presents the file size and numbers of unique DNA signatures of the target database against the non-target ones.

Table 6.7. Number of unique DNA signatures for the forward bacterial genome database as the target and two other non-target databases.

Databases	File size	Number of signatures
Unique signatures of the Forward bacterial genome database	67.5 GB	3,552,866,254
Forward + Reverse-Complement bacterial genome databases	52.53 GB	2,764,759,739
Forward + Reverse-Complement bacterial genome + Human genome databases	50.28 GB	2,646,494,945

6.1.4 Performance Evaluation and Computational Times.

For this experiment, we have applied two different platforms. The first one was a single node with 12 processors of Intel Core i7-4930K CPU at 3.40 GHz and 55 GB of RAM and 6 TB of hard disk. The operating system was Ubuntu 12.04.5 LTS, Java SE Version “1.8.0–25”,

Hadoop Version 1.2.1, and Hive-0.12.0. We have installed this node as a single-node Hadoop cluster.

Another platform was a multi-node cluster with seven nodes including the master node and six slave nodes. The master node was an Intel Core2 Quad CPU Q6600 at 2.40 GHz and 8 GB of RAM and 3.2 TB of hard disk, while slaves had 4 GB of RAM, Intel Core i3-2100 CPU at 3.10 GHz and 500 GB of hard disk, all with the desktop version of Ubuntu 14.04.1 LTS 64-bit, Java Version "1.7.0–65" OpenJDK, Hadoop 12.1, and Hive-0.12.0.

The first phase of the pipeline executed with GkmerG took 156 minutes with five nodes and 780 minutes with a single node from the second platform to generate 18-mers from the original bacterial genome database (9.7 GB). As an output, we got 2,773 files containing 18-mers with a total size of 177.35 GB.

Table 6.8 contains the corresponding computational results and the size of the files in the second and third phases of the pipeline for both platforms.

Table 6.8. A comparison of computational results of the first and second platforms in the second and third phases of the pipeline in order to find unique DNA signatures and their related species in the forward genome database (time in minutes).

Phases	File size (GB)	Time for the first platform Single-node	Time for the second platform Multi-node
Copy k-mers generated by GkmerG to the HDFS	177.35	60	63
WordCount process	177.35	447	1169
Copy the result from HDFS to a local directory	103.03	34	27
Extracting unique signatures and creating tables in Hive	67.5	60	60
Loading unique signatures to the Hive table	67.5	23	26
Loading *k*-mers and reference numbers to the Hive table	220.35	79	83
Executing the queries and copy the result to a local directory	83.83	1120	959
Total computational time		1823	2387

Although the whole computation on the second platform took about nine hours more than on the first one, comparing the RAM and CPU capacity of the two platforms confirms the ability of a cluster of low-cost computers that are commonly available with research facilities to accelerate big data analytics.

Since the WordCount program uses a sorting algorithm, the platform with a larger amount of RAM is faster; therefore, the single-node platform is faster for the second phase of the pipeline. For the third phase of the pipeline, the running time of querying in the multi-node cluster is faster, because the process is linear; therefore, parallel computing in a distributed system can speed up the process.

Table 6.9 compares the size of the files for frequencies 1–3 and the time of loading and processing the queries on the first platform. The file containing 220.35 GB data was used as the second table for all the implementations.

Table 6.9. A comparison of loading and execution times of the frequencies 1–3 in the third phase.

Frequency	**1**	**2**	**3**
Size of the file containing signatures (GB)	67.5	13.1	4.66
Time for loading file into the Hive table (minutes)	23	4	1
Execution time and copy the result to local directory (minutes)	1120	661	557

6.2 The Results of the Method Based on Using Bitmap Indexes and NoSQL

The output file of the method contains the Rowid (Row identification) numbers of the short reads (first column as the read id-number) and its related bacteria. As an example, for the bacteria b1 (first column of the indexes) in Table 6.10, we have 2,7,10 which means, the signatures of bacteria b1 are in these 3 short reads. Moreover, it means that we have bacteria b1 in the metagenomics sample.

Table 6.10. An example of input table for Hive, that is created by merging files to a single file.

		b1	b2	b3	b4	b5
1	R1	0	0	0	1	0
2	R2	1	0	0	0	0
3	R3	0	0	0	1	0
4	R4	0	0	0	0	0
5	R5	0	0	1	0	0
6	R6	0	0	0	0	1
7	R7	1	0	0	0	0
8	R8	0	0	0	0	0
9	R9	0	0	0	0	0
10	R10	1	0	0	0	0

In order to prevent repeating the queries or writing long commands and queries, we can add all bacterial files with '0' and '1' one after the other and create a single column in a single file. It can be done with scripting in the incremental order to add as much bacteria as we need at the end of each other. In this method, we need also to repeat short reads in a single column as much as the number of bacteria. For instance, if we have 500,000 short reads and 1000 bacteria, then we should repeat short reads in one column and for1000 times.

In this case, we need to create a table with 3 columns Rowid, read string, and “0” or “1” as the indexing results; then run the query just once. The results will be in one column. We can easily extract the information with Rowid numbers. It leads to a larger table (6,635,250 rows) and files size, but a faster implementation. After getting the results, we can delete these large tables. More details can be found in the related paper [41].

Running time of querying on the first tables (52 columns, 132,706 rows, and the loaded file size of 44.6 MB) was 21.31 seconds in average for each bacterium. 1065.927 Seconds for 50 bacteria. It would be much higher if we consider the time for changing and repeating the queries. Running time of querying on the second tables (3 columns and 6,635,250 rows and the loaded file size of 1.6 GB) was 59.9 seconds for the whole process. Although the second table was much larger in file size and 50 times more number of rows, the computational time of querying in compare with the first table is very short.

This implementation shows the speed and efficiency of Hive to search millions or billions of rows with Bitmap Index techniques.

Since the nature of Hive is to write once and to read several times, updating the tables is not possible. There is a possibility of integrating Hive and Hbase. This feature allows Hive QL statements to access Hbase tables for both reads (SELECT) and write (INSERT). It is even possible to combine access to Hbase tables with native Hive tables via joins and unions. Real-time reading and writing are possible in Hbase. These features help us updating and having a faster implementation. Query optimization can also help to have shorter running time. The next two queries on the same table are evident.

File Size: 1.6 GB
*Time taken with **SELECT*** and **GROUP BY** query:*
Total MapReduce CPU Time Spent: 54 seconds 640 msec
*Time taken: **59.901 seconds***

*Time taken with **SELECT** query which is faster:*
Total MapReduce CPU Time Spent: 43 seconds 630 msec
*Time taken: **44.452 seconds***

The implementation above was for the first group of 50 bacteria which are common in 100 bacterial samples. As we expected, we could find some short reads containing the signatures for every bacteria. The number of short reads is a range between 1 for b16 to 812 for b4. As we expected, for the second group of 50 bacteria which differs by 100 samples, we could not find any short reads containing the signatures. The average time taken for the implementation was almost the same as the first group.

This implementation shows that Bitmap Index techniques are very efficient to speed up the Hive queries, and Hive itself is powerful enough to search for very large tables with millions or billions of rows or columns in commodity hardware. Moreover, with this method, we could show that it is possible to identify species with DNA signatures from metagenomics samples without assembling and alignment and with any size of data.

6.3 The Results of the SRIdent Pipeline

The first 200 species from the bacterial genome database were selected to simulate their short reads by Metasim. This application simulated 2,030,000 reads with Roche 454 sequencing model and default simulation parameter settings. The size of the original reads file was 896.59 MB. RkmerG was used to generate all the possibilities of 19-mers for each short read, and to append the identification number (1 to 2,030,000) to the 19-mers. The result was an 11.91 GB file containing 2 columns of identification numbers and k-mers. RkmerG finished this process within 144 minutes with a single computer of Intel Core i3-2100 CPU at 3.10GHz, 4 GB of RAM, and 500 GB of hard disk. We created two tables in the hive. The file containing 19- mers and their identification numbers were loaded into one of them and the file containing the DNA signatures and their reference numbers into another. Then we ran SELECT and JOIN queries on both tables in order to select the k-mers of the reads, matching to the signatures. The result was a file containing 3 columns: the signature, the reference number of the signature and the identification number of k-mers that indicates their related short reads. Table 6.11 shows a small part of the Hive output file.

Table 6.11. A sample of the results obtained from Hive. The genome reference number indicates the species that owns the signature and the identification number indicates the short read that contains the k-mer.

Signature	Genome reference number	Read's identification number
AAAACAAGCAATCGCCCGA	36	462891
AAAACAAGCGAAGGTGCGG	1	373679
AAAACAAGCGAAGGTGCGG	1	374030
AAAACAGCTTTGCTTGACG	50	1468571
AAAACAGCTTTGCTTGACG	50	1982597
AAAACCAACACCGGCTTAT	13	339456
AAAACCAACGTCTATGCGC	41	210636
AAAACCAGCCGGCCGTGGC	49	1756507
AAAACCATCGATAAACTCG	27	1594247
AAAACCATGGAAGGTGTAG	20	166134

The first row of Table 6.11 says that the signature in this row belongs to the bacteria number 36 and exists in the read number 462891. In the other word, it means bacteria number 36 is one of the species in the sample. This is what we were looking for, as the aim of this research to identify the species in a metagenomics sample, directly from the raw sequencing reads without assembling and alignment. There are too many repeated copies of the short reads produced by sequencing technologies. This is the reason that we see a considerable number of repeated signatures and reference numbers in the Hive output. For instance, in rows 4 and 5 of Table 1, we can see the same signature of bacteria 50 in two short reads. On the one hand, it means these two short reads are overlapped in one part, on the other hand, it can be an error caused by sequencing technology or it can be a read belongs to unknown species that should be checked. This pipeline and presented software can be used to facilitate assembling, alignment, and mapping of the reads with searching the position of DNA signatures in the reference genome and checking whether the read containing the signature can be matched with the reference genome. Extracting the reads with same signatures and reference numbers from Hive output and checking whether they are overlapped is an easy and significant way for assembling and alignment of the reads from known species. Neither of the signatures of the species with reference numbers between 51 to 100 was found in the short reads. These signatures have been selected from 50 species apart from those 200 species as the sample in order to check the accuracy of the pipeline.

6.3.1 Performance Evaluation and Computational Times

In this experiment, we applied the second phase of the pipeline with two different platforms. The first one was a multi-node cluster with 5 nodes including the Master node and 4 Slave nodes. All with 4 GB of RAM, Intel Core i3- 2100 CPU at 3.10GHz and 500 GB of hard disk, the desktop version of Ubuntu 14.04.1 LTS 64-bit, Java version ”1.7.0- 65” OpenJDK, Hadoop 12.1 and Hive-0.12.0. The second platform was a single node with the same hardware and software setup. Table 6.12 shows computational time differences of two mentioned platforms.

Table 6.12. Computational times of two platforms (Time is measured in seconds).

Platform	Loading k-mers (11.91 GB)	Loading signatures (80.14 MB)	Query running time
Single-node	206	1	2909
Multi-node	231	1	585

Since multi-node needs more communication between the nodes through a network, the loading time is longer. Running time of querying in the multi-node cluster is much faster, because the process is linear; therefore, parallel computing in a distributed system can speed up the process.

The important goal of this research is providing a method for rapid identification of species directly from the raw short reads generated in Next-Generation Sequencing (NGS) and Whole-Genome Sequencing (WGS) technologies without considering the complicated processes such as assembling and alignment. The obtained results showed the ability of this method to reach the goals. Moreover, this pipeline and presented software can be used to facilitate assembling, alignment, and mapping of the reads according to the Hive output.

Another goal of this research is using commodity hardware for the analysis in order to enhance the applicability of the analysis as a routine process for the entire research community and particularly medical laboratories. The computational resources that are used in the proposed pipelines prove the ability of parallel and distributed computing to reach the goal.

It is notable that, the integration of the HTSFinder and SRIdent pipeline can speed up the process in the second phase of the SRIdent pipeline. DNA signature preparation (explained in Chapter 5) is not required and we can use the output of the third phase of HTSFinder in the second phase of SRIdent [166].

6.4 Performance Comparison of Proposed Pipelines with an Existing State-of-the-Art Method

In order to choose a method to compare the performance of the HTSFinder pipeline with the existing methods mentioned in Chapter 4, we have extended Table 4.1 with adding the resources of our pipeline in Table 6.13.

Table 6.13. The extension of Table 4.1 for better comparison of the computational resources of HTSFinder with the existing methods.

Name	Data format	Adopted platform according to the publication	BLAST specificity	Ability for a single sequence	Ability for multiple sequences
TOFI	FASTA	64 x 1.5 GHz Itanium 2 processors with 64 GB of shared memory	✓	✓	×
TOPSI	FASTA	98-cores Linux cluster	✓	✓	✓
Insignia	FASTA	192-node Linux cluster	✓	✓	✓
CaSSiS	rRNA	Intel Core i7 CPU (4 cores, 2.67 GHz) with 24 GB of RAM	×	✓	✓
CMD and PISD	ESTs	Dell PowerEdge R900 server with two Intel Xeon E7430 2.13 GHz quad-core CPUs, 12 GB RAM	×	✓	×
IMUS	ESTs	Intel 2.93GHz CPU	×	✓	×
PIMUS	ESTs	Intel Core i7 870 2.93GHz quad-core CPU and 16 GB RAM	×	✓	×
DDCSD	ESTs	A Master Node: Intel Core i7 CPU 870 at 2.93 GHz and 16 GB RAM, 10 Slave nodes: Intel Core i7 CPU 3770 K at 3.50 GHz and 32 GB of RAM for each one	×	✓	✓
HTSFinder	FASTA	The master node: Intel Core2 Quad CPU Q6600 at 2.40 GHz and 8 GB of RAM and 6 slaves Intel Core i3-2100 CPU at 3.10 GHz with 4 GB of RAM for each one	×	✓	✓

Our pipeline is compatible with FASTA-formatted files. Although TOFI, TOPSI, and Insignia use FASTA format, they use BLAST specificity for checking the uniqueness of their signatures against the whole background genomes which is not the aim of this research. Moreover, they use a large cluster of computers with more than a hundred of nodes which are neither available for us, nor for many of other research communities. CaSSiSS uses rRNA database that is also not compatible with our pipeline. IMUS, PIMUS, CMD, PISD, and DDCSD use ESTs sequences that are not appropriate for our pipeline as well. Moreover, the download source of DDCSD that has provided in the related paper in BMC Bioinformatics

journal is not available. Furthermore, most of the applications and algorithms of DNA signature discovery that are introduced in Chapter 4 are suitable for a single sequence or a single targeted genome in a run. Hadoop and Hive are designed to handle large data analytics. Although they can process a small amount of data, regarding the time and computational resources it is not reasonable.

Same with HTSFinder pipeline, Distributed Divide-and- Conquer-based Signature Discovery (DDCSD) uses a parallel and distributed computation system. The human whole-genome EST database that has approximately 2.46G bases is used in DDCSD for detecting unique signatures in this database. Since this database is derived from the human whole-genome database (2.9G bases) and DDCSD did not use BLAST specificity for checking the uniqueness of the signatures, it seems this is the only possible method for doing a comparison, although the computational resources are not the same. Table 6.14 contains a comparison of hardware setup and discovery time between HTSFinder and DDCSD for finding unique DNA signatures of the whole human genome database and human whole-genome EST database.

Table 6.14. Comparison of HTSFinder and DDCSD (d = mismatch tolerance for the DDCSD algorithm).

Hardware and software information	**HTSFinder**	**DDCSD**	
Number of master nodes	1	1	
Master node information	Intel Core2 Quad CPU Q6600 at 2.40GHz	Intel Core i7 CPU 870 at 2.93 GHz	
Number of slave nodes	6	10	
Slave nodes information	Intel Core i3-2100 CPU at 3.10GHz	Intel Core i7 CPU 3770 K at 3.50 GHz	
RAM capacity of master node	8 GB	16 GB	
RAM capacity of slave nodes	4 GB (24 GB all together)	32 GB (320 GB all together)	
Master node disk space	3.25 TB	1.5 TB	
Slave nodes disk space	500 GB	1 TB	
Operation system	Ubuntu 14.04.1 LTS 64-bit	CentOS release 6.3	
Human genome size	2.9 GB (hs-ref-GRCh38)	2.46 GB (Human genome ESTs)	
Length of signature (k)	24	24	
Discovery time (minutes)	**532**	**196**	d = 2
		873	d = 4

Comparing the hardware setup of the two clusters and their discovery time for detecting unique DNA signatures of the Human genome with k=24, we can state that HTSFinder is a fast and powerful method that uses ordinary hardware to find DNA signatures. Although the number of nodes and the total amount of RAM and CPU for the cluster that is used with the DDCSD algorithm is several times higher than for HTSFinder, the discovery time of HTSFinder is considerably small. Moreover, HTSFinder could find not only all the unique signatures but also other frequencies of k-mers in the whole human genome at the same time. HTSFinder detects all the possibilities of unique DNA signatures and other frequencies without discarding even one signature in the database, regardless of determining mismatch tolerance level. Meanwhile, it should also be noted that the discovery time which is calculated for HTSFinder includes the time for generating k-mers (65 minutes). Once we finish this process, we can keep the k-mers for other implementations.

The read classifiers such as Kaiju, Kraken, and CLARK are used for taxonomic assignment of the short reads. The same with our method using SRIdent pipeline, they use data in FASTA format. These applications have focused on the speed of the running time of the process. They are trying to shortage the process using High-performance computation systems shown in Table 6.15.

Table 6.15. A comparison of read classifier applications with SRIdent, according to the data format, computational resources and ability to process single or multiple sequences.

Name	Data Format	Adopted Platform according to the publication	BLAST specificity	Ability for single sequence	Ability for multiple sequences
Kaiju	FASTA	HP Apollo 6000 System ProLiant XL230a Gen9 Server, with two 64-bit Intel Xeon E5-2683 2 GHz CPUs (14 cores each), 128 GB DDR4 memory	✓	✓	✓
Kraken	FASTA	48 AMD Opteron 6172 2.1 GHz CPUs and 252 GB of RAM, running Red Hat Enterprise Linux 5.	✓	✓	✓
CLARK	FASTA	Dell PowerEdge T710 server, dual Intel Xeon X5660 2.8 GHz, 12 cores, 192 GB of RAM	✓	✓	✓
SRIdent	FASTA	The Master node and 4 Slave nodes. All with 4 GB of RAM, Intel Core i3- 2100 CPU at 3.10GHz	✕	✓	✓

Although SRIdent is also a read classifier, it is developed with the focus on the applicability of the pipeline for ordinary computers which are available everywhere, in order to utilize Big Data analysis as a routine process for the research community and in particular for the medical laboratories. In the publications related to the three mentioned applications in the above table, the source of collecting unique DNA signatures is not exactly determined, however, it is mentioned that they use BLAST specificity. Along with the mentioned differences, unavailability of the resources for us is an extra reason for the inability to do the comparison.

Chapter 7

Conclusion and Future Works

7.1 Summary and Conclusion

One of the key elements of the medicine of the future is to bring the applications of Next-Generation Sequencing (NGS), Whole Genome Sequencing, and Single Cell Sequencing techniques as the routine tools for diagnostic and public health microbiology.

The dramatic reduction in the cost of sequencing has resulted in the applicability of bacterial genome sequencing for a great number of labs; however, the explosive growth of data has resulted in a cost shift from sequencing to assembly, analysis, and managing of the sequencing data. Therefore, the computational analysis is likely to be a few years away.

Due to the massive data volumes and complexity of the analysis, computational resources are expensive and time-consuming. Moreover, the complexity of the applications requires adequate knowledge of computer science. Hence, these problems are the barriers for the applicability of the analysis as a routine process for the entire research community.

The overriding purpose of this research was to overcome the limitations in the computational analysis of rapid diagnostic identification and characterization of species and infectious pathogens from raw reads sequencing data, in particular, complex metagenomics data. This research has focused on 2 main primary goals:

1- Reducing the cost and time-consumption of the computational resources by the use of parallel and distributed computing and related optimization techniques, in order to utilize ordinary desktop computers for Big Data analysis.

2- Proposing a fast, flexible, and independent alignment-free method for real-time identification of microorganisms from High-throughput sequencing reads.

In the earlier chapters of this doctoral thesis, especially in Chapters 1, 2, and 4, we discussed the critical importance of employing DNA signature as an alignment-free method for real-time identification of species and pathogens in metagenomics data analysis and clinical diagnostics assays. Due to the large size of the datasets, the process is computationally intensive.

Chapter 4 of this thesis has introduced a powerful pipeline (HTSFinder) that we have developed for searching DNA signatures (small fragments of a genome that are specific for a species) among arbitrarily targeted genome databases, using Hadoop and MapReduce as the application of parallel and distributed computing in commodity hardware.

HTSFinder consists of three phases:

1- Generating all the possibilities of k-mers of genome sequences in FASTA format with GkmerG software.
2- Extracting all unique signatures or group-specific signatures due to the frequency of k-mers.
3- Determining the owners (species) of the signatures.

Data obtained in this research using HTSFinder, clearly shows the efficiency of our proposed pipeline to find all possible DNA signatures of a target database. In this pipeline, we intended to overcome some limitations of the existing state-of-the-art methods for DNA signature discovery by focusing on efficiency issues to detect all the possibilities of unique and common DNA signatures in a database, regardless of the challenges such as pairwise alignment and mismatch tolerance. Another important feature of this pipeline is its ability to select a target and non-target databases. From the standpoint of this research, non-target genome database is not necessarily defined as the entire background genome databases such as BLAST for the assessment and specificity evaluation of DNA signatures. It can be determined due to the requirements. General applicability is another issue that is considered in the pipeline; it can be launched either in a cluster of low-cost nodes or in an HPC environment. Although the volumes of the datasets in this study are very large (e.g., 287.85 GB in a single run), DNA signatures are detected very precisely and comprehensively in the target databases and the execution times are reasonably short. The proposed experiment is just the basic idea, and there is a great flexibility to design implementations for phases of this approach. Once the pipeline is implemented, the users will find how to manipulate their datasets according to the requirements.

After finding the DNA signatures of each species, the next challenge is how to match the signatures, the short reads (raw sequence data), and their related species.

In Chapter 5 we have proposed two methods for matching the signatures, the reads, and their related species.

In the first method, we employed optimization techniques issued from databases to speed up searching and matching of DNA signatures in the short reads of hundreds (thousands) of different microorganisms deployed in Hive. We have adapted the concept of bitmap indexes, routinely used for indexing large database tables on the attributes with little cardinality.

The results of this method show the efficiency of optimization and bitmap indexing techniques to speed up querying of large tables in Hive. However, constructing the index tables is a time-consuming process. Therefore, we attempted to develop a better solution.

In the second method, we introduced SRIdent pipeline that consists of two computational stages:

1- Generating the k-mers from the reads, where k is equal to the length of signatures. This stage is done by the software called “RkmerG”.
2- Creating the related tables and using Hive queries to find the matches is the next stage.

The results obtained from SRIdent pipeline prove the efficiency of this approach for real-time identification of species in Metagenomics and clinical diagnostic assays.

The three presented methods in this doctoral thesis can be an efficient approach, not only for DNA signature discovery and real-time identification of species but also for other purposes in bioinformatics and metagenomic studies such as the alignment and assembly of short reads and next-generation sequencing analysis. They can facilitate assembling, alignment, and mapping of the reads according to the Hive output. Another advantage is the applicability of the methods on low-cost computers.

7.2 Future Work

An important challenging issue discussed above is the high cost of computational resources and the complexity of the analysis and the applications that require adequate knowledge of computer science. These two parameters prevent the use of DNA signature and other classifier methods as a routine technique for High-throughput sequencing analysis. Thus, we were motivated to consider Hadoop as a model for developing integrated software called HTSFinder-alfa.

The software is very easy to perform for everybody. We employed the divide and concur algorithm and parallel programming along with artificial intelligence and machine learning techniques to make an automated and powerful application for the whole process that we discussed in this doctoral thesis.

According to the size of data and the amount of RAM and the number of CPU threads and cores, the software will decide how to break down the data into the smaller pieces and how to process the data in parallel and in an intelligent manner. This software is still under the development.

In the final version of this software we can easily copy the target databases of DNA signatures in a specific folder and copy the short reads to another folder and run the software with a short command; the software search the DNA signatures (unique or common with arbitrary length), then it matches the signatures, the reads, and their related species automatically.

We are trying to develop this software for distributed systems. The software can perform the process on a desktop computer with 4GB RAM in Linux system.

There are several optimization issues that we recommend as the future work in the proposed pipelines, in order to enhance their capabilities.

Running time, CPU usage, memory usage, speed of disk reading of the nodes, and managing the maps according to the size of data and memory are the subjects of optimization in a

Hadoop cluster. Optimizing the configurations and parameters of Hadoop is also required in order to reduce the data transfer and communication between nodes of the cluster. Moreover, query optimization and designing the tables in Hive for preventing a repetition of queries are very important.

References

[1] Savage, Dwayne C. "Microbial ecology of the gastrointestinal tract." Annual Reviews in Microbiology 31, no. 1 (1977): 107-133.

[2] Gerritsen, Jacoline, Hauke Smidt, Ger T. Rijkers, and Willem M. Vos. "Intestinal microbiota in human health and disease: the impact of probiotics." Genes & nutrition 6, no. 3 (2011): 209.

[3] Curtis, Thomas P., William T. Sloan, and Jack W. Scannell. "Estimating prokaryotic diversity and its limits." Proceedings of the National Academy of Sciences 99, no. 16 (2002): 10494-10499.

[4] Dykhuizen, Daniel E. "Santa Rosalia revisited: why are there so many species of bacteria?." Antonie van Leeuwenhoek 73, no. 1 (1998): 25-33.

[5] Schloss, Patrick D., and Jo Handelsman. "Status of the microbial census." Microbiology and Molecular Biology Reviews 68, no. 4 (2004): 686-691.

[6] Wagner, Michael, Rudolf Amann, Hilde Lemmer, and Karl-Heinz Schleifer. "Probing activated sludge with oligonucleotides specific for proteobacteria: inadequacy of culture-dependent methods for describing microbial community structure." Applied and environmental microbiology 59, no. 5 (1993): 1520-1525.

[7] Amann, Rudolf I., Wolfgang Ludwig, and Karl-Heinz Schleifer. "Phylogenetic identification and in situ detection of individual microbial cells without cultivation." Microbiological reviews 59, no. 1 (1995): 143-169.

[8] Ben-Dov, Eitan, Esti Kramarsky-Winter, and Ariel Kushmaro. "An in situ method for cultivating microorganisms using a double encapsulation technique." FEMS microbiology ecology 68, no. 3 (2009): 363-371.

[9] Saiki, Randall K., Stephen Scharf, Fred Faloona, Kary B. Mullis, Glenn T. Horn, Henry A. Erlich, and Norman Arnheim. "Enzymatic amplification of b-globin genomic sequences and restriction site analysis for diagnosis of sickle cell anemia." Science 230, no. 4732 (1985): 1350-1354.

[10] Powledge, Tabitha M. "The polymerase chain reaction." Advances in physiology education 28.2 (2004): 44-50.

[11] Al-Soud, Waleed Abu, and Peter Rådström. "Capacity of nine thermostable DNA polymerases to mediate DNA amplification in the presence of PCR-inhibiting samples." Applied and environmental microbiology 64, no. 10 (1998): 3748-3753.

[12] Bolognia, Jean L., Dennis L. Cooper, and Earl J. Glusac. "Toxic erythema of chemotherapy: a useful clinical term." Journal of the American Academy of Dermatology 59, no. 3 (2008): 524-529.

[13] Smith, Cindy J., and A. Mark Osborn. "Advantages and limitations of quantitative PCR (Q-PCR)-based approaches in microbial ecology." FEMS microbiology ecology 67, no. 1 (2009): 6-20.

[14] Garibyan, Lilit, and Nidhi Avashia. "Polymerase chain reaction." Journal of Investigative Dermatology 133, no. 3 (2013): 1-4.

[15] Schrenzel, Jacques, Tanja Kostic, Levente Bodrossy, and Patrice Francois. "Introduction to Microarray-Based Detection Methods." Detection of Highly Dangerous Pathogens: Microarray Methods for BSL 3 and BSL 4 Agents (2009): 1-34.

[16] Wooley, John C., Adam Godzik, and Iddo Friedberg. "A primer on metagenomics." PLoS Comput Biol 6, no. 2 (2010): e1000667.

[17] Van de Peer, Yves, Sabine Chapelle, and Rupert De Wachter. "A quantitative map of nucleotide substitution rates in bacterial rRNA." Nucleic acids research 24, no. 17 (1996): 3381-3391.

[18] Klindworth, Anna, Elmar Pruesse, Timmy Schweer, Jörg Peplies, Christian Quast, Matthias Horn, and Frank Oliver Glöckner. "Evaluation of general 16S ribosomal RNA gene PCR primers for classical and next-generation sequencing-based diversity studies." Nucleic acids research 41, no. 1 (2013): e1-e1.

[19] Větrovský, Tomáš, and Petr Baldrian. "The variability of the 16S rRNA gene in bacterial genomes and its consequences for bacterial community analyses." PloS one 8, no. 2 (2013): e57923.

[20] Reyes-López, Miguel Ángel, Alfonso Méndez-Tenorio, Rogelio Maldonado-Rodríguez, Mitchel J. Doktycz, James T. Fleming, and Kenneth L. Beattie. "Fingerprinting of prokaryotic 16S rRNA genes using oligodeoxyribonucleotide microarrays and virtual hybridization." Nucleic acids research 31, no. 2 (2003): 779-789.

[21] Klappenbach, Joel A., Paul R. Saxman, James R. Cole, and Thomas M. Schmidt. "rrndb: the ribosomal RNA operon copy number database." Nucleic acids research 29, no. 1 (2001): 181-184.

[22] Acinas, Silvia G., Luisa A. Marcelino, Vanja Klepac-Ceraj, and Martin F. Polz. "Divergence and redundancy of 16S rRNA sequences in genomes with multiple rrn operons." Journal of bacteriology 186, no. 9 (2004): 2629-2635.

[23] Pei, Anna Y., William E. Oberdorf, Carlos W. Nossa, Ankush Agarwal, Pooja Chokshi, Erika A. Gerz, Zhida Jin et al. "Diversity of 16S rRNA genes within individual prokaryotic genomes." Applied and environmental microbiology 76, no. 12 (2010): 3886-3897.

[24] Rusch, Douglas B., Aaron L. Halpern, Granger Sutton, Karla B. Heidelberg, Shannon Williamson, Shibu Yooseph, Dongying Wu et al. "The Sorcerer II global ocean sampling expedition: northwest Atlantic through eastern tropical Pacific." PLoS biology 5, no. 3 (2007): e77.

[25] Yilmaz, Pelin, Renzo Kottmann, Elmar Pruesse, Christian Quast, and Frank Oliver Glöckner. "Analysis of 23S rRNA genes in metagenomes–A case study from the Global Ocean Sampling Expedition." Systematic and applied microbiology 34, no. 6 (2011): 462-469.

[26] Logares, Ramiro, Shinichi Sunagawa, Guillem Salazar, Francisco M. Cornejo-Castillo, Isabel Ferrera, Hugo Sarmento, Pascal Hingamp et al. "Metagenomic 16S rDNA Illumina tags are a powerful alternative to amplicon sequencing to explore diversity and structure of microbial communities." Environmental microbiology (2013).

[27] Siqueira Jr, Jose F., Ashraf F. Fouad, and Isabela N. Roˆcas. "Pyrosequencing as a tool for better understanding of human microbiomes." Journal of oral microbiology 4 (2012).

[28] Welch, R. A., V. Burland, G. Plunkett, P. Redford, P. Roesch, D. Rasko, E. L. Buckles, S. R. Liou, A. Boutin, J. Hackett, D. Stroud, G. F. Mayhew, D. J. Rose, S. Zhou, D. C. Schwartz, N. T. Perna, H. L. T. Mobley, M. S. Donnenberg, and F. R. Blattner. 2002. Extensive mosaic structure revealed by the complete genome sequence of uropathogenic Escherichia coli. Proc. Natl. Acad. Sci. U. S. A. 99:17020-17024.

[29] Robinson, Courtney J., Brendan JM Bohannan, and Vincent B. Young. "From structure to function: the ecology of host-associated microbial communities." Microbiology and Molecular Biology Reviews 74, no. 3 (2010): 453-476.

[30] Sanger F, Nicklen S, Coulson AR. DNA sequencing with chain-terminating inhibitors.PNAS 1977;74:5463-5467.

[31] Gene expression analysis by massively parallel signature sequencing (MPSS) on microbead arrays Nature Biotechnology 18, 630 - 634 (2000).

[32] Venter, J. Craig, Mark D. Adams, Eugene W. Myers, Peter W. Li, Richard J. Mural, Granger G. Sutton, Hamilton O. Smith et al. "The sequence of the human genome." science 291, no. 5507 (2001): 1304-1351.

[33] Lander, Eric S., Lauren M. Linton, Bruce Birren, Chad Nusbaum, Michael C. Zody, Jennifer Baldwin, Keri Devon et al. "Initial sequencing and analysis of the human genome." Nature 409, no. 6822 (2001): 860-921.

[34] Wheeler, David L., Tanya Barrett, Dennis A. Benson, Stephen H. Bryant, Kathi Canese, Vyacheslav Chetvernin, Deanna M. Church et al. "Database resources of the national center for biotechnology information." Nucleic acids research 36, no. suppl_1 (2008): D13-D21.

[35] Treangen, Todd J., and Steven L. Salzberg. "Repetitive DNA and next-generation sequencing: computational challenges and solutions." Nature Reviews Genetics 13, no. 1 (2012): 36-46.

[36] Alkan, Can, Saba Sajjadian, and Evan E. Eichler. "Limitations of next-generation genome sequence assembly." Nature methods 8, no. 1 (2011): 61-65.

[37] Bleidorn, Christoph. "Third generation sequencing: technology and its potential impact on evolutionary biodiversity research." Systematics and Biodiversity 14, no. 1 (2016): 1-8

[38] Stein, J., T. Marsh, K. Wu, H. Shizuya, and E. DeLong. 1996. Characterization of uncultivated prokaryotes: isolation and analysis of a 40-kilobasepair genome fragment from a planktonic marine archaeon. J. Bacteriol. 178:591–599

[39] Handelsman, Jo, Michelle R. Rondon, Sean F. Brady, Jon Clardy, and Robert M. Goodman. "Molecular biological access to the chemistry of unknown soil microbes: a new frontier for natural products." Chemistry & biology 5, no. 10 (1998): R245-R249.

[40] Robinson, Courtney J., Brendan JM Bohannan, and Vincent B. Young. "From structure to function: the ecology of host-associated microbial communities." Microbiology and Molecular Biology Reviews 74, no. 3 (2010): 453-476.

[41] Karimi, Ramin, Ladjel Bellatreche, Patrick Girard, Ahcene Boukorca, and Andras Hajdu. "BINOS4DNA: Bitmap Indexes and NoSQL for Identifying Species with DNA Signatures through Metagenomics Samples." In International Conference on Information Technology in Bio-and Medical Informatics, pp. 1-14. Springer International Publishing, 2014.

[42] Thomas, T., Gilbert, J., Meyer, F.: Metagenomics-a guide from sampling to data analysis. Microb Inform Exp 2(3) (2012)

[43] Sharpton, Thomas J. "An introduction to the analysis of shotgun metagenomic data." Frontiers in plant science 5 (2014): 209.

[44] Siegwald, Léa, Hélène Touzet, Yves Lemoine, David Hot, Christophe Audebert, and Ségolène Caboche. "Assessment of Common and Emerging Bioinformatics Pipelines for Targeted Metagenomics." PLoS One 12, no. 1 (2017): e0169563.

[45] Suenaga, Hikaru. "Targeted metagenomics unveils the molecular basis for adaptive evolution of enzymes to their environment." Frontiers in microbiology 6 (2015).

[46] Sharma, Shrikant, Shashank Rana, and Raghvendar Singh. "A SHORT NOTE-METAGENOMICS." International Journal of Biomedical Research 3, no. 4 (2012): 181-186.

[47] McLean, Robert JC, and Kavita S. Kakirde. "Enhancing metagenomics investigations of microbial interactions with biofilm technology." International journal of molecular sciences 14, no. 11 (2013): 22246-22257.

[48] Pallen, Mark J., Nicholas J. Loman, and Charles W. Penn. "High-throughput sequencing and clinical microbiology: progress, opportunities and challenges." Current opinion in microbiology 13, no. 5 (2010): 625-631.

[49] Loman, Nicholas J., Chrystala Constantinidou, Martin Christner, Holger Rohde, Jacqueline Z-M. Chan, Joshua Quick, Jacqueline C. Weir et al. "A culture-independent sequence-based metagenomics approach to the investigation of an outbreak of Shiga-toxigenic Escherichia coli O104: H4." Jama 309, no. 14 (2013): 1502-1510.

[50] Mokili, John L., Forest Rohwer, and Bas E. Dutilh. "Metagenomics and future perspectives in virus discovery." Current opinion in virology 2, no. 1 (2012): 63-77.

[51] Mulcahy-O'Grady, Heidi, and Matthew L. Workentine. "The challenge and potential of metagenomics in the clinic." Frontiers in immunology 7 (2016).

[52] Xia, Xuhua. Bioinformatics and the cell: modern computational approaches in genomics, proteomics and transcriptomics. Springer Science & Business Media, 2007.

[53] Smith, Temple F., and Michael S. Waterman. "Identification of common molecular subsequences." Journal of molecular biology 147, no. 1 (1981): 195-197.

[54] Needleman, Saul B., and Christian D. Wunsch. "A general method applicable to the search for similarities in the amino acid sequence of two proteins." Journal of molecular biology 48, no. 3 (1970): 443-453.

[55] Kemena, Carsten, and Cedric Notredame. "Upcoming challenges for multiple sequence alignment methods in the high-throughput era." Bioinformatics 25, no. 19 (2009): 2455-2465.

[56] Hillier, LaDeana W., Gabor T. Marth, Aaron R. Quinlan, David Dooling, Ginger Fewell, Derek Barnett, Paul Fox et al. "Whole-genome sequencing and variant discovery in C. elegans." Nature methods 5, no. 2 (2008): 183-188.

[57] Howe, Adina, and Patrick SG Chain. "Challenges and opportunities in understanding microbial communities with metagenome assembly (accompanied by IPython Notebook tutorial)." Frontiers in microbiology 6 (2015): 678.

[58] Schloss, Patrick D., and Jo Handelsman. "A statistical toolbox for metagenomics: assessing functional diversity in microbial communities." BMC bioinformatics 9, no. 1 (2008): 34.

[59] Sharpton, Thomas J., Samantha J. Riesenfeld, Steven W. Kembel, Joshua Ladau, James P. O'Dwyer, Jessica L. Green, Jonathan A. Eisen, and Katherine S. Pollard. "PhylOTU: a high-throughput procedure quantifies microbial community diversity and resolves novel taxa from metagenomic data." PLoS Comput Biol 7, no. 1 (2011): e1001061.

[60] Mavromatis, Konstantinos, Natalia Ivanova, Kerrie Barry, Harris Shapiro, Eugene Goltsman, Alice C. McHardy, Isidore Rigoutsos et al. "Use of simulated data sets to evaluate the fidelity of metagenomic processing methods." Nature methods 4, no. 6 (2007): 495-500.

[61] Mende, Daniel R., Alison S. Waller, Shinichi Sunagawa, Aino I. Järvelin, Michelle M. Chan, Manimozhiyan Arumugam, Jeroen Raes, and Peer Bork. "Assessment of metagenomic assembly using simulated next generation sequencing data." PloS one 7, no. 2 (2012): e31386.

[62] Domazet-Lošo, Mirjana, and Bernhard Haubold. "Alignment-free detection of local similarity among viral and bacterial genomes." Bioinformatics 27, no. 11 (2011): 1466-1472.

[63] Chan, Cheong Xin, and Mark A. Ragan. "Next-generation phylogenomics." Biology Direct 8, no. 1 (2013): 3.

[64] Chor, Benny, David Horn, Nick Goldman, Yaron Levy, and Tim Massingham. "Genomic DNA k-mer spectra: models and modalities." Genome biology 10, no. 10 (2009): R108.

[65] Göke, Jonathan, Marcel H. Schulz, Julia Lasserre, and Martin Vingron. "Estimation of pairwise sequence similarity of mammalian enhancers with word neighbourhood counts." Bioinformatics 28, no. 5 (2012): 656-663.

[66] Michael, H., Isidore Rigoutsos, and Mark A. Ragan. "Pattern-Based Phylogenetic Distance Estimation and Tree Reconstruction." Evolutionary Bioinformatics 2006, no. 2 (2007): 0-0.

[67] Vinga, S., Almeida, J.: Alignment-free sequence comparisona review. Bioinformatics 19(4) (2003) 513-523

[68] Srinivasan, S.M., Guda, C.: MetaID: A novel method for identification and quan- ti cation of metagenomic samples. BMC Genomics 14(8) (2013) 1-12

[69] Rødland, Einar Andreas. "Compact representation of k-mer de Bruijn graphs for genome read assembly." BMC bioinformatics 14, no. 1 (2013): 313.

[70] Compeau, Phillip EC, Pavel A. Pevzner, and Glenn Tesler. "How to apply de Bruijn graphs to genome assembly." Nature biotechnology 29, no. 11 (2011): 987-991.

[71] Karimi, Ramin, and Andras Hajdu. "HTSFinder: Powerful Pipeline of DNA Signature Discovery by Parallel and Distributed Computing." Evolutionary bioinformatics online 12 (2016): 73.

[72] Kaderali, Lars, and Alexander Schliep. "Selecting signature oligonucleotides to identify organisms using DNA arrays." Bioinformatics 18, no. 10 (2002): 1340-1349.

[73] Francois, Patrice, Yvan Charbonnier, Jean Jacquet, Dominic Utinger, Manuela Bento, Daniel Lew, Gerhard M. Kresbach, Markus Ehrat, Werner Schlegel, and Jacques Schrenzel. "Rapid bacterial identification using evanescent-waveguide oligonucleotide microarray classification." Journal of microbiological methods 65, no. 3 (2006): 390-403.

[74] Coenye, Tom, and Peter Vandamme. "Use of the genomic signature in bacterial classification and identification." Systematic and applied microbiology 27, no. 2 (2004): 175-185.

[75] Ogilvie, Lesley A., Lucas D. Bowler, Jonathan Caplin, Cinzia Dedi, David Diston, Elizabeth Cheek, Huw Taylor, James E. Ebdon, and Brian V. Jones. "Genome signature-based dissection of human gut metagenomes to extract subliminal viral sequences." Nature communications 4 (2013).

[76] Pride, D. T., Wassenaar, T. M., Ghose, C. & Blaser, M. J. Evidence of host-virus co-evolution in tetranucleotide usage patterns of bacteriophages and eukaryotic viruses. BMC Genomics 7, 8 (2006).

[77] Dick, G. J. et al. Community-wide analysis of microbial genome sequence signatures. Genome Biol. 10, R85 (2009).

[78] Lawrence, Jeffrey G., and Howard Ochman. "Molecular archaeology of the Escherichia coli genome." Proceedings of the National Academy of Sciences 95, no. 16 (1998): 9413-9417.

[79] Sandberg, Rickard, Carl-Ivar Bränden, Ingemar Ernberg, and Joakim Cöster. "Quantifying the species-specificity in genomic signatures, synonymous codon choice, amino acid usage and G+ C content." Gene 311 (2003): 35-42.

[80] Duhaime, M. B., Wichels, A., Waldmann, J., Teeling, H. & Glöckner, F. O. Ecogenomics and genome landscapes of marine Pseudoalteromonas phage H105/1. ISME J. 5, 107–112 (2011).

[81] Saeed, I., Tang, S.-L. & Halgamuge, S. K. Unsupervised discovery of microbial population structure within metagenomes using nucleotide base composition. Nucleic Acid Res. 40, e34 (2011).

[82] Teeling, H., Meyerdierks, A., Bauer, M., Amann, R. & Glöckner, F. O. Application of tetranucleotide frequencies for the assignment of genomic fragments. Environ. Microbiol. 6, 938–947 (2004).

[83] Ghai, R. et al. New abundant microbial groups in aquatic hypersaline environments. Sci. Rep. 1, 135 (2011).

[84] Pignatelli, M. et al. Metagenomics reveals our incomplete knowledge of global diversity. Bioinformatics 24, 2124–2125 (2008)]

[85] Scherer, Stewart, Mary Sara McPeek, and Terence P. Speed. "Atypical regions in large genomic DNA sequences." Proceedings of the National Academy of Sciences 91, no. 15 (1994): 7134-7138.

[86] Dufraigne, Christine, Bernard Fertil, Sylvain Lespinats, Alain Giron, and Patrick Deschavanne. "Detection and characterization of horizontal transfers in prokaryotes using genomic signature." Nucleic Acids Research 33, no. 1 (2005): e6-e6.

[87] van Passel, Mark WJ, Eiko E. Kuramae, Angela CM Luyf, Aldert Bart, and Teun Boekhout. "The reach of the genome signature in prokaryotes." BMC evolutionary biology 6, no. 1 (2006): 84.

[88] Mrázek, Jan, and Samuel Karlin. "Distinctive features of large complex virus genomes and proteomes." Proceedings of the National Academy of Sciences 104, no. 12 (2007): 5127-5132.

[89] Sumathi, S., and S. N. Sivanandam. "Introduction to data mining principles." Introduction to Data Mining and its Applications (2006): 1-20.

[90] Silberschatz, Abraham, Peter B. Galvin, Greg Gagne, and A. Silberschatz. Operating system concepts. Vol. 4. Reading: Addison-wesley, 1998.

[91] Coulouris, George F., Jean Dollimore, and Tim Kindberg. Distributed systems: concepts and design. pearson education, 2005.

[92] Jia, Weijia, and Wanlei Zhou. Distributed network systems: from concepts to implementations. Vol. 15. Springer Science & Business Media, 2004.

[93] Pike, Robert C., and Kenneth L. Thompson. "Distributed computing system." U.S. Patent 5,623,666, issued April 22, 1997.

[94] Tanenbaum, Andrew S., and Maarten Van Steen. Distributed systems: principles and paradigms. Prentice-Hall, 2007.

[95] Davenport, Thomas H., and James E. Short. "The new industrial engineering: information technology and business process redesign." (1990).

[96] Symonds, Judith, ed. Ubiquitous and Pervasive Computing: Concepts, Methodologies, Tools, and Applications: Concepts, Methodologies, Tools, and Applications. IGI Global, 2009.

[97] Loke, Seng. Context-aware pervasive systems: architectures for a new breed of applications. CRC Press, 2006.

[98] Pacheco, Peter. An introduction to parallel programming. Elsevier, 2011.

[99] Kumar, Vipin, Ananth Grama, Anshul Gupta, and George Karypis. Introduction to parallel computing: design and analysis of algorithms. Vol. 400. Redwood City, CA: Benjamin/Cummings, 1994.

[100] Bloom, Kenneth, and Richard Gerber. "Report of the Snowmass 2013 Computing Frontier Working Group on Distributed Computing and Facility Infrastructures." arXiv preprint arXiv:1311.2208 (2013).

[101] Flynn, Michael J. "Some computer organizations and their effectiveness." IEEE transactions on computers 100, no. 9 (1972): 948-960.

[102] Rajaraman, V. Elements of parallel computing. PHI Learning Pvt. Ltd., 2006.

[103] Irabashetti, Prabhudev S. "Parallel processing in processor organization." International Journal of Advanced Research in Computer and Communication Engineering 3, no. 1 (2014): 5150-5153.

[104] Tsuchiyama, Ryoji, Takashi Nakamura, Takuro Iizuka, and Akihiro Asahara. "Jeongdo Son, and Satoshi Miki." The opencl programming book, Fixstars (2010).

[105] Wilkinson, Barry, and Michael Allen. Parallel Programing. Vol. 999. Upper Saddle River, NJ: Prentice hall, 1999.

[106] Levinthal, Adam E., Thomas K. Porter, Thomas DS Duff, and Loren C. Carpenter. "Selective operation of processing elements in a single instruction multiple data stream (SIMD) computer system." U.S. Patent 5,045,995, issued September 3, 1991.

[107] Collins, Jamison, Perry Wang, Bernard Lint, Koichi Yamada, Asit Mallick, Richard A. Hankins, and Gautham Chinya. "Enabling multiple instruction stream/multiple data stream extensions on microprocessors." U.S. Patent 7,768,518, issued August 3, 2010.

[108] Chiappetta, Shawn JA. Non-overlapping domain decomposition parallel algorithms for convection-diffusion problems. The University of Wisconsin-Milwaukee, 2009.

[109] Mattson, Timothy G., Beverly Sanders, and Berna Massingill. Patterns for parallel programming. Pearson Education, 2004.

[110] Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." Communications of the ACM 51, no. 1 (2008): 107-113.

[111] Patterson, David A., and John L. Hennessy. Computer organization and design: the hardware/software interface. Newnes, 2013.

[112] Apache Hadoop. Available at: http://hadoop.apache.org/.

[113] White T. Hadoop: The Definitive Guide. O'Reilly Media, Inc.; 2012.

[114] Cloudera. Hadoop and Big Data. Available at: http://www.cloudera.com/content/cloudera/en/about/hadoop-and-big-data.html.

[115] Shvachko, K., Kuang, H., Radia, S., Chansler, R.: The hadoop distributed file system. In: Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium On, pp. 1-10 (2010)

[116] Frampton, Michael. Big Data made easy: A working guide to the complete Hadoop toolset. Apress, 2014.

[117] Dean, J., Ghemawat, S.: Mapreduce: simplified data processing on large clusters. Communications of the ACM 51(1), 107-113 (2008)

[118] McKenna, Aaron, Matthew Hanna, Eric Banks, Andrey Sivachenko, Kristian Cibulskis, Andrew Kernytsky, Kiran Garimella et al. "The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data." Genome research 20, no. 9 (2010): 1297-1303.

[119] Vavilapalli, Vinod Kumar, Arun C. Murthy, Chris Douglas, Sharad Agarwal, Mahadev Konar, Robert Evans, Thomas Graves et al. "Apache hadoop yarn: Yet another resource negotiator." In Proceedings of the 4th annual Symposium on Cloud Computing, p. 5. ACM, 2013.

[120] Apache Hadoop NextGen MapReduce (YARN). Available at: http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html.

[121] Sriharsha, Ram, Tim Tully, Supreeth Rao, and Reynold Xin. "Spark satellite clusters to hadoop data stores." U.S. Patent Application 14/470,704, filed August 27, 2014.

[122] Capriolo E, Wampler D, Rutherglen J. Programming Hive. O'Reilly Media, Inc.; 2012.

[123] Thusoo A, Sarma JS, Jain N, et al. Hive: a warehousing solution over a mapreduce framework. Proceedings of the VLDB Endowment. 2009;2(2):1626–9.

[124] Apache Hive. Available at: https://hive.apache.org/.

[125] Thusoo, A., Sarma, J.S., Jain, N., Shao, Z., Chakka, P., Zhang, N., Antony, S., Liu, H., Murthy, R.: Hive-a petabyte scale data warehouse using hadoop. In: Data Engineering (ICDE), 2010 IEEE 26th International Conference on, IEEE (2010) 996-1005.

[126] NoSQL Relational Database Management System homepage http://www. strozzi.it/cgi-bin/CSA/tw7/I/en_US/NoSQL/Home%20Page

[127] Michael, M., Moreira, J.E., Shiloach, D., Wisniewski, R.W.: Scale-up x scale-out: A case study using nutch/lucene. In: Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International, IEEE (2007) 1-8.

[128] Bondi, A.B.: Characteristics of scalability and their impact on performance. In: Proceedings of the 2nd international workshop on Software and performance, ACM (2000) 195-203.

[129] Land, Miriam, Loren Hauser, Se-Ran Jun, Intawat Nookaew, Michael R. Leuze, Tae-Hyuk Ahn, Tatiana Karpinets et al. "Insights from 20 years of bacterial genome sequencing." Functional & integrative genomics 15, no. 2 (2015): 141.

[130] Binnewies, Tim T., Yair Motro, Peter F. Hallin, Ole Lund, David Dunn, Tom La, David J. Hampson, Matthew Bellgard, Trudy M. Wassenaar, and David W. Ussery. "Ten years of bacterial genome sequencing: comparative-genomics-based discoveries." Functional & integrative genomics 6, no. 3 (2006): 165-185.

[131] Li, F., Stormo, G.D.: Selection of optimal dna oligos for gene expression arrays. Bioinformatics 17(11),1067-1076 (2001).

[132] Tembe, W., Zavaljevski, N., Bode, E., Chase, C., Geyer, J., Wasieloski, L., Benson, G., Reifman, J.: Oligonucleotide fingerprint identification for microarray-based pathogen diagnostic assays. Bioinformatics 23(1), 5{13 (2007)

[133] Satya, R.V., Zavaljevski, N., Kumar, K., Reifman, J.: A high-throughput pipeline for designing microarray-based pathogen diagnostic assays. BMC bioinformatics 9(1), 185 (2008)

[134] Satya, R.V., Kumar, K., Zavaljevski, N., Reifman, J.: A high-throughput pipeline for the design of real-time pcr signatures. BMC bioinformatics 11(1), 340 (2010)

[135] Vijaya R Satya, Zavaljevski N, Kumar K, Bode E, Padilla S, Wasieloski L, Geyer J, Reifman J: In silico microarray probe design for diagnosis of multiple pathogens. BMC Genomics 2008, 9(1):496.

[136] Phillippy, A.M., Ayanbule, K., Edwards, N.J., Salzberg, S.L.: Insignia: a dna signature search web server for diagnostic assay development. Nucleic acids research, 286 (2009)

[137] Insignia Database and Web Interface. http://insignia.cbcb.umd.edu/

[138] Kurtz, Stefan, Adam Phillippy, Arthur L. Delcher, Michael Smoot, Martin Shumway, Corina Antonescu, and Steven L. Salzberg. "Versatile and open software for comparing large genomes." Genome biology 5, no. 2 (2004): R12.

[139] Bader, K.C., Grothoff , C., Meier, H.: Comprehensive and relaxed search for oligonucleotide signatures in hierarchically clustered sequence datasets. Bioinformatics 27(11), 1546-1554 (2011)

[140] Lee, H.P., Sheu, T.-F., Tang, C.Y.: A parallel and incremental algorithm for efficient unique signature discovery on dna databases. BMC bioinformatics 11(1), 132 (2010)

[141] Lee, H.P., Sheu, T.F., Tsai, Y.T.: Efficient discovery of unique signatures on whole-genome est databases. In: Proceedings of the 2005 ACM Symposium on Applied Computing, pp. 100-104 (2005)

[142] Zheng, J., Close, T.J., Jiang, T., Lonardi, S.: Efficient selection of unique and popular oligos for large est databases. Bioinformatics 20(13), 2101-2112 (2004)

[143] Lee, H.P., Huang, Y.-H., Sheu, T.-F.: Rapid dna signature discovery using a novel parallel algorithm. In: ICCGI 2012, The Seventh International Multi-Conference on Computing in the Global Information Technology, pp. 83-88 (2012)

[144] Lee, H.P., Sheu, T.-F.: An algorithm of discovering signatures from dna databases on a computer cluster. BMC bioinformatics 15(1), 339 (2014)

[145] Marcais, G., Kingsford, C.: A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. Bioinformatics 27(6), 764-770 (2011)

[146] Kaderali, Lars, and Alexander Schliep. "An algorithm to select target specific probes for DNA chips." Bioinformatics 18, no. 10 (2002): 1340-1349.

[147] Rouillard, Jean-Marie, Michael Zuker, and Erdogan Gulari. "OligoArray 2.0: design of oligonucleotide probes for DNA microarrays using a thermodynamic approach." Nucleic acids research 31, no. 12 (2003): 3057-3062.

[148] Wernersson, Rasmus, and Henrik Bjørn Nielsen. "OligoWiz 2.0—integrating sequence feature annotation into the design of microarray probes." Nucleic acids research 33, no. suppl 2 (2005): W611-W615.

[149] Nordberg, Eric K. "YODA: selecting signature oligonucleotides." Bioinformatics 21, no. 8 (2005): 1365-1370.

[150] Ashelford, Kevin E., Andrew J. Weightman, and John C. Fry. "PRIMROSE: a computer program for generating and estimating the phylogenetic range of 16S rRNA oligonucleotide probes and primers in conjunction with the RDP-II database." Nucleic acids research 30, no. 15 (2002): 3481-3489.

[151] Ludwig, Wolfgang, Oliver Strunk, Ralf Westram, Lothar Richter, Harald Meier, Arno Buchner, Tina Lai et al. "ARB: a software environment for sequence data." Nucleic acids research 32, no. 4 (2004): 1363-1371.

[152] Adams, Mark D., and Jenny M. Kelley. "Complementary DNA sequencing: expressed sequence tags and human genome project." Science 252, no. 5013 (1991): 1651.

[153] Baxevanis, Andreas D., and BF Francis Ouellette. Bioinformatics: a practical guide to the analysis of genes and proteins. Vol. 43. John Wiley & Sons, 2004.

[154] Choudhary, M., Mackenzie, C., Nereng, K.S., Sodergren, E., Weinstock, G.M., Kaplan, S.: Multiple chromosomes in bacteria: structure and function of chromosome ii of rhodobacter sphaeroides 2.4. 1t. Journal of bacteriology 176(24), 7694-7702 (1994).

[155] Tótha, A., Karimi, R. “Optimization of Hadoop Cluster for Analyzing Large-scale Sequence Data in Bioinformatics.” Journal of Annales Mathematicae et Informaticae (submitted) 2017.

[156] Menzel, Peter, Kim Lee Ng, and Anders Krogh. "Fast and sensitive taxonomic classification for metagenomics with Kaiju." Nature communications 7 (2016).

[157] Wood, Derrick E., and Steven L. Salzberg. "Kraken: ultrafast metagenomic sequence classification using exact alignments." Genome biology 15, no. 3 (2014): R46.

[158] Ounit, Rachid, Steve Wanamaker, Timothy J. Close, and Stefano Lonardi. "CLARK: fast and accurate classification of metagenomic and genomic sequences using discriminative k-mers." BMC genomics 16, no. 1 (2015): 236.

[159] Karande, N.D.: Efficient indexing technique using bitmap indices for data ware-houses. International Journal 1(4) (2013).

[160] Bellatreche, L., Missaoui, R., Necir, H., Drias, H.: A data mining approach for selecting bitmap join indices. JCSE 1(2) (2007) 177-194.

[161] Metasim Homepage http://ab.inf.uni-tuebingen.de/software/metasim/

[162] Richter, D.C., Ott, F., Auch, A.F., Schmid, R., Huson, D.H.: Metasima sequencing simulator for genomics and metagenomics. PloS one 3(10) (2008) e3373

[163] Karimi, R., and Hajdu, A., "SRIdent: A novel pipeline for real-time identification of species from high-throughput sequencing reads in Metagenomics and clinical diagnostic assays." In Engineering in Medicine and Biology Society (EMBC), 2015 37th Annual International Conference of the IEEE, pp. 6481-6484. IEEE, 2015.

[164] Ulrich, R.L., Ulrich, M.P., Schell, M.A., Kim, H.S., DeShazer, D.: Development of a polymerase chain reaction assay for the specific identification of burkholderia mallei and differentiation from burkholderia pseudomallei and other closely related burkholderiaceae. Diagnostic microbiology and infectious disease 55(1), 37-45 (2006).

[165] Godoy, D., Randle, G., Simpson, A.J., Aanensen, D.M., Pitt, T.L., Kinoshita, R., Spratt, B.G.: Multilocus sequence typing and evolutionary relationships among the causative agents of melioidosis and glanders, burkholderia pseudomallei and burkholderia mallei. Journal of Clinical Microbiology 41(5), 2068-2079 (2003).

[166] Karimi, R., Hajdu, A. "How Can Bring the Big Data Analysis of High-Throughput Sequencing Technologies into the Routine Clinical Diagnostic Assays?" Journal of Annales Mathematicae et Informaticae (submitted) 2017.

Appendix

Download sources and supplementary materials

HTSFinder

The software and supplementary material of **HTSFinder** pipeline are freely available at:

1- https://drive.google.com/file/d/0B8ZRZRHhuPUfMmFNMTlRZTQxeWM/view?usp=sharing

2- https://sourceforge.net/projects/htsfinder/

SRIdent

The software and supplementary material of **SRIdent** pipeline are freely available at:

1- https://sourceforge.net/projects/srident/

2- https://drive.google.com/file/d/0B8ZRZRHhuPUfd0E5dE5oejRhcGs/view?usp=sharing