UNIVERSITY OF CALIFORNIA
RIVERSIDE

Some Clustering and Classification Problems in High-Throughput Metagenomics and Cheminformatics

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Olga Tanaseichuk

December 2013

Dissertation Committee:

Dr. Tao Jiang, Chairperson
Dr. Stefano Lonardi
Dr. Marek Chrobak
Dr. Thomas Girke

The Dissertation of Olga Tanaseichuk is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

I wish to express sincere thanks to all who have influenced the writing of this dissertation.

I am especially grateful to my advisor, Dr. Tao Jiang, for guidance, understanding and patience during my graduate studies. I appreciate the freedom he gave me to choose the research projects as well as his constructive suggestions and critical comments that always worked as motivational boosters. I would like to thank the members of my committee, Dr. Stefano Lonardi, Dr. Marek Chrobak, and Dr. Thomas Girke, for their helpful feedback and invaluable comments. I would also like to thank Dr. Yingyao Zhou, Director of Informatics and IT at the Genomics Institute of the Novartis Research Foundation (GNF), for offering me amazing summer internship opportunities in his group. Furthermore, I am thankful to Nickolay Pihtar, my school mathematics teacher, for rising my interest in science during my mid-school years, and for being my mentor through the years and across the miles.

I would like to thank my friends for their help and support in different aspects of my grad life, and for all the wonderful time we had together: Anton Polishko, Lucy Ulanova and Denis Ulanov, Konstantin Choumiline, Olga Kapralova, Zoia Comarova and Serghei Mangul, Christine Ren, Matt Zimmerman, Mark Vega, Kouyki Takenaka. It has been a pleasure to work with my lab mates: Yu-Ting Huang, Yi-Wen Yang, Li Yan, Wei Li, Dennis Duma, Rachid Ounit, Jonathan Dautrich Jr. and others. Additionally, I am grateful to my "Bonair" friends for being my stress release in the past few month. Thank you all for being there and supporting me during the most difficult times. Without you, this dissertation would be impossible.

To my parents Lyuda and Sergey Tanaseichuk. Thank you for your endless love and support. I love you.

ABSTRACT OF THE DISSERTATION

Some Clustering and Classification Problems in High-Throughput Metagenomics and Cheminformatics

by

Olga Tanaseichuk

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, December 2013
Dr. Tao Jiang, Chairperson

In this dissertation, we address three different problems in high-throughput metagenomics and cheminformatics.

(1) Metagenomics studies the genomic content of an entire microbial community by simultaneously sequencing all genomes in an environmental sample. The advent of next-generation sequencing (NGS) technologies has drastically reduced sequencing time and cost, leading to the generation of millions of sequences (reads) in a single run. An important problem in metagenomic analysis is to determine and quantify species (or genomes) in a metagenomic sample. The problem is challenging due to an unknown number of genomes and their abundance ratios, presence of repeats and sequencing errors, and the short length of NGS reads. We propose two algorithms to address these challenges. First, we present an algorithm for separating short paired-end reads from genomes with similar abundance levels. Second, we propose a method to accurately estimate the abundance levels of species. The algorithm automatically determines the number of abundance groups in a metagenomic dataset and bins the reads into these groups.

(2) NGS coupled with metagenomics has led to the rapid growth of sequence databases and enabled a new branch of microbiology called comparative metagenomics. It is a fast growing field that requires the development of novel supervised learning techniques. In particular, the problem of microbial community classification may have useful applications enabling efficient organization and search in rapidly growing metagenomic databases, detection of disease phenotypes in clinical samples, and forensic identification. We propose a novel supervised classification method for metagenomic samples that takes advantage of the natural structure in microbial community data encoded by a phylogenetic tree.

(3) In modern drug discovery, ultra-high-throughput screening is applied to millions of drug-like compounds in one experiment. Hierarchical clustering is an important step in the drug discovery process. Standard implementations of the exact algorithm for hierarchical clustering require $O(n^2)$ time and $O(n^2)$ memory. Even though approximate hierarchical clustering methods overcome this problem, they either rely on embedding into spaces that are not biologically sensible, or produce very low resolution hierarchical structures. We present a hybrid hierarchical clustering algorithm requiring approximately $O(n\sqrt{n})$ time and $O(n\sqrt{n})$ memory while still preserving the most desirable properties of the exact algorithm.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 High-Throughput Technologies in Bio- and Cheminformatics

Advances in biotechnology have revolutionized biomedical research, allowing to uncover various biological questions unattainable with conventional methods. Automation of classical cell biology techniques has lead to the development of high-throughput technologies that allow to perform simultaneous measurements of biological molecules at an unprecedented scale. For example, the recent advent of *next-generation sequencing (NGS)* technologies [76, 11] has drastically reduced sequencing time and cost, producing millions of reads in a single run. The increased sequencing depth has made it possible to attempt genome-wide discovery in genomic sequences and whole transcriptomes at a single nucleotide level [49]. Deep sequencing also allows to study complex microbial populations at a high resolution, thus enabling detection of rare species [126], and providing a deep insight into phylogenetic composition and functional diversity of microbial communities. *High-throughput* and *ultra-high-throughput screening* (*HTS* and

uHTS) technologies [50] are examples of technological innovations in drug discovery. The invention of HTS and uHTS has made it possible to screen hundreds of thousands of chemical compounds against a biological target. The large volumes of data generated by high-throughput technologies require the development of novel computational techniques and tools to efficiently analyze the data.

In this dissertation, we address three different problems that arise in high-throughput metagenomics and cheminformatics: how to separate genomic sequences (reads) from different organisms in a metagenomic sample, how to take advantage of the phylogeny to classify new metagenomic samples, and how to efficiently cluster large datasets of drug-like chemical compounds. Brief introductions to each of the three problems are presented in Sections 1.2, 1.3 and 1.4, respectively.

1.2 Binning Metagenomic Sequences

The diversity of the microbial world had been hidden from the eyes of scientists until the advent of metagenomics. Despite the vital roles of microbes in our planet's ecology, evolution and human health, large populations of bacteria remain poorly characterized because the majority of bacterial species have not been successfully cultivated [3]. The metagenomics approach has offered a remedy: bypassing the need for isolation and cultivation, all the sequences present in an environmental sample are sequenced simultaneously, making it possible to access the genetic information of otherwise hidden organisms. It provides hope for a better understanding of natural diversity of microorganisms as well as their roles and interactions. It also opens new opportunities for medicine, biotechnology, agricultural studies and ecology.

The field of metagenomics has been advanced by the recent improvements in DNA sequencing technologies. The advent of NGS technologies [76, 11] has drastically improved sequencing time and cost, leading to an exponential increase in environmental sequencing data which makes it possible to study microbial communities at a much higher resolution due to increased sequencing depth [97]. The drawback of the NGS technology is that read length is reduced.

In metagenomics, a sample contains sequence reads from various organisms. Therefore, an important problem in a metagenomic analysis is to determine and quantify the species (or genomes) in a sample. The identification of phylogenetically related groups of reads in a metagenomic dataset is usually referred to as *binning*. Among the existing computational tools for metagenomic analysis, there are similarity-based methods that use reference databases to align reads, and composition-based methods that use composition patterns (*i.e.*, frequencies of short words or l-mers) to cluster reads. Similarity-based methods are unable to classify reads from unknown species without close references (which constitute the majority of reads). Since composition patterns are preserved only in significantly large fragments, composition-based tools cannot be used for very short reads, which becomes a significant limitation with the development of NGS. Recently, several new metagenomic binning algorithms that can deal with NGS reads and do not rely on reference databases were developed. However, all of them have difficulty with handling samples containing low-abundance species.

In Chapter 2, we present a two-phase heuristic algorithm for separating short paired-end reads from different genomes in a metagenomic dataset, called TOSS (*i.e.*, TOol for Separating Short reads). The algorithm could handle very short reads and sequencing errors. Initially, it was designed to handle genomes with similar abundance levels. We also propose a new method to accurately estimate the abundance levels of

species based on a novel probabilistic model for counting l-mer frequencies in a metagenomic dataset. This method automatically determines the number of abundance groups in a dataset and bins the reads into these groups. Finally, we incorporate the abundance-based binning method into TOSS to enhance its performance.

1.3 Classification of Metagenomic Samples

NGS technologies have led to the rapid increase in the number and sizes of metagenomic sequencing projects. Exponential growth of sequence data has enabled comparative analysis of microbial communities, leading to a new branch of microbiology called comparative metagenomics. Comparative analysis extends insights into the structure and function of microbial communities: it may help to identify community specific properties of different environments as well as discriminative properties between different conditions, and to determine how microbial community composition is affected by specific environmental changes. Comparative metagenomics has broad implications for various fields of environmental science and human biology. It may help to address the intriguing question of identifiability of a core human microbiome [102], to understand the relationship between human microbiome and health, and to study how microbial composition and function vary between distinct body sites and across the human population.

To access the taxonomic composition of a metagenomic sample, both single marker gene sequencing and whole community shotgun sequencing are widely used [57]. Each metagenomic sample is represented as a list of *operational taxonomic units (OTUs)* and their frequencies. Comparative analysis may involve the identification of compositional patterns across samples from similar environments as well as discriminatory fea-

tures between different communities, associations between human bacterial communities and disease phenotypes, prediction of unknown labels for new samples, *etc.* These tasks require the development of new supervised learning techniques that would take into consideration challenges associated with metagenomic data. One of the challenges is that features defined by OTUs do not necessarily represent specific taxonomic units because taxonomic levels are hard to define due to the fact that only relatively a small number of bacteria have been cultured. Moreover, it is hard to determine which taxonomic resolution level provides features with the best discriminative or predictive properties [61]. The environment-specific patterns may even be comprised of different lineages at varying phylogenetic depth. Finally, a low overlap in species between samples results in sparse and high dimensional feature vectors. On the other hand, the natural properties of microbial community data may provide useful information about the structure of the data. For example, similarity between species encoded by a phylogenetic tree captures the relationship between OTUs and may be useful for the analysis of complex microbial datasets where the diversity patterns are comprised of features at multiple taxonomic levels. Even though some of the challenges have been addressed by learning algorithms in the literature, none of the available methods takes advantage of the inherent properties of metagenomic data.

In Chapter 3, we propose a novel supervised classification method for metagenomic samples that takes advantage of the natural structure in microbial community data encoded by a phylogenetic tree. This model allows us to take advantage of environment-specific compositional patterns that may contain features at multiple granularity levels. Additionally, we propose a new simulation framework for generating metagenomic read counts that may be useful in comparative metagenomics research.

Our experimental results on simulated and real data show that the phylogenetic information used in our method improves the classification accuracy.

1.4 Hierarchical Clustering in Cheminformatics

The process of drug discovery has been revolutionized with the advent of HTS anf uHTS technologies that use automation to quickly assay the biochemical activity of a large number of drug-like compounds in a fast and cost-effective manner. uHTS allows to screen several million compounds in one experiment [50]. Clustering of compound libraries is a fundamental task in cheminformatics. Among all clustering methods, hierarchical clustering and k-means clustering are arguably the two most popular algorithms used due to their simplicity in result interpretation. In the cheminformatics field, Wards clustering [111] and Jarvis-Patrick clustering [88] are corresponding algorithms similar in spirit to hierarchical clustering and k-mean clustering, respectively. Although there is no definitive answer as to which algorithm is more accurate, hierarchical clustering has been applied more often in cheminformatics research because of its deterministic property and flexibility in flattening the resultant tree at different cutoff levels.

However, applying hierarchical clustering to large datasets is rather challenging. First, compared to the linear complexity of the k-means algorithm, the most popular average-linkage hierarchical clustering (AHC) requires $O(n^2)$ time; we even observed $O(n^3)$-time implementations in some popular bioinformatics tools [33]. Second, it requires $O(n^2)$ memory [81], which limits the number of input data points to $\sim 20\,000$ for a typical desktop computer. In cheminformatics research, modern drug discovery applies uHTS for several million compounds in one experiment [50]. Two problems arise from uHTS. First, to expand the screening compound collection, vendor catalogs of millions of

compounds ideally should be hierarchically clustered and prioritized for acquisition. But in practice, cheminformaticians resort to a greedy algorithm such as Sphere Exclusion [45], which relies on a predetermined similarity threshold. Second, instead of analyzing all compound profiles across a panel of screening assays, hierarchical clustering analyses have usually been compromised and restricted to $\sim$ 20 000 top screening hits due to memory limitations. Therefore, there exists a significant need to develop a hierarchical clustering algorithm for large datasets.

In Chapter 4, we present a hybrid hierarchical clustering algorithm requiring approximately $O(n\sqrt{n})$ time and $O(n\sqrt{n})$ memory while still preserving the most desirable properties of the exact *agglomerative hierarchical clustering (AHC)* algorithm. The algorithm was capable of clustering one million compounds within a few hours on a single processor.

1.5 Publications

This dissertation encompasses four publications. The TOSS paper (Chapter 2) is published in the 11th Workshop on Algorithms in Bioinformatics (WABI 2011) and in Algorithms for Molecular Biology. The abundance-based binning paper (Chapter 2) is published in the 12th Workshop on Algorithms for Bioinformatics (WABI 2012). The Phylogeny-Based classification paper (Chapter 3) has been accepted in Bioinformatics with minor revisions. The complete list of publications includes:

- Olga Tanaseichuk, James Borneman and Tao Jiang. Separating Metagenomic Short Reads into Genomes via Clustering. 11th Workshop on Algorithms for Bioinformatics (WABI 2011), Lecture Notes in Bioinformatics, 6833:298-313, Springer

Berlin/Heidelberg, 2011. Also appears in Algorithms for Molecular Biology, 2012, 7(1):27

- Olga Tanaseichuk, James Borneman and Tao Jiang. Probabilistic Approach to Accurate Abundance-Based Binning of Metagenomic Reads. 12th Workshop on Algorithms for Bioinformatics (WABI 2012), Lecture Notes in Bioinformatics, 7534:404-416, Springer Berlin/Heidelberg, 2012.

- Olga Tanaseichuk, James Borneman and Tao Jiang. Phylogeny-Based Classification of Microbial Communities. Accepted in Bioinformatics.

- Olga Tanaseichuk, Alireza Hadj Khodabakshi, Dimitri Petrov, Jianwei Che, Tao Jiang, Bin Zhou, Andrey Santrosyan and Yingyao Zhou. An Efficient Hierarchical Clustering Algorithm for Large Datasets. To be submitted to PLoS ONE.

Chapter 2

Binning of Metagenomic Short Reads

2.1 Introduction

Metagenomics [47] is a new field of study that provides a deeper insight into the microbial world compared to the traditional single-genome sequencing technologies. Traditional methods for studying individual genomes are well developed. However, they are not appropriate for studying microbial samples from the environment because traditional methods rely upon cultivated clonal cultures while more than 99% of bacteria are unknown and cannot be cultivated and isolated [90]. Metagenomics uses technologies that sequence uncultured bacterial genomes in an environmental sample directly [8], and thus makes it possible to study organisms which cannot be isolated or are difficult to grow in a lab.

Many well-known metagenomics projects use the whole genome shotgun sequencing approach in combination with Sanger sequencing technologies. This approach has produced datasets from the Sargasso Sea [106], Human Gut Microbiome [43] and

Acid Mine Drainage Biofilm [104]. However, new sequencing technologies have evolved over the past few years. The sequencing process has been greatly parallelized, producing millions of reads with much faster speed and lower cost. The exponential increase in environmental sequencing data makes it possible to study microbial communities at a much higher resolution due to increased sequencing depth [97]. NGS-based approaches have recently been applied to sequence several metagenomes from cow rumen [51], saliva microbiome [122], permafrost [73], etc.

The primary goals of metagenomics are to describe the populations of microorganisms and to identify their roles in the environment. Ideally, we want to identify complete genomic sequences of all organisms present in a sample. However, metagenomic data is very complex, containing a large number of sequence reads from many species. The number of species and their abundance levels are unknown. The assembly of a single genome is already a difficult problem, complicated by repeats and sequencing errors which may lead to high fragmentation of contigs and misassembly. In a metagenomic data, in addition to repeats within individual genomes, genomes of closely related species may also share homologous sequences, which could lead to even more complex repeat patterns that are very difficult to resolve. A lot of research has been done for assembling single genomes [22, 112, 35, 96]. But due to the lack of research on metagenomic assemblers, assemblers designed for individual genomes are routinely used in metagenomic projects [106, 104]. It has been shown that these assemblers may lead not only to misassembly, but also severe fragmentation of contigs [26]. A plausible approach to improve the performance of such assemblers is to separate reads from different organisms present in a dataset before the assembly.

Many computational tools have been developed for separating reads from different species or groups of related species (we will refer to the problem as the cluster-

ing of reads). Some of the tools also estimate the abundance levels and genome sizes of species. These tools are usually classified as similarity-based (or phylogeny-based) and composition-based. Similarity-based methods explore the taxonomic composition of metagenomic sequences by performing similarity search against databases of known genomes, genes and proteins [52, 62, 42, 79]. Small-scale approaches involving 16S rRNAs and 18S rRNAs [23] are commonly used to determine evolutionary relationships by analyzing fragments that contain marker genes and comparing them with known marker genes. These methods take advantage of small number of fragments containing marker genes and require reads to have at least 1000 bps. Two other tools handle a larger number of fragments: MEGAN [52] and CARMA [62]. MEGAN aligns reads to databases of known sequences using BLAST [2] and assigns reads to taxa by the lowest common ancestor approach. CARMA performs phylogenetic classification of unassembled reads using all Pfam domains and protein families as phylogenetic markers. These two methods have high accuracy and are suitable for very short reads (as short as 35 bps for MEGAN and 80 bps for CARMA). However, they rely on the availability of reference databases, while a lot of organisms in a sample may not be remotely related to any known species. As a consequence, a large fraction of read data may remain unclassified.

The second class of methods use compositional properties of the fragments (or reads). These methods are based on the fact that some composition properties, such as CG content and oligonucleotide frequencies are preserved across sufficiently long fragments of the same genome, and vary significantly between fragments from different organisms. K-mer frequency is the most widely used characteristics for binning. For example, the method in [130] utilizes the property that each genome has a stable distribution of k-mer frequencies for $k = 1..6$ in fragments as short as 1000 bps. It shows that these fragments have very similar "barcodes" and thus can be clustered based on

their barcode similarities. Barcode similarity also correlates with phylogenetic closeness between genomes. The main challenge in the k-mer frequency approach is that these frequencies produce large feature vectors, which can be even larger than the sizes of fragments. Different methods have been proposed to deal with this problem. CompostBin [27], which uses hexamer frequencies, adopts a modified principle component analysis to extract the top three meaningful components and then cluster the reads based on principal component values. Self-organizing maps are another way to reduce dimensionality by mapping multidimensional data to two dimensional space. The work in [24] uses SOMs for tri- and tetranucleotide frequency vectors. In TETRA [99], z-scores are computed for tetranucleotide frequencies and fragments are classified by the Pearson correlation of their z-scores. MetaCluster 3.0 [67] uses Spearman Footrule distance between k-mer feature vectors. Another composition feature is used in TACOA [34]: the ratio between observed oligonucleotide frequencies and expected frequencies given the CG content. To cluster fragments, the k-NN approach is combined with the Gaussian kernel function. Composition-based methods can accurately bin long fragments. However, due to local variation of DNA composition across a genome, the performance of these methods degrades with the decrease of the read length, making them unsuitable for NGS datasets. In general, these methods are not suitable for fragments shorter than 1000 bps [12].

Several recent unsupervised metagenomic binning algorithms have been developed to handle short NGS reads. In particular, MetaCluster 4.0 [110] exploits compositional properties of groups of reads rather than individual reads. Although it handles high-abundance species well, it does not perform well on datasets with low-abundance species. MetaCluster 5.0 [109] is an extension of MetaCluster 4.0 to deal with low-abundant species. Another unsupervised binning algorithm that handles NGS reads

is AbundanceBin [119]. It is designed to separate reads from genomes with different abundance levels. It computes frequencies of all l-mers in a metagenomic dataset and, assuming that these frequencies come from a mixture of Poisson distributions, predicts the abundance levels of genomes and clusters l-mers according to their frequencies. Then reads are clustered based on the frequencies of their l-mers. The limitation is that genomes whose abundance levels do not differ very much (within ratio 1:2) will not be separated.

In this chapter, we present two algorithms. The first algorithm is a two-phase heuristic algorithm for separating short paired-end reads from different organisms in a metagenomic dataset, called TOSS (*i.e.*, TOol for Separating Short reads). The basic algorithm is developed to separate genomes with similar abundance levels. It is based on several interesting observations about unique and repeated l-mers in a metagenomic dataset, which enables us to separate unique l-mers (each of which belongs to only one genome and is not repeated) from repeats (l-mers which are repeated in one or more genomes) at the beginning of the first phase of the algorithm. During the first phase, unique l-mers are clustered so that each cluster consists of l-mers from only one of the genomes. This is possible due to the observation that most l-mers are unique within a genome and, moreover, within a metagenomic dataset. During the second phase, we find connections between clusters through repeated regions and then merge clusters of l-mers that are likely to belong to the same organism. Finally, reads are assigned to clusters. We test the method on a large number of simulated metagenomic datasets for microbial species with various phylogenetic closeness according to the NCBI taxonomy [114, 10] and show that genomes can be separated if the number of common repeats is less then the number of genome-specific repeats. For example, genomes of different species of the same genus often have a large number of common repeats and thus are

very hard to separate. In the tests, our method is able to separate fewer than a half of groups of such closely related genomes. However, with the decrease in the fraction of common repeats, the ability to accurately separate genomes significantly increases. Due to the lack of appropriate short read clustering tools for comparison, we modify a well-known genome assembly software, Velvet [125], to make it behave like a genome separation tool and compare our clustering results with those of the modified Velvet.

The second algorithm is designed to automatically determine the number of abundance groups in the dataset and bin the reads into these groups. The method is based on a novel probabilistic model for counting l-mer frequencies in the metagenomic dataset that takes into account the frequencies of erroneous l-mers as well as repeats. An *expectation maximization (EM)* algorithm is used to learn the parameters of the model. We show that the method outperforms AbundanceBin on simulated and real datasets.

Finally, to handle metagenomic datasets with genomes of arbitrary abundance ratios, we incorporate the abundance-based binning step into TOSS to improve its performance in the presence of genomes with different abundance levels. The integrated method works for very short reads, and is able to handle multiple genomes with arbitrary abundance levels and sequencing errors. We compare the improved TOSS against recent metagenomic binning tools MetaCluster 4.0 and MetaCluster 5.0 on simulated NGS datasets, and show that it has a comparable performance overall but often achieves a better sensitivity and breaks fewer genomes.

The chapter is organized as follows. In Section 2.2.1, we introduce TOSS. Section 2.2.2 presents our abundance-based binning algorithm. In Section 2.2.3, we describe how to incorporate an abundance-based pre-processing step into TOSS. Section

2.3 gives the experimental evaluation of our methods and comparisons to other recent binning tools. Section 2.4 concludes the chapter.

2.2 Methods

2.2.1 Separating Metagenomic Short Reads from Genomes with Similar Abundance Levels

The algorithm we are going to present in Section 2.2.1, TOSS, is based on l-mers from metagenomic reads. We will first discuss some properties of l-mers that are important for our algorithm, and also make some important observations that lead to the intuition behind the algorithm. Then we will present the algorithm itself. In Section 2.2.3 we will extend the method to handle arbitrary abundance ratios.

2.2.1.1 Preliminaries

First, let us analyze the expected number of occurrences of l-mers in reads sequenced from a single genome of length G. Let the number of paired-end reads be N (which corresponds to $2N$ read sequences) and read length L. In shotgun sequencing projects, as well as NGS, the reads are randomly distributed across the genome. Since reads may begin at any positions of the genome with equal probability, Lander and Waterman suggested that the left ends of reads follow a Poisson distribution [66], which means that the probability for a read to begin at a given position of the genome is $\alpha = 2N/(G - L + 1)$ and the number of reads starting at each position has a Poisson distribution with parameter α. Consider a substring w_i of length l that begins at the i-th position of the genome. Let $x(w_i)$ be the number of reads that cover this particular l-mer. Since there are $L - l + 1$ possible starting positions for such reads, $x(w_i)$ has

a Poisson distribution with parameter $\lambda = \alpha(L - l + 1)$ (this parameter represents the effective coverage [66, 113]). This analysis assume that the l-mer w_i occurs uniquely in the genome, but in general, an l-mer may occur multiple times. Suppose that an l-mer w has $n(w)$ copies in the genome located at positions $i_1, \ldots, i_{n(w)}$. Then the total number of reads containing w is $x(w) = \sum_{j=1}^{n(w)} x(w_{i_j})$. If we assume that a read covers at most one copy of w, then $x(w_{i_j}), j = 1, ..., n(w)$, are independent and identically distributed. So by the additivity property of the Poisson distribution, the total number of occurrences of w in the reads, $x(w)$, follows a Poisson distribution with parameter $\alpha(L-l+1)n(w)$. In [68], this model is used to find repeat families for a single genome, where a repeat family is a collection of l-mers that have the same number of copies in the genome.

In a metagenome, besides repeats that occur within individual genomes, genomes of different species may share common l-mers. Consider S genomes $g_j, j = 1, ..., S$, and assume that an l-mer w has $n_j(w)$ copies in each genome $g_j, j = 1, ..., S$. Then the number of reads containing w is $x(w) = \sum_{j=1}^{S} \alpha_j(L - l + 1)n_j(w) = \sum_{j=1}^{S} \lambda_j n_j(w)$, where λ_j represents the effective coverage of genome g_j. Since sequencing depth is the same for all genomes, we will refer to it as the abundance. This model is quite difficult to use in practice because we do not know the number of genomes and their repeat structures, common repeats and abundance levels. A simplification of this model is used in AbundanceBin [119], by assuming that for large enough l, most l-mers appear only once in the genomes (note that in AbundanceBin, 20-mers are considered, compared to 12-mers considered in [68]). This allows the authors to estimate the abundance levels of genomes by modeling the abundance levels of the genomes as a mixture of Poisson distributions, where the parameters are the abundance levels of the genomes and their observed values are the counts of the l-mers (*i.e.*, the number of reads containing these l-mers). This approach works well if the abundance levels are sufficiently different. Also,

it is applicable only if the above simplifying assumption holds. In the next section, we will discuss the validity of this assumption in real bacterial genomes and make three important observations about the distribution of l-mers.

2.2.1.2 Observations

Before going into the details of the observations, let us introduce some notations. Consider two different genomes, g_1 and g_2, of lengths G_1 and G_2. Let n_1^{dist} denote the number of distinct l-mers in g_1, n_1^{uniq} the number of l-mers that have only one copy in g_1 (we will call them the unique l-mers in g_1) and n_1^{tot} the total number of l-mers in g_1 (including copies). Obviously $n_1^{tot} = G_1 - l + 1$. The notations for genome g_2 are defined similarly (see Figure 2.1 for an illustration of these notations). Our first observation is the following: (1) Most of the l-mers in a bacterial genome are unique in this genome. To confirm it, we have computed the ratio of unique l-mers to distinct l-mers for all complete bacterial genomes downloaded from NCBI. Figure 2.2 shows the estimated density of this value. We can conclude that fraction of unique l-mers with $l = 20$ is between 96% and 100% for most of complete bacterial genomes.

In order to explain the second observation, let us introduce more notations. Let us consider l-mers from two genomes g_1 and g_2. Denote by n^{dist} the total number of distinct l-mers in both genomes together. We say that an l-mer is unique if it is present only in one genome and, moreover, unique in this genome. Then n^{uniq} denotes the number of unique l-mers in the genomes. Obviously, $n_1^{uniq} + n_2^{uniq} \geq n^{uniq}$, because some l-mers that are unique in one genome may not be unique in both genomes due to common repeats. Our second observation is concerned with the percentage of unique l-mers in a metagenome: (2) Most l-mers are unique in a metagenome if it consists of

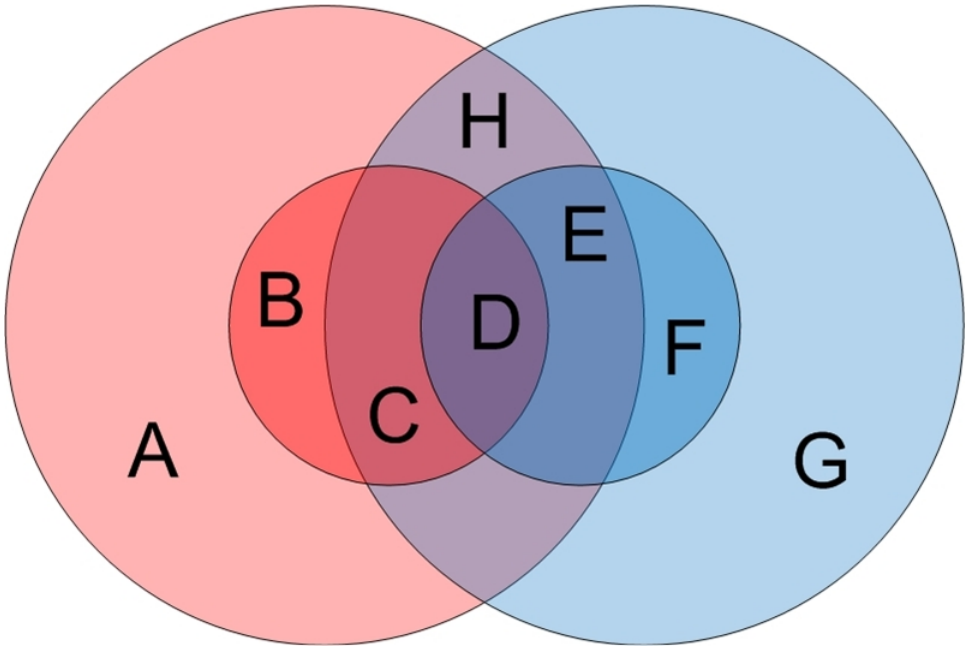

Figure 2.1: Unique and repeated l-mers. A and G: unique l-mers; B and F: individual repeats; C, D, E, and H: common repeats where C and E contain repeats only for one of the genomes, D contains repeats for both genomes, and H contains l-mers that are common to both genomes. Note that $n_1^{dist} = A+B+C+D+E+H$, $n_2^{dist} = C+D+E+F+G+H$, $n_1^{uniq} = A+E+H$, $n_2^{uniq} = G+C+H$, $n^{dist} = A+B+C+D+E+F+G+H$ and $n^{uniq} = A + G$.

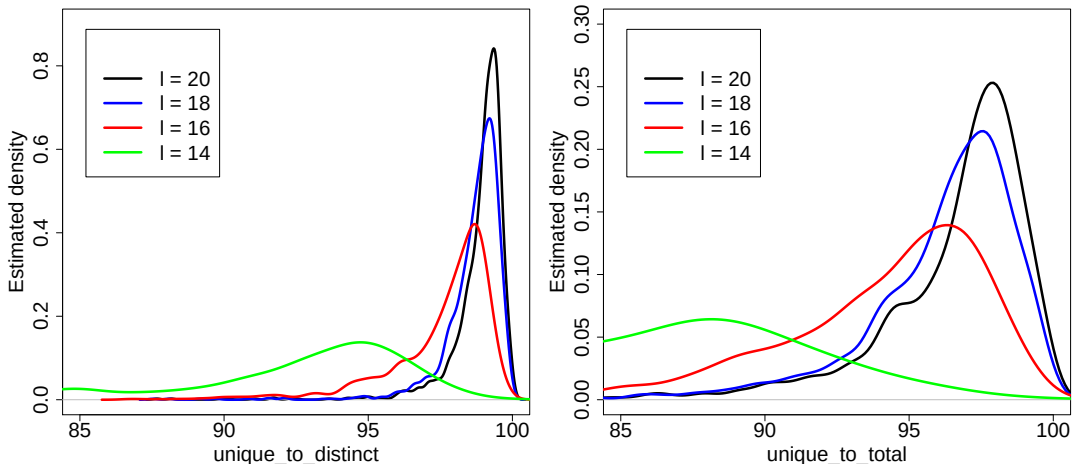

Figure 2.2: The fraction of unique l-mers. Estimated density functions of the fraction of unique l-mers in fully sequenced bacterial genomes for $l = 14, 16, 18, 20$. Left: the ratio of unique l-mers to distinct l-mers. Right: the ratio of unique l-mers to total l-mers.

genomes of species separated by sufficiently large phylogenetic distances. To validate it, we show that the number of l-mers that are unique in an individual genome but are not unique in the metagenome is small. We compute this value for pairs of genomes separated by different taxonomic distances. Figure 2.3 shows the density function of the fraction of l-mers that lost their uniqueness due to common repeats, *i.e.* $1 - n^{uniq}/(n_1^{uniq} + n_2^{uniq})$. We can see that the bigger is the phylogenetic distance, the fewer unique l-mers are lost.

From now on, by "unique l-mers" we will mean l-mers that appear only once in all the genomes. The remaining l-mers are repeats. We will further classify the repeats into two groups: *individual repeats* are l-mers which appear only in one genome (but have several copies) and *common repeats* are l-mers that appear in at least two genomes (see Figure 2.1). Our final observation is: (3) If genomes are separated by sufficient phylogenetic distances (they are at least from different families), then most of the repeats are individual repeats. In addition, the bigger is the phylogenetic distance between genomes, the fewer the common repeats. Figure 2.4 demonstrates the validity of this observation.

Our algorithm is based on these three observations. Since most of the l-mers are unique in a metagenome, we can cluster the unique l-mers by using their common membership in reads so that each cluster contains l-mers from only one genome in the first phase. The second phase of our algorithm uses the property that most of repeats are specific to an individual genome. This allows us to merge clusters using the repeated l-mers in the metagenome. Figure 2.5 illustrates a flowchart of our algorithm. Each main step of the algorithm is explained below.

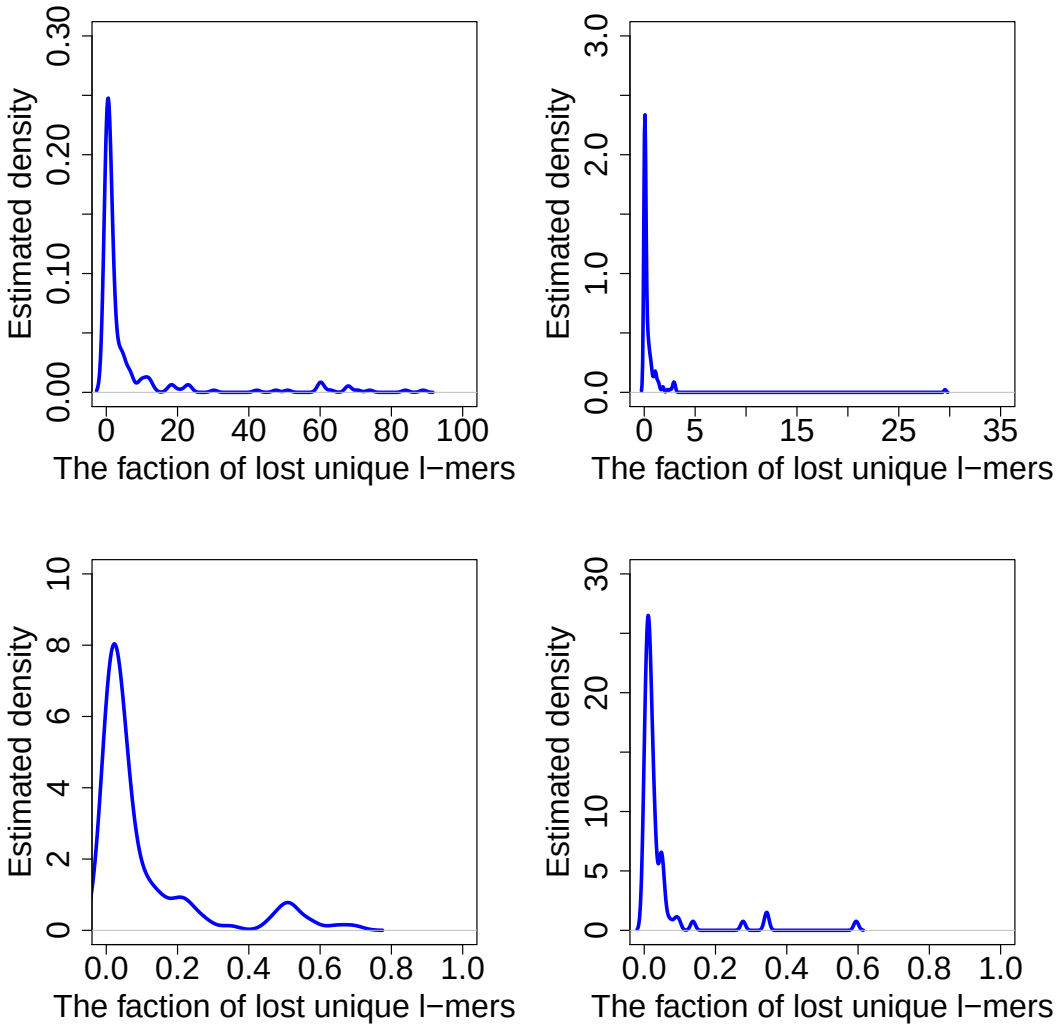

Figure 2.3: The fraction of lost unique l-mers. Estimated density of the ratio of the number of unique l-mers n^{uniq} to the total number of l-mers that are unique in an individual genome, $n_1^{uniq} + n_2^{uniq}$. Top left: pairs of genomes from the same genus but different species. Top right: pairs of genomes from the same family but different genera. Bottom left: pairs of genomes from the same order but different families. Bottom right: pairs of genomes from the same class but different orders.

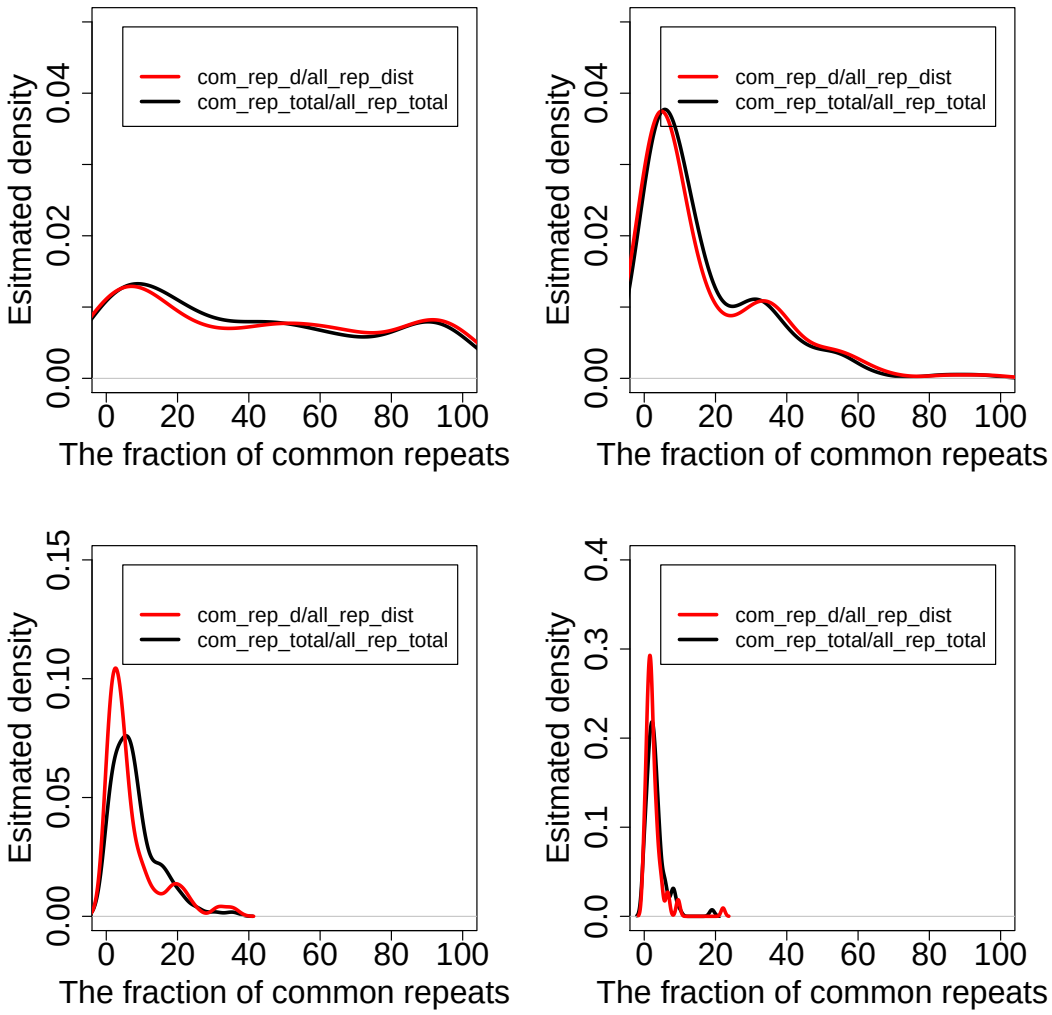

Figure 2.4: The fraction of common repeats. Estimated density function of the ratio of the number of common repeats (or distinct common repeats) to the total number of all distinct repeats (or all repeats, respectively). Top left: pairs of genomes from the same genus but different species. Top right: pairs of genomes from the same family but different genera. Bottom left: pairs of genomes from the same order but different families. Bottom right: pairs of genomes from the same class but different orders.

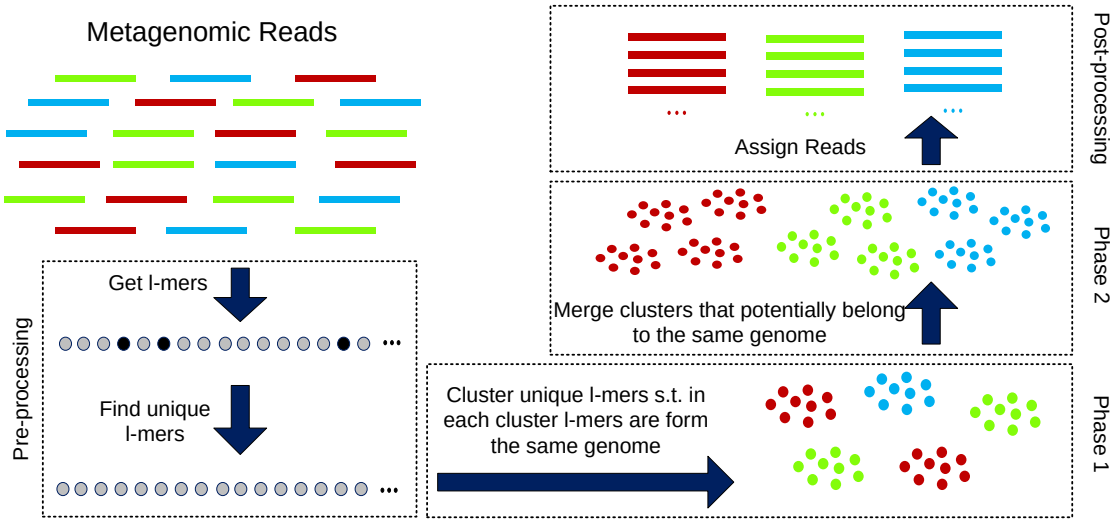

Figure 2.5: Flowchart of the algorithm.

2.2.1.3 Finding Unique l-mers

Before performing the first phase of the algorithm, which clusters the unique l-mers, l-mers have to be separated into unique l-mers and repeats. This is done by choosing a threshold value K for the counts of l-mers so that l-mers with counts less than K are most likely unique and the remaining are most likely repeats. Below, we discuss how to chose K.

First, consider error-free metagenomic reads of genomes with equal abundance levels. Let n be the number of distinct l-mers $w_1, w_2, ..., w_n$ with counts $x(w_1), x(w_2), ..., x(w_n)$. Let $n(i)$ be the number of distinct l-mers with counts i. As we discussed in the previous section, the unique l-mers follow a Poisson distribution and we may approximate the parameter of the Poisson distribution by the most frequent count of any l-mers because most l-mers are supposed to be unique. Then, given the estimated parameter, we can estimate the expected number of l-mers with counts i, $y(i)$. Figure 2.6 shows the count

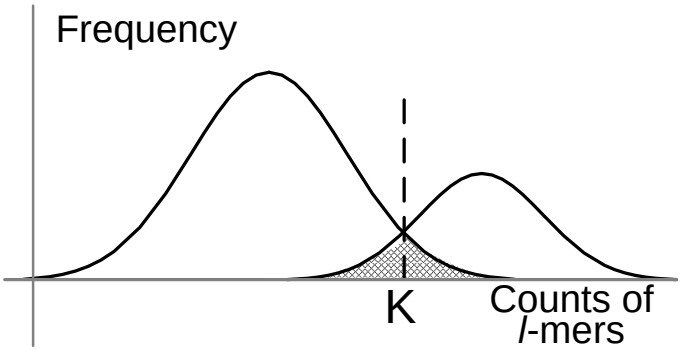

Figure 2.6: Threshold choice for the separation of l-mers from different distributions. K is a threshold to separate l-mers from two distributions.

distributions of unique and non-unique l-mers, where the non-unique l-mers (*i.e.*, repeats) are assumed to be from a mixture Poisson distributions and the shaded area shows the expected rate of misclassified l-mers for the given threshold value K. In the figure, if we choose the threshold higher or lower, more repeats or unique l-mers would be undetected, respectively. As a balance, we would choose the intersection point of the two distributions as shown in Figure 2.6. Although we do not know the distribution of the repeats, we can see that the observed number of l-mers with count K is twice the expected number of unique l-mers with count K, and this ratio increases for count values greater than K. Based on this intuition, we can estimate the value of K. The details are given in Algorithm 1. A similar approach is used to deal with sequencing errors, by finding a threshold value for counts of l-mers that separates unique l-mers and l-mers with errors.

The set U of unique l-mers is then used to construct a graph which can help detect more repeats and will be used to do the clustering. The nodes of the graph G correspond to the elements of U and there is an edge between two nodes if both l-mers are contained in a same read. To remove previously undetected repeats, we use the

Algorithm 1: Clustering of the unique l-mers. Given the graph of unique l-mers $G = (U, E)$, cluster l-mers in U.

```
begin
    T ← FindThreshold()
    i ← 1
    C_i ← Initialize()
    R ← ∅
    Q ← Elements from U \ ⋃_{j=1}^{i} C_i with at least T neighbors in C_i
    while |Q| > 0 do
        if |Q| > 2(L − (l + T) + 1) then
            R ← R ∪ Q
            i ← i + 1
            C_i ← Initialize()
        else
            C_i ← C_i ∪ Q
        Q ← Elements from U \ ⋃_{j=1}^{i} C_i with at least T neighbors in C_i
    N_C ← Number of Clusters()
    for every pair of cluster C_i and C_j, 1 ≤ i, j ≤ N_C do
        W_ij ← Number of paired-end reads connecting the two clusters
    for 1 ≤ i ≤ N_C do
        Top_i ← {j_1, j_2} so that W_ij1 and W_ij2 hold the two largest values
        among all W_ij's.
    for 1 ≤ i ≤ N_C do
        for every j ∈ Top_i do
            if i ∈ Top_j then
                Merge(C_i,C_j)
```

fact that nodes that correspond to truly unique l-mers cannot have more than $2(L - l)$ neighbors.

2.2.1.4 Clustering the Unique l-mers

We use graph G described above to perform the clustering. The purpose is to obtain clusters so that each cluster contains unique l-mers from only one genome. Note that the number of such clusters for each genome can be large. We initialize the first cluster with the l-mers from a randomly selected read and then iteratively find sets of unclustered nodes that are connected to at least T nodes in the current cluster (the

Algorithm 2: Detection of unique and repeated l-mers. Given l-mers $w_i, i = 1, \ldots, n$, and their counts $n(w_i)$, the algorithm classifies l-mers into repeats R and unique l-mers U.

```
begin
    R ← ∅
    U ← ∅
    Compute the number n(i) of l-mers with count i for each i
    λ = arg max_i n(i)
    K ← max(i)
    for i = λ, ..., max(i) do
        y(i) = λ^i e^{-λ} / i!
        if n(i) > 2y(i) then
            K ← i
            break
    for i = 1, ..., n do
        if n(w_i) > K then
            Add w_i to R
        else
            Add w_i to U
```

choice of T is discussed later in the subsection). It is important to note that the number of unique l-mers we can add at each step is limited by $2(L-(l+T)+1)$, since we could add l-mers from both ends of a read. If we need to add more than this many l-mers at some step, it means that we have encountered true repeats that have not been removed and thus we stop expanding the current cluster. We also stop expanding the current cluster if no more nodes could be added. Then we go to the next iteration and construct the next cluster. For each such subsequent iteration, we initialize a new cluster with l-mers from some read that does not correspond to any of the current clusters. A read corresponds to a cluster if at least a half of its l-mers belong to the particular cluster. We create new clusters until there are no more unclustered reads left. At the end of clustering, we obtain a set of disjoint clusters of l-mers. The paired-end information is then used to consolidate the clusters. The details are given in Algorithm 2.

Threshold T (the minimum required number of edges between an unclustered node and the nodes in a cluster so that the node can be added to this cluster) is chosen to make the expected number of coverage gaps less than one. Recall that the effective coverage is $Cov = 2N(L - (l + T) + 1)/(G - L + 1)$ and expected number of gaps is $2Ne^{-Cov}$ [113].

2.2.1.5 Merging Clusters and the Final Clustering of Metagenomic Reads

The goal of the second phase is to merge clusters obtained during the previous phase, based on the repeats and information provided by the paired-end reads. First, for each cluster C_i, we compute the set of repeats R_i that may potentially belong to the same genome as the unique l-mers in C_i. Each R_i consists of two types of l-mers. For each read corresponding to cluster C_i, it may contain some number of repeats. These repeated l-mers are assigned to the set R_i. For each read corresponding to C_i, we also consider its mate (in a paired-end read) and add to R_i all l-mers of the mate that have not been assigned to any clusters. Then for each pair of sets R_i and R_j, we find the intersection of these sets, R_{ij}. Then, we build a weighted graph F, where nodes correspond to clusters C_i and the weight of an edge (i, j) equals the size of set R_{ij}. Finally, the clusters are merged by using the algorithm MCL [105] on the graph F. MCL is an efficient algorithm for clustering sparse weighted graphs and ideal for our situation. To avoid confusion, we will call clusters produced by MCL the *m-clusters*. MCL has a parameter (we denote it by r), corresponding to granularity of clusters. We use an iterative algorithm to find the best parameter so that the m-clusters are big enough (in terms of the number of l-mers contained in each m-cluster) and the total weight of connections between elements within an m-cluster is higher than the total weight of connections between two different m-clusters. Let us call m-clusters that satisfy the

Algorithm 3: Clustering of the unique l-mers. Given the graph of unique l-mers $G = (U, E)$, cluster l-mers in U.

```
begin
    T ← FindThreshold()
    i ← 1
    C_i ← Initialize()
    R ← ∅
    Q ← Elements from U \ ⋃_{j=1}^{i} C_i with at least T neighbors in C_i
    while |Q| > 0 do
        if |Q| > 2(L − (l + T) + 1) then
            R ← R ∪ Q
            i ← i + 1
            C_i ← Initialize()
        else
            C_i ← C_i ∪ Q
        Q ← Elements from U \ ⋃_{j=1}^{i} C_i with at least T neighbors in C_i
    N_C ← Number of Clusters()
    for every pair of cluster C_i and C_j, 1 ≤ i, j ≤ N_C do
        W_ij ← Number of paired-end reads connecting the two clusters
    for 1 ≤ i ≤ N_C do
        Top_i ← {j_1, j_2} so that W_ij1 and W_ij2 hold the two largest values
        among all W_ij's.
    for 1 ≤ i ≤ N_C do
        for every j ∈ Top_i do
            if i ∈ Top_j then
                Merge(C_i,C_j)
```

first property *big*, and a subset of big m-clusters that satisfy the second property (with respect to all other big m-clusters) *valid*. We start with a parameter r which corresponds to a high granularity and evaluate the resultant clusters in terms of size and validity. Based on the evaluation, we either decrease the parameter to have less granularity or choose the current value of r as the parameter for MCL. We obtain final clusters of the unique l-mers by merging clusters that belong to the same m-cluster (see Algorithm 3 in for details).

Now we discuss how to define big and valid m-clusters. The minimum size of a big m-cluster is specified by the user based on the minimum expected length of a genome. Valid m-clusters are chosen from big m-clusters in the following way. Let W_{jj} and W_{ii} be the total weights of the connections within each of the m-clusters j and i, and W_{ij} the total weight of the connections between these two m-clusters. The big m-cluster i is defined to be valid if for every other big m-clusters j, the inequality $\sqrt{\frac{W_{ij}}{W_{ii}W_{jj}}} > 10^{-3}$ holds. The threshold of 10^{-3} is chosen empirically.

In the final step of the algorithm, the reads are assigned to the resultant clusters of unique l-mers. Iterative algorithm is used to assign the reads. At the first step, each reads that correspond to some cluster is assigned to this cluster. During the second step, unassigned reads that have assigned mates are assigned to the same clusters as their mates. In the third step, for each cluster of unique l-mers we add all the l-mers from the reads assigned to the cluster. We iteratively repeat the three steps for the unassigned reads until no more reads can be assigned. If the read correspond to several clusters, we assign it to one of the clusters.

2.2.1.6 Implementation

TOSS was implemented in C++. Its running time and memory requirement depend on the total length of all the genomes present in a metagenomic dataset and on the number of reads. The first phase of the algorithm is the most time and memory consuming. In this phase, a graph of l-mers is constructed and the clustering of unique l-mers is performed. The size of the graph is proportional to the total size of the genomes and 0.5 GB of RAM is required for every million bases of the genomes. In the experiments, we ran the algorithm on a single CPU with 2.8GHz AMD machine and 64GB RAM. Each of the small-scale tests involving 2-4 genomes of total length of

2-6 Mbps was completed within 1-3 hours and required 2-4 GB of RAM. A test on 15 genomes with the total length of 40 Mbps ran for 14 hours and required 20GB of RAM.

2.2.2 Abundance-Based Binning of Metagenomic Reads

In this section, we introduce a novel probabilistic model that can be used for computing the most probable abundance levels of the genomes in a metagenomic dataset and estimating the proportions of the reads corresponding to each abundance group. The problem of binning the reads is then reduced to the problem of determining the parameters of the model and classifying the reads according to the frequencies of l-mers comprising the reads.

2.2.2.1 Definitions and Notations

As we observed in Section 2.2.1.2, most of the l-mers in a bacterial genome occur only once within the genome. However, some l-mers may occur at multiple locations within the genome. Assume that w is an l-mer with n copies in the genome. Due to additivity of the Poisson distribution, the number of reads that cover w, denoted by $x(w)$, has a Poisson distribution with the parameter $n\lambda$. However, due to sequencing errors, the actual count of the l-mer w in the reads, denoted by $y(w)$, may differ from $x(w)$ (see Figure 2.7). Let $x^i(w)$ be the number of reads that cover the l-mer w with i errors in w. Clearly, $x(w) = \sum_i x^i(w)$ and $y(w) = x^0(w) + e_w$, where e_w is the number of times that w occurs in the reads due to errors in other l-mers.

Now, let us consider a metagenomic dataset. Assume that N reads are sequenced from S different genomes. The abundance value of genome g_j is $\lambda_j = N_j(L - l + 1)/(L_{g_j} - L + 1)$, where N_j is the number of reads corresponding to this genome,

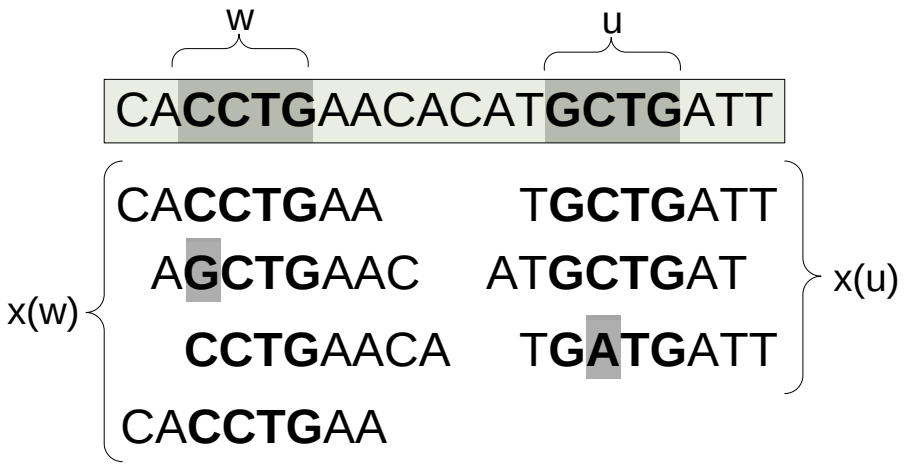

Figure 2.7: Coverage of l-mers and occurrences of l-mers in the reads. The coverage of l-mer w = CCTG is $x(w) = 4$. However, due to an error in one of the reads that cover w, w appears in the reads only 3 times, *i.e.* $y(w) = 3$. For the l-mer u = GCTG, $x(u) = 3$. Observe that even though there is an error in one of the reads that cover u, this l-mer also occurs in a read that covers w due to an error, and thus $y(u) = 3$.

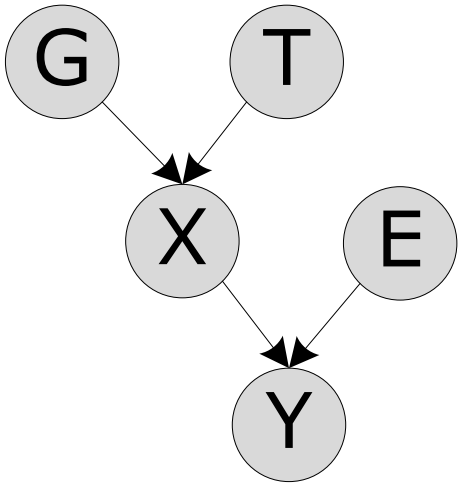

Figure 2.8: The proposed graphical model. Count Y of an l-mer depends on the coverage X and the number of errors E within the l-mer. In turn, the coverage depends on the abundance level G of the genome and the number of occurances T of the l-mer in the genome.

L_{g_j} is the length of the genome g_j, and L is the length of the reads. Let us enumerate all the substrings of length l in all the reads. Clearly, there are $M = N(L - l + 1)$ such substrings. Let us consider the i^{th} substring v_i, $i \in [1, M]$. This substring belongs to the read $r_i \in [1, N]$ which was sequenced from the genome $g_i \in [1, S]$. Let w_i be the original l-mer in the genome g_i corresponding to v_i. Let us assume that w_i has t_i copies in genome g_i. Let e_i be the number of sequencing errors (substitutions) within v_i. Note that e_i equals the Hamming distance between w_i and v_i. Also, let x_i be the number of reads that cover all the copies of w_i in the genome and y_i the number of times that l-mer v_i occurs in the reads.

In Section 2.2.2.2, we model the relationship between the abundance values of genomes, the coverage of l-mers, the number of errors in l-mers, and the counts of l-mers in the reads.

2.2.2.2 A Probabilistic Model for l-mer Frequencies

We define random variables G_i, X_i, Y_i, T_i, and E_i that are associated with the values g_i, x_i, y_i, t_i and e_i, respectively. The variables Y_i are observed by counting the number of occurrences of l-mers in the reads. The other variables cannot be observed directly, so they are hidden. Our goal is to determine the most likely assignment of the l-mers to the genomes. Figure 2.8 illustrates a graphical representation of the model.

Let π_j be a parameter that represents the proportion of the reads that come from the j^{th} genome. Let α_j^n be the fraction of l-mers that occur n times in the j^{th} genome. Let $\alpha_j = (\alpha_j^1, ..., \alpha_j^{n_{max}})$, where n_{max} is the maximum possible number of copies of an l-mer in a genome. For the convenience of notation, we define parameter vectors $\theta_j = (\lambda_j, \pi_j, \alpha_j)$ for all $j \in [1, S]$, and $\theta = (\theta_1, ..., \theta_S)$.

Assuming that the coverage of an l-mer with t copies in a genome g follows a Poisson distribution, the probability that the random variable X_i, associated with the coverage of l-mers in the genome g, takes a particular value c is

$$P(X_i = c|G_i = g, T_i = t, \theta) = \frac{c\,\text{Pois}(t\lambda_g, c)}{\sum_j j\,\text{Pois}(t\lambda_g, j)} = \text{Pois}(t\lambda_g, c-1),$$

where $\text{Pois}(\lambda, k)$ is the probability of a Poisson random variable taking the value k.

The variable Y_i associated with the count of the l-mer v_i in the reads conditionally depends on variables X_i and E_i. If $E_i = e, e > 0$, it means that the corresponding l-mer v_i contains e errors. To model the distribution of counts of l-mers that have e errors, we can borrow the idea from the Balls and Bins problem (`http://www.mathpages.com/home/kmath199.htm`).

Assume that n balls are randomly thrown into m bins. It is known that the expected fraction of bins that get exactly k balls can be approximated by a Poisson distribution with the parameter n/m. Based on this, the probability that an erroneous l-mer has frequency k in the reads is

$$P(Y_i = k|X_i = c, E_i = e, e > 0, \theta) = \frac{kc\,\text{Pois}(c/n_l(e), k)}{\sum_j j\,\text{Pois}(c/n_l(e), j)} = \text{Pois}(c/n_l(e), k-1),$$

where $n_l(e)$ is the number of different possibilities for e errors to occur within an l-mer.

The distribution of the counts of l-mers without errors can be modeled by the binomial distribution. The probability that an error-free l-mer has count k in the reads is

$$P(Y_i = k|X_i = c, E_i = 0) = \frac{k\,\text{Bin}(k, c, p_0)}{\sum_j j\,\text{Bin}(j, c, p_0)} = \frac{k\,\text{Bin}(k, c, p_0)}{cp_0}$$

where $\text{Bin}(j, c, p)$ is the probability that a variable following the binomial distribution takes the value j, and p_0 is the probability that an l-mer does not contain erros.

The above probabilities allow us to compute the probability of a given data point y_i given the values of unobserved variables and the parameter vector θ

$$
\begin{aligned}
&P(Y_i = y_i | G_i = g, X_i = c, E_i = e, T_i = t, \theta) \\
&= P(Y_i = y_i | X_i = c, E_i = e) P(X_i = c | G_i = g, T_i = t, \theta) P(G_i = g, T_i = t, \theta) P(E_i = e) \\
&= \pi_g \alpha_g^t P(Y_i = y_i | X_i = c, E_i = e) P(X_i = c | G_i = g, T_i = t, \theta) P(E_i = e) \qquad (2.1)
\end{aligned}
$$

2.2.2.3 Parameter Estimation

Now, let us consider the log-likelihood of the observed data Y given the parameter vector θ

$$L(Y|\theta) = \sum_i \log P(Y_i = y_i | \theta).$$

Our goal is to find the *maximum likelihood estimate (MLE)* of the parameter θ,

$$\hat{\theta} = \arg\max_\theta L(Y|\theta).$$

To find $\hat{\theta}$, we use the EM algorithm. The E-step requires the computation of the expected value of the log-likelihood function, with respect to the conditional distribution of unobservable variables given the data and current parameter estimates $\theta^{(t)}$:

$$
\begin{aligned}
Q(\theta|\theta^{(n)}) = \sum_i \sum_{g,c,e,t} &P(G_i = g, X_i = c, T_i = t, E_i = e | Y_i = y_i, \theta^{(t)}) \\
&\cdot P(Y_i = y_i, G_i = g, X_i = c, T_i = t, E_i = e | \theta).
\end{aligned}
$$

Here, the posterior probabilities $p_{G,X,E,T|Y,\theta}(g, c, e, t, k, \theta) = P(G_i = g, X_i = c, E_i = e, T_i = t | Y_i = k, \theta)$ of the unobserved data given current parameter estimates $\theta^{(t)}$ can be computed by applying Bayes' rule to Equation 1.

In the M-step, we find the parameter $\theta^{(t+1)}$ that maximizes $Q(\theta|\theta^{(n)})$ with respect to θ

$$\theta^{(n+1)} = \arg\max_{\theta} Q(\theta|\theta^{(n)}). \tag{2.2}$$

The updated parameters are thus

$$\lambda_g^{(n+1)} = \frac{\sum_{c,e,t,k} p_{G,X,E,T|Y,\theta}(g,c,e,t,k,\theta)c}{\sum_{c,e,t,k} p_{G,X,E,T|Y,\theta}(g,c,e,t,k,\theta)t},\ \alpha_g^{t\ (n+1)} = \frac{\sum_{c,e,k} p_{G,X,E,T|Y,\theta}(g,c,e,t,k,\theta)}{\sum_{c,e,j,k} p_{G,X,E,T|Y,\theta}(g,c,e,j,k,\theta)},$$

$$\pi_g{}^{(n+1)} = \frac{\sum_{c,e,k,j} p_{G,X,E,T|Y,\theta}(g,c,e,j,k,\theta)}{\sum_{i,c,e,j,k} p_{G,X,E,T|Y,\theta}(i,c,e,j,k,\theta)}$$

Once we estimate the parameters of the probabilistic model, we can assign l-mers to bins (or genomes) based on the counts of the l-mers in the reads. We assign an l-mer v_i that occurs y_i times in the reads to a bin g with probability $P(G_i = g|Y_i = y_i, \hat{\theta})$. Then, each read is assigned to a bin according to the frequencies of its l-mers in the dataset

$$P(r \in g_j) = \prod_{y_i \in r} P(G_i = g|Y_i = y_i, \hat{\theta}) / \sum_g \prod_{y_i \in r} P(G_i = g|Y_i = y_i, \hat{\theta}).$$

2.2.2.4 Detecting the Number of Bins

The EM algorithm described in Section 2.2.2.3 assumes that the number of bins (genomes) S and the maximum multiplicity of the repeats in the genome (the values that variables T_i may take) are provided. Selecting the best number of clusters is a challenging problem. Here, we propose an iterative algorithm to find the best value for S. We start with one bin and iteratively increase the number of bins until one of the following conditions is reached: (i) one or several bins are split into overlapping bins, making it impossible to assign the reads to the overlapping bins correctly and (ii) one or

Algorithm 4: Deciding the optimal number of bins S and maximum multiplicity of the repeats R. Given observed l-mer frequencies, the algorithm attempts to find the best values for S and R.

```
begin
    V ← ∞
    R, S ← 1, 1
    for r = 1, ..., R_max do
        s ← 1
        θ̂ ← EM(s, r)
        if StopCondition(θ̂) then
            break
        else
            if V(s, r) < V then
                V ← V(s, r)
                R, S ← r, s
        s ← s + 1
    return R, S
```

several bins are too small to represent a whole genome. In order to find the maximum multiplicity of the repeats, denoted by R, we repeat the above procedure for different values of R. For each pair of specific values $S = s$ and $R = r$, we record the distance between the observed and the expected frequencies of l-mers, $V(s,r) = \sum_i |M \cdot P(Y = i|\hat{\theta}_{r,s}) - \sum_{j=1..M} 1_{\{i\}}(y_j)|$. Here $M \cdot P(Y = i|\hat{\theta}_{r,s})$ is the expected number of l-mers with counts i, and $\sum_{j=1..M} 1_{\{i\}}(y_j)$ is the observed number of l-mers with counts i in the reads. Finally, we set S and R to the values s and r for which $V(s,r)$ reaches the minimum. See Algorithm 4 below for the details.

2.2.3 Handling Genomes with Arbitrary Abundance Levels

In Section 2.2.1 we presented TOSS, an algorithm designed to separate reads from genomes with similar abundance levels. In the first phase, TOSS creates clusters of l-mer so that all l-mer in each cluster are likely to originate from the same genome. In the second phase, clusters from the same genome are merged. We would like to extend

TOSS to metagenomic data containing genomes with different abundance levels. If the abundance level difference is not significant, TOSS would still work well. In this case, the number of wrongly determined unique l-mers and repeats in the first phase of the algorithm may slightly increase, but the clustering of l-mers based on their counts using the Poisson mixture model may incur a significantly higher drop of performance. For genomes with significantly different abundance levels, it makes sense to first separate reads according to genome abundance levels. Otherwise, repeats from genomes with lower abundance levels will not be detected, which could lead to a significant increase of granularity in the clustering result produced by the first phase of the above algorithm.

To handle genomes with significantly different abundance levels, TOSS requires a preprocessing step. First, we proposed to use the algorithm AbundanceBin [119] for the initial abundance-based binning of reads. In this case, we run the first phase of TOSS for each of the subsets of reads. For the second phase, we use all the reads to find the connections between clusters so that connections between clusters from genomes with low abundance levels are properly recovered, but MCL is performed on each subset separately.

A key question is what ratios of abundance levels should be considered as significant? This ratio depends on the actual values of abundance levels and also on the sizes of the genomes. Given abundance levels λ_1 and λ_2 ($\lambda_1 < \lambda_2$), genome sizes G_1 and G_2, and a threshold K for classifying l-mers into the two genomes based on count frequencies, we can estimate the expected rate of misclassified l-mers from the count distributions of the l-mers in these two genomes as discussed in Section 2.2.1.1. More specifically, the shaded area in Figure 2.6 represents the expected fraction of misclassification for two distributions. The number of l-mer in this area is $l_2 \sum_{i=1}^{K-1} \frac{\lambda_2^i e^{-\lambda_2}}{i!} + l_1 \sum_{i=K}^{Max} \frac{\lambda_1^i e^{-\lambda_1}}{i!}$. So, we first use AbundanceBin to predict the parameters of count distributions (*i.e.*, the

abundance ratios and genome sizes) and then compute the expected rate of misclassification. If this rate is unacceptable (we used 3% as the threshold in the experiments), it means that the abundance levels are not significantly different and thus we do not run AbundanceBin.

Clearly, the performance of TOSS is significantly affected by the performance of AbundaceBin. Specifically, the inability of AbundaceBin to accurately infer low-coverage genomes may result in bins with low sensitivity. When these bins are provided to TOSS as an input, the performance of TOSS would suffer. Therefore, in the latest version of TOSS, we replaced AbundaceBin with our abundance-based binning algorithm described in Section 2.2.2.

2.3 Experimental Results

We test the performance of out algorithms on simulated and real datasets. Although simulated datasets do not capture all characteristics of real metagenomic data, there are no real benchmark datasets for NGS metagenomic projects and thus they are the main available option.

In Section 2.3.3, we evaluate the performance of TOSS on a variety of synthetic datasets with different numbers of species, phylogenetic distances between species, abundance ratios and sequencing error rates. We modify a well-known genome assembly software, Velvet [125], to make it behave like a genome separation tool and compare our clustering results with those of the modified Velvet. In addition, we compare the performance TOSS with the well-known composition-based method CompostBin [27] on simulated metagenomic Sanger reads. We also apply the algorithm to a real metage-

nomic dataset obtained from gut bacteriocytes of the glassy-winged sharpshooter and achieve results consistent with the original study [117].

In Section 2.3.4 we evaluate performance of our abundance-based binning algorithm on simulated and real datasets and compare the results with AbundanceBin.

In Section 2.3.5, we evaluate our integrated binning algorithm. We test the improved TOSS on simulated NGS datasets and compare the results with those of TOSS that uses AbundanceBin as a preprocessor. Eventually, we compare the performance of the improved TOSS with two very recent binning tools MetaCluster 4.0 and MetaCluster 5.0 on simulated NGS data.

2.3.1 Simulated Data Sets

We use MetaSim [91] to simulate paired-end Illumina reads for various bacterial genomes to form metagenomic datasets. MetaSim is a software for generating metagenomic datasets with controllable parameters, such as the abundance level of each genome, read length, sequencing error rate and distribution of errors. Thus, it can be used to simulate different sequencing technologies and generate reads from available completely sequenced genomes (for example, those in the NCBI database). In our experiments, paired-end reads of length 80 bps are considered, with the mean insert size 500 bps and deviation 20 bps. The sequencing error model is set according to the error profile of 80 bps Illumina reads.

2.3.2 Performance Evaluation

To evaluate the results of clustering, there are a number of factors that should be considered. First of all, we would like most of the reads from each genome to be located in one cluster. In other words, each genome should correspond to a unique

cluster that contains most of its reads. We say that a genome has been *broken* if there is no cluster that contains more than a half of all its reads. It may happen that several genomes correspond to the same cluster. In this case, we assign the cluster to all the genomes, and say that the genomes are not separated. We will measure the performance of the algorithms in terms of pairwise separability. For example, if a dataset contains 5 genomes, where 3 of them are located in one cluster, and each of the other two are located in its own cluster, then in the pairwise evaluation, we consider the separability of all 10 pairs of genomes. Since 3 pairs of genomes are not separated while the other 7 are separated, the separability rate is 70%. During the separability analysis, we remove broken genomes from consideration. Besides separability, we are interested in the precision and sensitivity of our algorithm on the separated genomes. Since we assign a genome to the cluster that has most of its reads, it is also interesting to know how many of its reads are wrongly assigned to other clusters. We call this *sensitivity*. One way to estimate sensitivity is to compute how many reads are correctly assigned to each cluster and divide it by the total number of reads that should be in this cluster. Here, true positives are the reads from all genomes located in this cluster. However, consider the case when we have two genomes in a cluster, of lengths 1 Mbps and 5 Mbps respectively. Then, even if sensitivity is very low for the first genome, the overall sensitivity (for all genomes in the cluster) will not be significantly affected. Another way to normalize sensitivity is by computing sensitivity for each genome in the cluster separately and then to find the average of these sensitivities. We use the second approach. To compute *precision* of a cluster, we find the ratio of the reads that are wrongly assigned to the cluster to the total number of reads in the cluster.

To summarize the results for a set of experiments, we compute separability based on the total number of pairs of genomes in all the experiments. For the precision and sensitivity, we take the average values for all the clusters from all the experiments.

2.3.3 Performance of TOSS

2.3.3.1 Simulated Data

The first experiment is designed to test the performance of TOSS on a large number of datasets of varying phylogenetic distances. For this experiment, we create 182 synthetic datasets of 4 categories. Each dataset of the first category contains genomes from the same genus but different species. Datasets in the second category consist of genomes from the same family but different genera, datasets in the third category involve genomes from the same order but different families, and datasets in the fourth category involve genomes from the same class but different orders. Genomes in each test are randomly chosen according to a category of phylogenetic distances and assumed to have the same abundance levels. The number of genomes in the datasets varies from 2 to 10 and depends on the number of available complete sequences for each taxonomic group and on the level of the group. Tests on genomes from the same genus typically involve 2 to 4 genomes since such genomes are similar to each other and hard to separate, while tests on genomes from the same class may involve up to 10 genomes. Totally, we have 79 experiments concerning a genus, 66 concerning a family, 29 concerning an order, and 8 concerning a class. These datasets involve 515 complete genomes from the NCBI. The number of reads for each experiment is adjusted to produce sufficient coverage depth (ranging between 15 and 30).

We also performed some small-scale experiments to test the performance on genomes with different abundance levels and on reads with sequencing errors. For each of the experiments, we choose 10 random sets of genomes from the 182 datasets. For each set of genomes, two metagenomic dataset are simulated, one with abundance ratio 1:2 and and the second with the error model but abundance ratio 1:1. Finally, we test the performance of the combination of TOSS and AbundanceBin on a dataset of 4 genomes with abundances 1:1:4:4.

2.3.3.2 Comparison with Modified Velvet

Due to the lack of methods for separating short NGS reads into genomes, we modify a well-known genome assembler, Velvet [125], so it behaves like a genome separator. Genome assemblers such as Velvet often work with metagenomic data and produce contigs that may actually correspond to sections of individual genomes. Hence, we run Velvet to obtain a set of contigs and use each contig to define a cluster of l-mers. This is equivalent to the first phase of TOSS. The only difference is that all l-mers (instead of unique l-mers) are clustered. For each read contained in a cluster, we add the l-mers in the mate of the read to the cluster, and then construct a weighted graph whose nodes represent clusters and edges are weighted by the number of common l-mers shared by the clusters connected by each edge. Finally, we apply the merging algorithm to the constructed graph. Based on a series of experiments with the Velvet parameters, we chose l-mer length as 31, which results in the highest N50 in most of the experiments. We also set a very low coverage cutoff so that Velvet can deal with genomes with different abundance levels without filtering out low coverage contigs.

2.3.3.3 Experiments on Genomes Separated by Different Phylogenetic Distances

Our experimental results on metagenomic datasets containing genomes with different phylogenetic distances are summarized in Table 2.1. For genomes from the same genus, separability rate is 45%. It increases to 77% for genomes from the same family and more than 97% for higher level taxonomic categories. For separated pairs of genomes, sensitivity of TOSS increases from 90% for genomes from the same genus to 95% for genomes from the same order. The range of precision is from 95% to 98%. These results are consistent with our expectation for correlation between separability and phylogenetic distance. Figure 2.9 shows the estimated density functions of the fraction of common repeats for separated and unseparated pairs of genomes at the genus and family levels. It confirms our conjecture that the higher is the faction of common repeats, the harder is separation and the worse is the accuracy of our method. Table 2.1 also shows the performance of the Velvet-based approach. Its separability rate is lower than that of TOSS (32% for genus, 62% for family and 91% for order). Its sensitivity and precision numbers are also worse than those of TOSS at the genus and family levels but become slightly higher at the order level.

2.3.3.4 Handling Sequencing Errors

Our approach for handling sequencing errors is very simple: we filter out l-mers with counts lower than a certain threshold, since these infrequent l-mers are likely to contain errors. However, there is a simple intuition behind it. We can aggressively remove potential errors without attempting to correct them or being afraid to lose im-

Table 2.1: Performance of TOSS and the Velvet-based approach on pairs of genomes with different phylogenetic distances.

		# of genomes	# of pairs	Broken	Separated	Sensitivity		Precision	
						mean	stdv	mean	stdv
Species	TOSS	229	184	3	83	90.38	8.71	95.55	5.96
	Velvet	229	156	22	51	84.11	12.70	92.25	7.88
Genus	TOSS	171	147	2	113	93.40	9.39	97.07	5.52
	Velvet	171	123	15	77	89.96	12.97	94.95	8.45
Family	TOSS	80	79	2	75	94.98	6.56	97.46	5.86
	Velvet	80	78	2	71	91.90	8.90	97.25	4.16
Order	TOSS	35	71	0	70	95.14	4.87	97.79	2.49
	Velvet	35	71	0	69	95.79	4.17	98.77	1.69

portant information, assuming that the genomes are sufficiently covered by the reads. Note that we could be more aggressive than genome assemblers in throwing out infrequent l-mers here because (i) when the genomes are sufficiently covered, the filtration will not lead to many more gaps, and (ii) we are less concerned with the fragmentation of genomes.

In Table 2.2, we summarize our experimental results on pairs of genomes with and without sequencing errors. We can see that TOSS is able to separate more pairs of genomes when the reads are error-free. However, when broken genomes are discounted, TOSS actually achieves a slightly higher sensitivity and precision on data with errors. The Velvet-based method has a slightly worse performance, it separates fewer pairs of genomes and achieves a lower sensitivity and precision on both data with and without errors.

2.3.3.5 The Issue of Abundance Levels

In this section, we analyze the ability of TOSS to separate genomes with different abundance levels. First, we test TOSS (without any modification) on pairs of genomes with abundance ratio 1:2 and compare the results with those on the same set

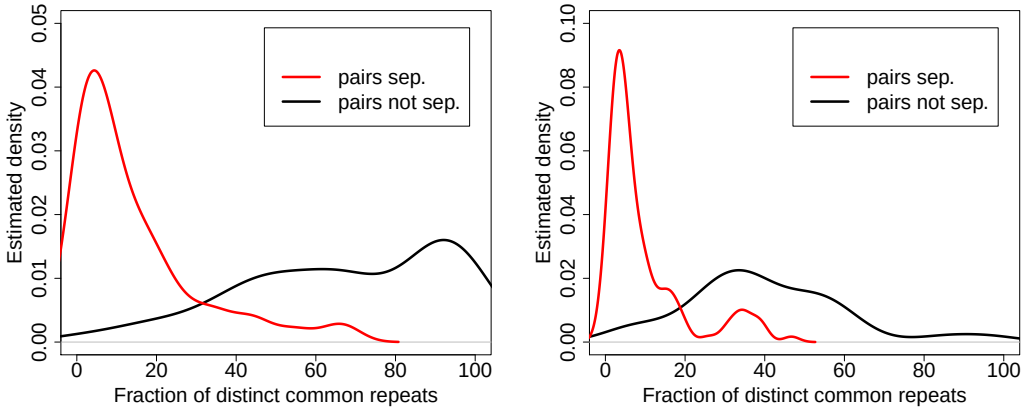

Figure 2.9: The fraction of distinct common repeats for separated and unseparated genomes. Estimated density function of the ratio of the number of distinct common repeats to the number of distinct repeats. Left: pair of genomes from the same genus but different species. Right: pairs of genomes from the same family but different genera.

of pairs of genomes but with identical abundance levels. The results are summarized in the Table 2.3. We can see that sensitivity slightly drops on genomes with different abundance levels, but precision actually improves a little. On the other hand, separability of the Velvet-based method drops significantly.

We also test the performance of a combination of AbundanceBin and TOSS on a set of four genomes with abundance levels $(1 : 1 : 4 : 4)$ and compare the result with that of TOSS on the same set of genomes with identical abundance levels. The results are summarized in Table 2.4. As we can see, the result on data with varying abundance levels is actually better. Sensitivity and precision increase from 92% and 93% on data with identical abundance levels to 97% and 99% on data with varying abundance levels. In order to explain this (somewhat counter-intuitive) phenomenon, we analyzed intermediate results, and found that two of the six pairs of genomes, (1,3) and (2,4), have high percentages of common repeats. These common repeats negatively

Table 2.2: Performance of TOSS on data with and without sequencing errors.

		# of genomes	# of pairs	Broken	Separated	Sensitivity		Precision	
						mean	stdv	mean	stdv
Species	TOSS	229	184	3	83	90.38	8.71	95.55	5.96
	Velvet	229	156	22	51	84.11	12.70	92.25	7.88
Genus	TOSS	171	147	2	113	93.40	9.39	97.07	5.52
	Velvet	171	123	15	77	89.96	12.97	94.95	8.45
Family	TOSS	80	79	2	75	94.98	6.56	97.46	5.86
	Velvet	80	78	2	71	91.90	8.90	97.25	4.16
Order	TOSS	35	71	0	70	95.14	4.87	97.79	2.49
	Velvet	35	71	0	69	95.79	4.17	98.77	1.69

Table 2.3: Performance on synthetic datasets with abundance ratio 1:2.

		# of genomes	# of pairs	Broken	Separated	Sensitivity	Precision
TOSS	DiffAbund	24	18	0	17	91.00	98.25
	IdentAbund	24	18	0	18	93.48	96.08
Velvet	DiffAbund	24	16	1	13	91.88	97.48
	IdentAbund	24	18	0	17	88.24	95.06

affected the result on data with identical abundance levels. However, they did not cause any trouble for the test on data with varying abundance levels since for both pairs, reads from different genomes were separated by AbundanceBin early on due to the difference in their abundance levels. On the other hand, the precision of the Velvet-based method drops significantly.

2.3.3.6 Comparison with CompostBin

In this section, we compare the performance of TOSS with a composition-based binning algorithm, CompostBin [27]. Note that composition-based methods require sufficiently long reads while TOSS is designed to separate short NGS reads. On the other hand, our method requires a high coverage depth. To compare the performance with CompostBin, we use the simulated paired-end Sanger reads of length 1000 bps provided

Table 2.4: Performance on synthetic datasets with abundance ratio (1:1:4:4).

		# of genomes	# of pairs	Broken	Separated	Sensitivity	Precision
TOSS	DiffAbund	4	6	0	6	97.42	99.81
	IdentAbund	4	6	0	6	92.10	93.81
Velvet	DiffAbund	4	6	0	5	91.41	85.24
	IdentAbund	4	6	0	6	90.44	92.65

in [27]. We slightly adapt our method to handle longer reads and lower coverage. In particular, we use a higher threshold in the prediction of unique l-mers. Also, we cut the Sanger reads into fragments of length 80 bps before constructing the graph of unique l-mers in order to minimize memory usage. The linkage information of the fragments belonging to a same read will be recovered and taken advantage of later in the cluster merging phase. Normalized error rates (as defined in [27]) for TOSS and for CompostBin are reported in Table 2.5. Note that in the last three datasets, the average coverage of genomes with lower abundance levels (not shown in the table) is close to 1 and, therefore, is insufficient for our algorithm. In addition, we simulate Illumina reads from the same sets of genomes with a coverage depth between 15 and 30. Normalized error rates for these datasets are shown in the last column of Table 2.5. The highest error rates of TOSS on Sanger and Illumina reads are 4.74% and 4.92% respectively, and are less than 10% for CompostBin. For some of the Sanger datasets, the performance of TOSS is slightly worse compared to CompostBin and for the others, it is slightly better. The performance of TOSS on the corresponding Illumina datasets is better in most of the cases. Clearly, the higher coverage depths in Illumina datasets helped. A high coverage depth is essential for the accurate prediction of unique and repeated l-mers in the preprocessing phase of TOSS.

Table 2.5: Comparison with CompostBin on the datasets described in [27]. Note that some datasets involve genomes separated at several different taxonomic levels.

Species	Ratio	Phylogenetic Distance	CompostBin's Error	TOSS (Sanger) Error	TOSS (Illumina) Error
Bacillus halodurans & *Bacillus subtilis*	1:1	Species	6.48	1.05	1.38
Gluconobacter oxydans& *Granulibacter bethesdensis*	1:1	Genus	3.39	4.72	4.92
Escherichia coli & *Yersinia pestis*	1:1	Genus	10.00	3.11	2.58
Methanocaldococcus jannaschii & *Methanococcus maripaludis*	1:1	Family	0.51	0.22	0.01
Pyrobaculum aerophilum & *Thermofilum pendens*	1:1	Family	0.28	1.05	0.01
Gluconobacter oxydans & *Rhodospirillum rubrum*	1:1	Order	0.98	4.74	0.01
Gluconobacter oxydans, *Granulibacter bethesdensis* & *Nitrobacter hamburgensis*	1:1:8	Family and Order	7.7	-	6.45
Escherichia coli, *Pseudomonas putida* & *Bacillus anthracis*	1:1:8	Order and Phylum	1.96	-	0.15
Escherichia coli, *Pseudomonas putida*, *Thermofilum pendens*, *Pyrobaculum aerophilum*, *Bacillus anthracis* & *Bacillus subtilis*	1:1: 1:1: 2:14	Species, Order, Family, Phylum, and Kingdom	4.52	-	0.80

2.3.3.7 Performance on a Real Dataset

A metagenomic dataset obtained from gut bacteriocytes of the glassy-winged sharpshooter, *Homalodisca coagulata*, is known to consist of (Sanger) reads from *Baumannia cicadellinicola*, *Sulcia muelleri* and some miscellaneous unclassified reads [117] and studied in [27]. We apply TOSS, adapted to handle Sanger reads as discussed in the previous section, to the dataset. As in [27], we only measure our ability to separate the reads from *Baumannia cicadellinicola* and *Sulcia muelleri*. The sensitivity of the classification achieved by TOSS is 92.21% and the normalized error rate is 1.59%, which is lower than the normalized error rate of 9.04% achieved by CompostBin as reported in [27].

2.3.4 Performance of the Abundance-Based Binning Algorithm

2.3.4.1 Performance on a Simulated Data

We compare the performance of our abundance-based binning algorithm against AbundanceBin. In this test, we are mainly concerned with the ability of both algorithms to separate reads from genomes with different abundance levels. The detailed datasets and performance of the two algorithms are summarized in Table 2.6. On most of the datasets, the sensitivity and precision of our method were better than those of AbundanceBin by 4-10%. In tests S4 and S7, AbundanceBin failed to separate the two genomes totally. In test S6, AbundanceBin could identify only 2 bins, while combining the reads from two genomes into one bin. Our method was more ambitious and separated all three genomes at the cost of lowered precision and sensitivity. However, when we set the number of bins to two for the dataset in test S6, our algorithm was able

to achieve a high sensitivity and precision above 95%, compared to 81% and 88% for AbundanceBin.

2.3.4.2 Performance on a Real Dataset

We test the performance of our abundance-based binning method on a dataset obtained from the acid mine drainage [104]. This dataset has been well studied and is known to contain five dominant genomes. The two most abundant species belong to *Leptospirillum* group II and *Ferroplasma* group II. The three species with a lower abundance levels belong to *Leptospirillum* group III, *Ferroplasma* group I and *Sulfobacill.* The dataset consists of approximately 120K Sanger reads. Only 56% percent of the reads can be mapped to the reference sequences of the five dominant genomes. We apply both our algorithm and AbundanceBin to the unfiltered dataset. Then we BLAST the reads of each bin against reference sequences of the five organisms. We measure the ability of the algorithms to separate reads from the two main abundance groups. Although both algorithms could correctly identify the two bins, our algorithm slightly outperforms AbundanceBin in terms of precision and sensitivity. Our method achieves 82% sensitivity and 81% precision, while the corresponding values are 78% and 79% for AbundanceBin. Note that due to the overlap of the bins, it would be very difficult to separate the reads with much better sensitivity and precision based on l-mer frequencies only.

2.3.5 Performance of the Improved TOSS

We compare the performance with the version of TOSS that employs AbundanceBin as a preprocessor. Also, we make a comparison with the most recent metagenomic NGS binning tools MetaCluster 4.0 and MetaCluster 5.0. The results of the

comparison are summarized in Table 2.6. Note that here we only measure the ability of the algorithms to separate high-abundance genomes (with abundance levels ≥ 7, as done in [109]). The improved TOSS obviously outperforms the version of TOSS that relies on AbundanceBin. Compared to the MetaCluster tools, our algorithm often achieves the highest sensitivity and breaks fewer genomes.

2.4 Conclusion

While the NGS sequencing technologies are very promising for metagenomic projects due to their great sequencing depths and low costs, they also present new challenges in the analysis of metagenomic data because of their short read lengths. In this chapter, we developed an algorithm for separating short paired-end NGS reads from different bacterial genomes of similar abundance levels and then combined the algorithm with AbundanceBin [119] to handle arbitrary abundance ratios. We have shown that our algorithm is able to separate genomes when the number of common repeats is small compared to the number of genome-specific repeats. Since the fraction of common repeats in genomes correlates with the phylogenetic distance between the genomes, it is hard to separate genomes of closely related species. However, for the genomes that are separated by sufficient phylogenetic distance, they share few l-mers and can be separated with high precision and sensitivity.

We also introduced a novel probabilistic model for counting l-mer frequencies in a metagenomic dataset. The model allows us to identify the most probable abundance levels of the genomes in a metagenomic sample accurately and estimate the proportions of reads from corresponding genomes. We have shown that our model can serve as a useful preprocessing tool for further metagenomic analysis.

Table 2.6: Comparison with AbundanceBin on simulated datasets. The bold numbers indicate improved sensitivity and precision. The numbers in parentheses are normalized sensitivity and precision.

ID	# genomes	Coverage	Length Mbp	Ours			AbundanceBin		
				Sens.	Prec.	Sep.	Sens.	Prec.	Sep.
S1	2	5 10	2.0 1.9	0.80 **(0.84)**	**0.84 (0.84)**	1	0.80 (0.75)	0.77 (0.76)	1
S2	3	5 5 11	2.7 2.6 3.0	**0.89 (0.89)**	**0.89 (0.89)**	1	0.86 (0.85)	0.86 (0.85)	1
S3	2	5 9	0.6 0.6	**0.79 (0.81)**	**0.78 (0.80)**	1	0.74 (0.69)	0.74 (0.69)	1
S4	2	4 8	4.4 5.2	**0.73 (0.82)**	**0.81 (0.81)**	**1**	-	-	0
S5	3	3 3 8	5.7 4.4 6.0	0.87 **(0.93)**	**0.88 (0.93)**	1	**0.91** (0.89)	0.80 (0.89)	1
S6	3	3 8 15	4.6 4.1 4.7	0.75 (0.83)	0.83 (0.82)	**1**	**0.81 (0.84)**	**0.88 (0.84)**	0.66
S7	6	2,2 2,6 6,6	1.5,1.8 2.0,1.7 1.8,2.0	**0.86 (0.75)**	**0.85 (0.84)**	**1**	-	-	0

Table 2.7: Performance of the improved TOSS and comparison with the previous TOSS, MetaCluster 4.0 and MetaCluster 5.0. The bold numbers indicate the best performance among all four methods.

ID	# gen.	Covr.	Ours+TOSS				ABin+TOSS				MC 4.0				MC 5.0			
			Sens.	Prec.	Sep.	Brok.	Sens.	Prec.	Sep.	Brok.	Sens.	Prec.	Sep.	Brok.	Sens.	Prec.	Sep.	Brok.
T1	4	4,4, 10,10	**1.0**	**0.84**	**1.0**	**0**	-	-	0	0	-	-	-	1	0.75	0.69	**1.0**	**0**
T2	3	4,10,10	**1.0**	0.96	**1.0**	**0**	0.62	0.72	**1.0**	**0**	0.91	**1.0**	**1.0**	**0**	0.79	**1.0**	**1.0**	**0**
T3	3	4,12,12	**1.0**	**1.0**	**1.0**	**0**	**1.0**	0.99	**1.0**	**0**	0.97	0.96	**1.0**	**0**	0.82	**1.0**	**1.0**	**0**
T4	4	7,7,13,13	0.86	0.82	0.83	**0**	0.76	0.76	0.67	**0**	**0.89**	1.0	**1.0**	2	0.84	**1.0**	**1.0**	**0**
T5	10	1,1,1,2, 2,2,1.5, 1.5,10,10	**1.0**	0.92	0.83	**0**	-	-	0	0	0.78	**0.97**	**1.0**	**0**	-	-	-	4
T6	10	1.5,1.5,1.5, 1.5,1.5,1.5, 9,9,9,9	0.91	0.87	**1.0**	**0**	-	-	0	2	0.80	0.96	**0**	0	0.74	**1.0**	1.0	2
T7	18	2,2,2,2, 3,3,3,3, 3,3,3,4, 4,4, 11, 12,12,12	0.87	0.75	0.73	**0**	-	-	0	0	**0.9**	**0.9**	**1**	**0**	0.88	**0.9**	**1**	**0**

Chapter 3

Phylogeny-Based Classification of Microbial Communities

3.1 Introduction

Next-generation sequencing coupled with metagenomics has led to the rapid growth of sequence databases and enabled a new branch of microbiology called comparative metagenomics. Comparative metagenomic analysis studies compositional patterns within and between different environments providing a deep insight into the structure and function of complex microbial communities. Comparative analysis has already led to the discovery of three major enterotypes associated with the human gut microbiota [5] and the differences between lean and obese individuals [103].

Microbial communities can be compared at the levels of sequence composition, taxonomic diversity or functional potential. Taxonomic diversity provides detailed evolutionary information regarding the community composition and therefore has been a focus of many environmental and human studies [30, 71]. To access the taxonomic composition of a microbial community genomic sample, both single marker gene sequencing

and whole community shotgun sequencing are widely used [57]. The 16S ribosomal RNA gene (or 16S rDNA) is a commonly used marker for bacterial identification due to its universal distribution among all bacterial species and a slow rate of sequence evolution. To reduce the dimensionality of large sequence datasets generated by high-throughput sequencing of 16S rDNAs, the reads are clustered into operational taxonomic units (OTUs) [123] that roughly represent taxa at phylogenetic levels defined by a user-defined sequence similarity cutoff. The abundance of each OTU is defined as the number of sequences in the OTU. Representative sequences from each OTU are chosen and used to assign taxonomy to the OTUs and to construct phylogenetic trees. Packages such as QIIME [21] and Mothur [93] provide integrated pipelines for the analysis described above. The whole genome shotgun sequencing approach can also be used to study microbial community composition. It offers a more global view of the community, but may not be deep enough to detect rare species in a sample, and is sensitive to the DNA extraction and sequencing protocols. Even though the two approaches may not always lead to the same conclusions about the community structure [94], they became standard tools in microbial community analysis. In our study, we will focus on samples based on the sequencing of 16S rDNAs although our proposed method can be easily extended to samples based on whole genome sequencing.

For 16S rDNA datasets, each sample is represented as a list of OTUs and their frequencies. We will refer to these lists as *feature vectors*. A feature vector corresponds to one metagenomic sample, and the vector's elements (*features*) characterize OTU frequencies in the sample. The evolutionary relationship between all of the OTUs is captured by a phylogenetic tree. The downstream analysis may involve the identification of compositional patterns across samples from similar environments as well as discriminatory features between different communities, associations between human

bacterial communities and disease phenotypes, prediction of unknown labels for new samples, *etc.* These tasks require the development of new supervised learning techniques that would take into consideration challenges associated with metagenomic data (see Section 1.3).

The problem of classification of microbial communities is not well-studied, even though classification techniques have been widely employed in the field of bioinformatics, including classification of microarray cancer samples [44], gene expression profiles [6], protein families [124], *etc.* Classification of metagenomes may have useful applications in efficient organization and search in rapidly growing metagenomic (or 16S rDNA) databases, detection of disease phenotypes in clinical samples, and forensic identification.

One of the first applications of supervised learning techniques to comparative metagenomics [121] was the classification of soil and sediment samples according to environment types using Support Vector Machines (SVM) and K-Nearest Neighbors (KNN) algorithms. Recently, the feasibility of applying standard supervised classification techniques to metagenomic data was studied on several benchmark datasets of human microbiota [60]. MetaDistance [69] is the first dedicated algorithm for multi-class classification of human microbiota. The algorithm combines the advantages of instance-based and model-based methods, such as KNN and SVM. The above-mentioned methods proved to be efficient learning techniques, and address some of the challenges associated with the properties of metagenomic data. However, none of the methods have yet taken advantage of the inherent properties of metagenomic data, although phylogenetic information contained in metagenomic samples has proved to be useful in comparative metagenomics. For example, similarity measures that take into account phylogeny, *e.g.* UniFrac [70] and its generalized versions [25], outperform non-phylogenetic distance in their ability to recover natural clusters of metagenomic communities [72]. To incorporate

phylogeny into the similarity measure, UniFrac-based methods calculate the degree to which the input samples share branch length on a phylogenetic tree. Another example of a phylogeny-based similarity measure is the parsimony test [92] which employs Fitch's parsimony algorithm to compute the number of minimal changes along the phylogenetic tree necessary to explain all the labels of the sequences. A recursive phylogenetic distance was defined in Meta-Storms [98] for the purpose of fast indexing of metagenomic databases.

Although phylogeny provides important information about the natural hierarchical grouping of features, it has not yet been adopted in classification algorithms for metagenomic communities. To incorporate the underlying structural information among the features in learning problems, several regularization methods have been proposed. For example, in the problem of tumor class prediction from gene expression measurements, functional groups of genes form the natural structure of data, and the combination of L_1 and L_2 norms was used to encode the groups and variables within the groups [53]. The group lasso penalty coupled with logistic regression was applied for classification problems with feature groupings and proved to be useful in short DNA motif modeling and splice site detection [78]. Composite Absolute Penalty (CAP) [129] extended the grouping approach to deal with overlapping groups and with hierarchical orderings of the input variables that reflect the order in which the variables should be included in the solution. The idea of hierarchical grouping was later employed for the multi-task regression learning problem where a tree encodes relationships between the elements of a multidimensional dependent variable, and a balanced weighting scheme to weight the hierarchically overlapping groups was proposed [58]. Here, we show that the hierarchical grouping ideas [129, 58] can be efficiently applied to the multi-class classi-

fication problem where information about the natural hierarchical grouping of features is available.

In this chapter, we propose a new multi-class classification method for 16S rDNA sequence (or metagenomic) samples that takes advantage of the natural structure of microbial community data encoded by a phylogenetic tree. The leaf nodes represent features corresponding to individual OTUs, and the internal nodes may be considered as super-features. The hierarchical structure captures similarities and differences among the features and allows for the consideration of features at different granularity levels, which may be desirable given that the features do not necessarily correspond to specific taxonomic units and only represent groups of similar sequences. Thereby, some environment-specific patterns may be comprised of features at multiple granularity levels. We propose a multinomial logistic regression model with a tree-guided penalty that incorporates the hierarchy in the feature space, and provide an efficient optimization algorithm to learn the model parameters.

We apply our algorithm to several real datasets of 16S rDNA sequences from the human microbiota. We compare the classification performance of our method with several state-of-the-art learning algorithms, and show that incorporating the natural structure of the microbial data results in a model with a better predictive power. We also perform a comprehensive analysis of the proposed method by applying it to simulated datasets of different complexity. Because the problem of classifying 16S rDNA sequence samples is relatively new, there are no simulated data generators that would generate the data appropriate for our goal. Current comparative studies use simulations where features are generated independently from each other rather than according to a phylogeny. Therefore, we propose a new simulation approach that employs the phylogenetic structure of microbial communities to generate OTU count data and simulate

scenarios where community-specific patterns may be comprised of features at multiple levels of granularity. We show that by taking advantage of the phylogeny, our algorithm has a robust performance with respect to the choice of feature resolution, which corresponds to the selection of a similarity threshold when defining OTUs for the real data.

3.2 Methods

First, we define a classical multinomial logistic regression model associated with our classification problem for 16S rDNA sequence samples. Then, we provide a background on how prior knowledge about the data, such as sparsity and natural grouping, can be effectively incorporated into learning models. Finally, we present a model for the classification problem that takes advantage of the hierarchical groups of features, and describe a fast optimization algorithm.

3.2.1 The Multinomial Logistic Regression Model

Consider a supervised learning problem where K microbial communities correspond to different environments or conditions. Each community (class) is represented by several samples. Let N be the total number of samples present in the dataset. A sample $x^i, i \in \{1, ..., N\}$ is characterized by feature values representing the abundance values of each OTU, $x^i = (x^i_1, x^i_2, ..., x^i_M)$, where M denotes the number of features. Each feature x^i_j of x^i is a continuous random variable. We first consider features as independent variables, but will later add more definitions and extend the model to incorporate the information about the feature space encoded by a phylogenetic tree. Let y^i be a K-dimensional class variable, where each component takes on binary values, so that $y^i_j = 1$ if x^i belongs to the community j, and $y^i_j = 0$ otherwise. Our goal is to model

class labels as a function of the input features, that is, to find a classification rule so that a new 16S rDNA sample x can be assigned to one of the classes. The goodness of the fit is measured by the loss function $L(x, y, \beta)$, where β is a vector of model coefficients. We formulate the problem as a multinomial logistic regression so that the probability of a class label given the feature vector is modeled by the logistic function:

$$P(y_j = 1|x, \beta) = \frac{e^{x\beta_k^T}}{\sum_{k'} e^{x\beta_{k'}^T}},$$

where $\beta_k = (\beta_1^i, \beta_2^i, ..., \beta_M^i)$ is the vector of parameters corresponding to the k-th class.

A new sample is assigned to a class with the highest conditional probability estimate. The model coefficients β are obtained to better fit the observed data, that is, to minimize the loss function defined as a log-likelihood:

$$L(x, y, \beta) = \sum_i \sum_k y_k^i x^i \beta_k - \sum_i \ln \sum_k e^{x^i \beta_k^T}.$$

3.2.2 Bayesian Regularization

To incorporate additional knowledge about the parameters, a Bayesian approach is often used. For example, the Gaussian prior favors parameter values that are close to 0 while the Laplace prior favors sparse solutions. Maximum a posteriori estimates of the model coefficients yield penalty terms in the form of the L_2 and L_1 norms for the Gaussian and Laplace priors, respectively. Finally, the optimal coefficients are obtained as a solution to a joint minimization of the loss function and the penalty function:

$$\arg\min_{\beta} L(x, y, \beta) + \lambda T(\beta),$$

where $T(\beta) = ||\beta||_2$ and $T(\beta) = ||\beta||_1$ for the two cases described above.

3.2.3 Tree-Guided Regularization

In some situations, it may be desirable to consider some features as a group. Let β^{G_i} be a set of coefficients that correspond to the i-th group of features. The group penalty may be defined as a combination of norms for the groups and for coefficients within each group. To incorporate the hierarchical nature of the features, groups may be chosen to hierarchically overlap. Let T be a tree that reflects the hierarchical relationship among the features. Let us consider a node in a tree $v \in V$, and denote β^{G_v} to be a set of coefficients that correspond to features descendant from the node. As suggested in [58], the tree-guided penalty may be defined as follows:

$$T(\beta) = \sum_k \sum_{v \in V} \omega_v ||\beta_k^{G_v}||_2,$$

where the weight ω_v is associated with the node v and reflects a correlation within the group of features descending from v. Since each coefficient β_j^i may correspond to multiple hierarchically overlapping groups of coefficients, it is important to penalize the coefficients equally. In particular, for each particular coefficient β_j^i, the weights of all the groups that contain this coefficient should sum to one. A weighting scheme that assures this property was proposed in [58]:

$$\omega_v = \begin{cases} g_v \prod\limits_{m \in Anc(v)} s_m & \text{if } v \text{ is an internal node,} \\ \prod\limits_{m \in Anc(v)} s_m & \text{otherwise.} \end{cases}$$

Eventually, the solution to our problem is found as the minimum of the following function:

$$-\sum_i \sum_k y_k^i x^i \beta_k + \sum_i \ln \sum_k e^{x^i \beta_k^T} + \lambda \sum_k \sum_{v \in V} \omega_v ||\beta_k^{G_v}||_2.$$

3.2.4 Cyclic Coordiante Descent

Estimation of the model coefficients requires solving a convex optimization problem and therefore an efficient algorithm that scales up well to handle large high-dimensional data is necessary. Newton methods are intractable for high dimensional data, while feature selection methods may not have good statistical foundations as most of them consider features in isolation which is not desirable in our framework. Another challenge is that the penalty term is not differentiable at 0. A cyclic coordinate descent algorithm is a fast and simple algorithm that has been efficiently applied for the binary ridge logistic regression model [128] and extended to handle the non-smooth case of the lasso-penalty [75]. Our implementation for the tree-guided penalty is based on the above algorithms.

The cyclic coordinate descent method minimizes a function by cyclically updating each coefficient while setting other coefficients fixed. The procedure continues until the convergence criterion is met. Each update involves a Newton step. Since Newton's method is based on the quadratic approximation of the objective function, large updates should be avoided when the quadratic approximation is poor. We use an updating rule suggested in the CLG algorithm [128] and outlined in Algorithm 5 [75]. When computing the step size $\delta_{v_{kj}}$, unlike the CLG algorithm, we use the second derivatives directly rather then their upper bounds. For a smooth function, the update $\delta_{v_{kj}}$ would be

Algorithm 5: Coordinate descent algorithm.

```
begin
    for k = 1, ..., K; j = 1, ..., M do
        β_kj ← 0
        δ_kj ← 1
    for t = 1, 2... do
        for j = 1, ..., M do
            for k = 1, ..., K do
                compute δ_{v_kj}
                δ_{β_kj} ← min(max(δ_{v_kj}, −δ_kj), δ_kj)
                β_kj ← β_kj + δ_{β_kj}
                δ_kj ← max(2|δ_{β_kj}|, δ_kj/2)
```

$$\delta_{v_{kj}} = -\frac{\frac{\partial}{\partial \beta_{kj}}(L(x, y, \beta) + \lambda T(\beta))}{\frac{\partial^2}{\partial \beta_{kj}^2}(L(x, y, \beta) + \lambda T(\beta))}.$$

However, the penalty function does not have a derivative at $\beta_{kj} = 0$ because some of the terms in the penalty function contain the absolute value of β_{kj}. This happens because the L_2 norm $||\beta_k^{G_v}||_2$ degenerates to the L_1 norm $||\beta_{kj}||_1$ when either v is a leaf node that corresponds to the j-th feature, or v is an ancestral node of j and all the other coefficients descendant from v are equal to 0. We will denote the set of nodes v that have one of the above properties by V_{kj}. We follow the approach in Madigan *et al.* (2005) to handle the the non-smooth penalty. Specifically, when $\beta_{kj} \neq 0$ we use the following update rule:

$$\delta_{v_{kj}} = -\frac{\frac{\partial}{\partial \beta_{kj}}(L(x, y, \beta) + \lambda D_1(k, j, \beta))}{\frac{\partial^2}{\partial \beta_{kj}^2}(L(x, y, \beta) + \lambda D_2(k, j, \beta))},$$

where

$$D_1(k, j, \beta) = \sum_{\substack{v \in Anc(j) \\ v \notin V_{kj}}} \omega_v \frac{\partial ||\beta_k^{G_v}||_2}{\partial \beta_{kj}} + \sum_{v \in V_{kj}} \omega_v Sign(\beta_{kj}),$$

$$D_2(k, j, \beta) = \sum_{\substack{v \in Anc(j) \\ v \notin V_{kj}}} \omega_v \frac{\partial^2 ||\beta_k^{G_v}||_2}{\partial \beta_{kj}^2}.$$

If updating β_{kj} results in a sign change, β_{kj} is set to 0. If $\beta_{kj} = 0$, we try updating β_{kj} in both directions, and if any of the updates results in a decrease in the objective function $L(x, y, \beta) + \lambda T(\beta)$, we update β_{kj} accordingly.

3.2.5 Implementation Details

The implementation details that improve the computational efficiency include (1) maintaining a list of nonzero data entries for each dimension, (2) storing the dot product of x^i and β_k and updating the result once β_k is updated, (3) storing the sums of the exponentials $\sum_k e^{x^i \beta_k^T}$ and updating when the dot products in the exponents are updated, (4) traversing the tree in the bottom-up manner to locate the sets V_{kj} of ancestors for each β_k^j to quickly find which terms in the penalty function degenerates to the L_1 norms and (5) computing partial sums of the hierarchically overlapping sets of coefficients in a top-down manner.

3.3 Experimental Results

We propose a new simulation framework for the OTU count data generation that incorporates knowledge about phylogenetic relatedness of the species. We use the framework for a comprehensive evaluation of our algorithm's performance under a variety of simulation settings. We compare the performance of our algorithm with the state-of-the-art classification algorithms on simulated and real datasets. The results indicate that our algorithm has a better classification accuracy and is more robust with respect to the granularity level of the features. The performance is evaluated based on

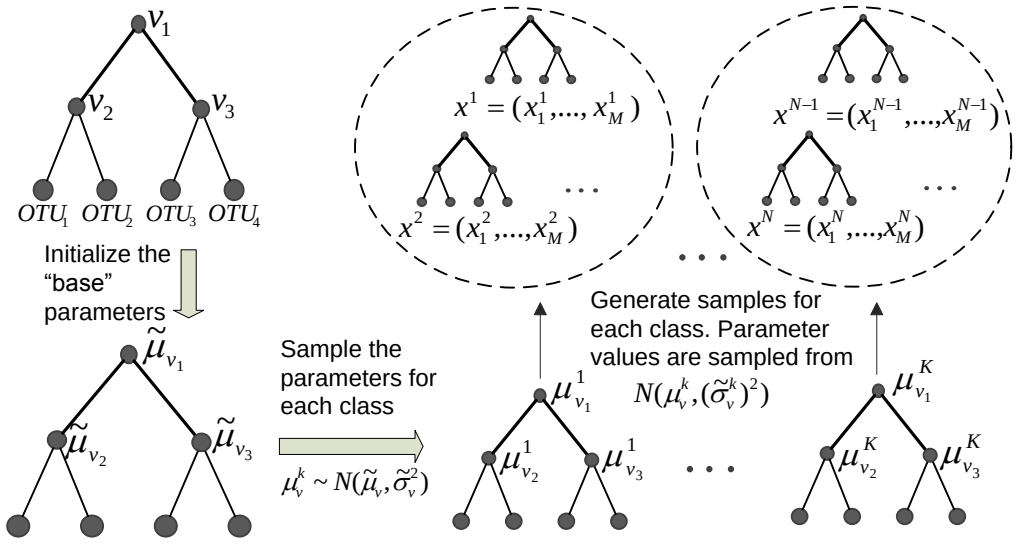

Figure 3.1: The simulation pipeline.

Table 3.1: Relative performance of our algorithm compared to LR, SVM, RF and MetaDistance across 30 simulated datasets for $\tilde{\gamma} = 1$, $\gamma = 1$ and $K = 5$. The bold numbers indicate the best performance among all five methods.

LR		SVM		RF		MD	
mean	std	mean	std	mean	std	mean	std
-4.3	3.9	-12.3	3.7	-10.0	5.8	-5.0	3.3

the classification error rate. We will not attempt to evaluate the algorithms based on their computational efficiency. However, we will comment on the running time when necessary.

3.3.1 Synthetic Framework and Performance Analysis

Simulated data is often used for extensive performance evaluation of algorithms under varying settings for which real data may not be available. Current data simulators that are used in comparative metagenomics simulate counts for each OTU independently not taking into consideration phylogenetic relationship between the species.

For example, to evaluate the performance of MetaStats [115], a statistical method for the comparison of clinical metagenomic samples (represented as counts of individual features such as organisms, genes and functional groups), read counts for each feature were generated according to negative binomial distributions independently from the other features. To evaluate phylogeny-based distance measures, such as UniFrac, microbial communities were represented as circles or ellipses [25] with the overlap patterns between these ellipses expressing similarity between the communities. Species were simulated by sampling points from the interior of the ellipses. In this work, we propose a novel simulation approach to generate OTU count data that takes into account the input phylogeny and provides the flexibility for generating community-specific patterns at multiple granularity levels.

Our starting point is the common phylogenetic tree T that relates OTUs in all the 16S rDNA samples. For the ease of presentation, we will consider binary trees, although the model can be easily generalized to handle any trees. To generate samples for a class k, we traverse the tree systematically, deciding for each internal node v what fraction of species would come from each of the subtrees rooted at the child nodes of v. We associate two parameters with each node v for each class k. Let μ_v^k denote the average proportion of species that correspond to the subtree rooted at the left child node of v in the k-th class, and let $(\sigma_v^k)^2$ denote the variance of this proportion within the class. A new class sample is generated by sampling the proportions of species at each node v according to the normal distributions $N(\mu_v^k, (\sigma_v^k)^2)$. The parameters values μ_v^k are in turn sampled from the normal distribution $N(\tilde{\mu}_v, \tilde{\sigma}_v^2)$, where parameters $\tilde{\sigma}_v^2$ characterizes the variance between the classes, and $\tilde{\mu}_v$ are some "base" values that are initialized randomly. The simulation pipeline is outlined in Figure 3.1.

We control the within- and between-class variances using the parameters $(\sigma_v^k)^2$ and $\tilde{\sigma}_v^2$, respectively. We observed that in real-world datasets both variances tend to increase towards the leaves of the tree (see Figures 3.6 and 3.7). To incorporate such a behavior into our framework, we define coefficients $\tilde{\gamma}$ and γ that describe the overall variances between and within the classes, respectively. We then sample the exact values of $\tilde{\sigma}_v$ and σ_v^k at each tree node v according to $N(0, \tilde{\gamma}d(v))$ and $N(0, \gamma d(v))$, where $d(v)$ is the distance between v and the tree root. Note that the parameters $\tilde{\gamma}$ and γ influence the difficulty of the classification problem, which is proportional to γ and inversely proportional to $\tilde{\gamma}$.

3.3.2 Comparisons

We compare the performance of our proposed method with some of the popular classification techniques such as SVMs, logistic regression (LR) and Random Forests (RFs), and with a dedicated classification algorithm for metagenomic (or 16S rDNA sequence) samples, MetaDistance. SVM [9] is a robust and powerful classification method that has a widespread applications in many fields including computational biology. The idea behind the SVM technique is to find the separating hyperplane in a feature space with the largest margin between the classes. We consider the L_1-regularized version of SVMs due to its ability to handle high-dimensional sparse data. RF [15] is another popular classification algorithm that consists of a collection of tree predictors and makes the overall prediction by the majority voting. The algorithm is well-known for its capability of dealing with small sample sizes, high-dimensional feature space, and complex data. LR [118] has been widely used in statistics for many years, and has recently received much attention in the machine learning community. In particular, its L_1-regularized version is known to have good generalization performance in the presence of many irrel-

evant features. MetaDistance [69] is a classification method for samples of 16S rDNA sequence counts. It is based on simultaneously minimizing the intraclass distance and maximizing the interclass distance by combining instance-based and model-based learning techniques. For the comparison with the SVM and LR classifiers, we used the implementations in the MLPY Python package [1], for the RF classifier, we used scikit-learn Python package [84], and for MetaDistance, we used the Matlab implementation [69]. For all the classifiers, we performed 10-fold cross-validation to find the optimal model parameters and 3-fold (or 5-fold) cross-validations for the performance evaluation on simulated (or real, respectively) datasets.

3.3.3 Performance on Simulated Data

We simulated datasets of 2, 3, 5 and 10 classes with 20 samples for each class. To generate datasets of different complexity, we considered the variance coefficients $\tilde{\gamma}$ and γ of 0.5, 1 and 1.5. A complete binary tree of height 10 was used to guide the simulation. For each parameter set, we generated datasets with different granularity by cutting the tree at different heights to produce feature vectors with different resolutions. The performance of our algorithm for all the parameter sets and resolution levels is shown on Figure 3.2. We observe that the classification task becomes harder when the number of classes is increased. Also, as expected, the classification error rate is low when γ is decreased and $\tilde{\gamma}$ is increased. We also observe that for some parameter settings the best performance is achieved at the highest resolution levels, while for the other settings the best performance may be at some intermediate resolutions. This happens when the between-class variance is dominated by the within-class variance at high resolution levels, leading to an increased overlap between the classes. Using the information about

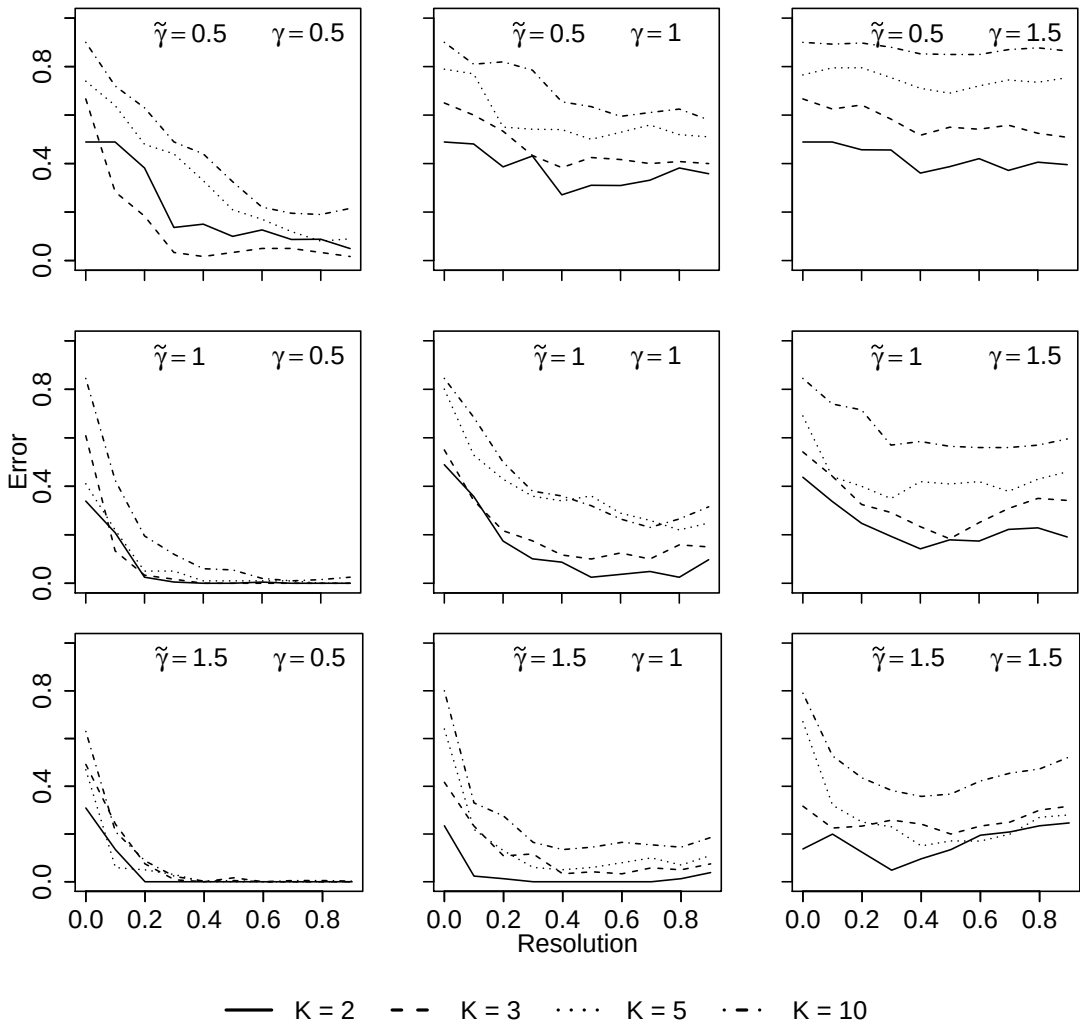

Figure 3.2: Performance for varying number of classes and within- and between-class variances.

the hierarchical grouping of the features makes it possible to take advantage of the lower resolution feature space where the classes may be better separated. Figure 3.2 shows that even for these cases the accuracy at the highest resolution level is still reasonable compared to the accuracy at the optimal resolution level.

We compared our algorithm with the other classifiers under a variety of parameter sets and plotted the results in Figures 3.3 and 3.4. The mean and standard deviations of the relative performance of the algorithms at the highest resolution level

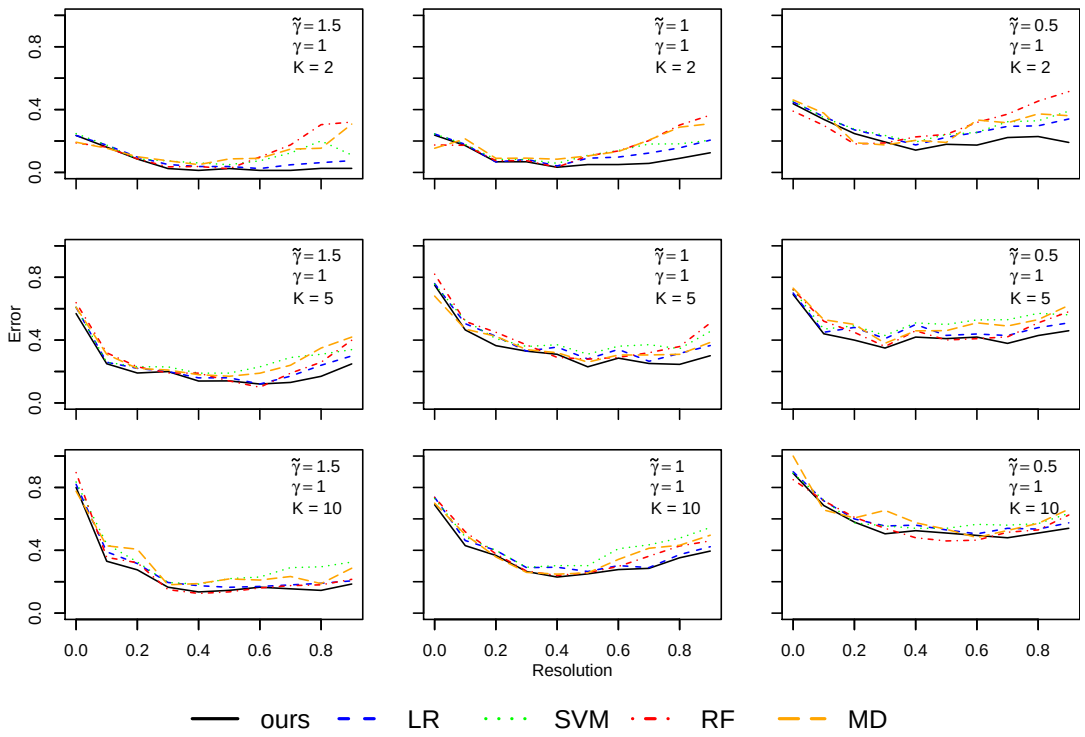

Figure 3.3: Comparison with LR, SVM, RF and MetaDistance on simulated datasets for varying number of classes and between-class variances. The top, middle and bottom plots correspond to datasets with 2, 5 and 10 classes, respectively. The within-class variance $\gamma = 1$. The between-class variance $\tilde{\gamma}$ is 1.5, 1 and 0.5 on the left, middle and right plots.

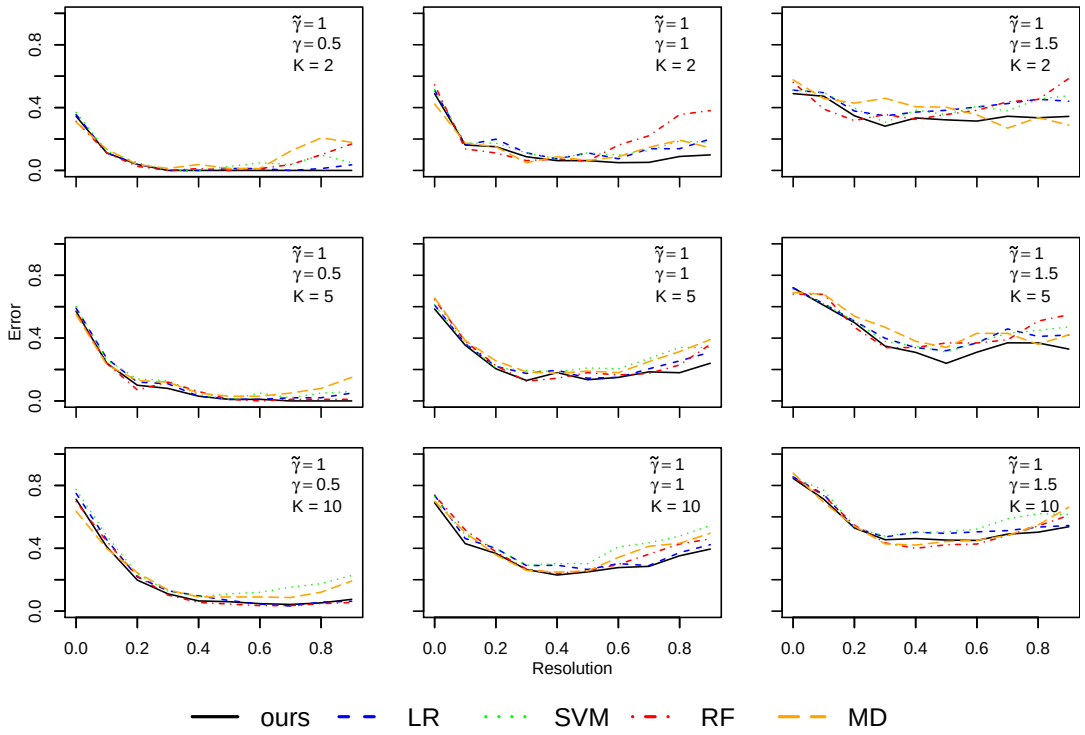

Figure 3.4: Comparison with LR, SVM, RF and MetaDistance on simulated datasets for varying number of classes and within-class variances. The top, middle and bottom plots correspond to datasets with 2, 5 and 10 classes, respectively. The between-class variance $\tilde{\gamma} = 1$. The within-class variance γ is 0.5, 1 and 1.5 on the left, middle and right plots.

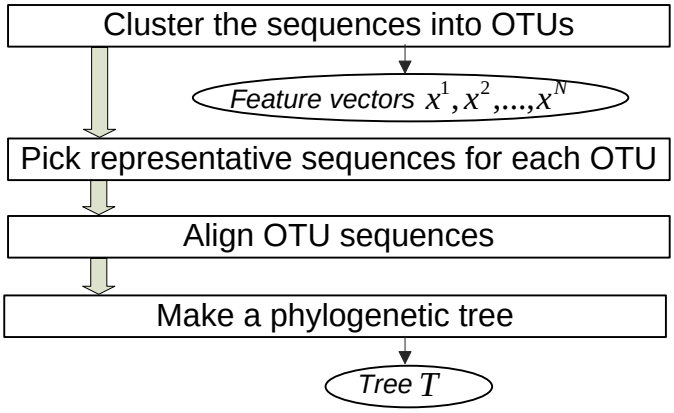

Figure 3.5: The real data pre-processing pipeline. Rectangular boxes show the QIIME steps. Ellipses show the input data for our classification algorithm. First, all the reads are clustered into OTUs based on a user-defined similarity cutoffs using UClust [38]. For each sample, a feature vector of OTU frequencies is constructed. The most abundant sequence in each OTU is picked as the representative sequence. A multiple sequence alignment of the representative sequences is built using PyNAST [20]. Finally, the phylogenetic tree relating the OTUs is constructed from the multiple sequence alignment using FastTree [86].

are shown in Table 3.1. Overall, LR performs reasonably well for the most of the test cases, while SVM has an inferior performance compared to the other algorithms. RFs perform poor for the small number of classes, but improve significantly when the number of classes is increased. In most of the test cases, our algorithm achieves the best performance and is more robust with respect to the resolution level. That is, when the optimal classification accuracy is achieved at some intermediate resolution, the performance at the highest resolution is still reasonably close to the optimal. Figures 3.3 and 3.4 show that the performance difference between the algorithms is especially noticeable at the highest resolution.

Table 3.2: Comparison of the error rates (%) with LR, SVM, RF and MetaDistance classifiers on real datasets.

	Alg.	65	70	75	80	85	90	95	97
D1	Ours	12.9	**10.4**	**8.2**	**7.1**	**6.7**	**5.7**	**5.5**	**5.3**
	SVM	12.9	11.0	10	8.8	7.0	6.9	6.5	6.3
	LR	12.9	10.8	9.6	8.2	7.6	7.1	6.5	6.1
	RF	**12.7**	**10.4**	10.0	9.8	9.4	8.2	7.8	7.6
	MD	14.9	12.2	10.8	8	6.9	6.7	-	-
D2	Ours	29.2	**27.4**	**24.6**	**24.6**	23.9	**21.0**	**19.2**	**16.7**
	SVM	30.3	29.6	26.7	28.9	26.7	26.4	26.7	23.1
	LR	29.9	29.2	27.7	28.5	25.6	26.0	23.9	21.4
	RF	**28.5**	29.2	26.0	25.6	24.5	26.3	26.7	27.8
	MD	30.2	30.2	30.2	26.0	**23.1**	22.1	23.4	20.9
D3	Ours	9.3	10.4	**7.2**	**4.9**	**3.8**	**3.5**	**3.3**	-
	SVM	8.7	10.5	**7.2**	5.7	4.4	4.1	4.5	-
	LR	**8.3**	10.2	7.7	5.4	4.3	4.5	3.8	-
	RF	8.7	**10.0**	7.4	5.7	4.9	5.1	5.1	-
	MD	12.8	10.3	8.7	6.2	-	-	-	-

3.3.4 Performance on Real Data

We used three real datasets of the human microbiota to evaluate our algorithm. All the datasets were taken from the 16S rRNA sequencing studies and pre-processed as shown in Figure 3.3.3 using the QIIME software [21]. Using different OTU similarity cutoffs, we generated feature vectors of different granularity for each of the datasets. The dataset $D1$ is described in [30] and is comprised of samples from six major body areas: external ear, gut, hair, nostril, oral cavity and skin. The second dataset $D2$ contains gut samples from lean, obese and overweight subjects [101]. The third dataset $D3$ is described in the study of microbiota in healthy adults [29] and contains samples from five body habitats: oral, gastrointestinal, urogenital, nasal and skin. Datasets $D1$ and $D3$ represent relatively easy classification problems because they are comprised of microbial communities from different body sites, which are known to be significantly different. On the other hand, $D2$ illustrates an example of a more challenging classification task because the classes correspond to microbial communities from the same body habitat

and thus are very similar. To support the conclusions about the difficulty of the problems represented by the three datasets, we have computed the average within- and between-class variances for each dataset (see Figures 3.6 and 3.7). We observe that, compared to $D1$ and $D3$, dataset $D2$ on average has a lower between-class variance which confirms that the corresponding classification problem is more challenging.

The classification error rates for all the classification algorithms at all the granularity levels are shown in Table 3.2. On average, all the algorithms show a better performance at higher resolution levels which correspond to the higher OTU similarity cutoffs. Our algorithm shows a comparable performance at the lower resolution levels and outperforms the other methods at the higher resolution levels.

We do not evaluate the computational efficiency of the algorithms systematically because they are implemented in different programming languages. Some of the entries are missing in Table 3.2 due to overly long processing time. More specifically, we were unable to pre-process the dataset D3 at 97% similarity cutoff with QIIME on our desktop PC. Therefore, we did not run any of the classification algorithms on the dataset at 97% similarity cutoff. MetaDistance was two orders of magnitudes slower than the other algorithms. For example, the training step for the dataset $D1$ with 90% OTU similarity cutoff for just one set of parameters took approximately 1 second for LR and SVM, 3 seconds for our algorithm, 5 seconds for RF and 12 minutes for MetaDistance. Therefore, running cross-validation for multiple sets of parameters would take days or weeks to compute on some datasets for MetaDistance.

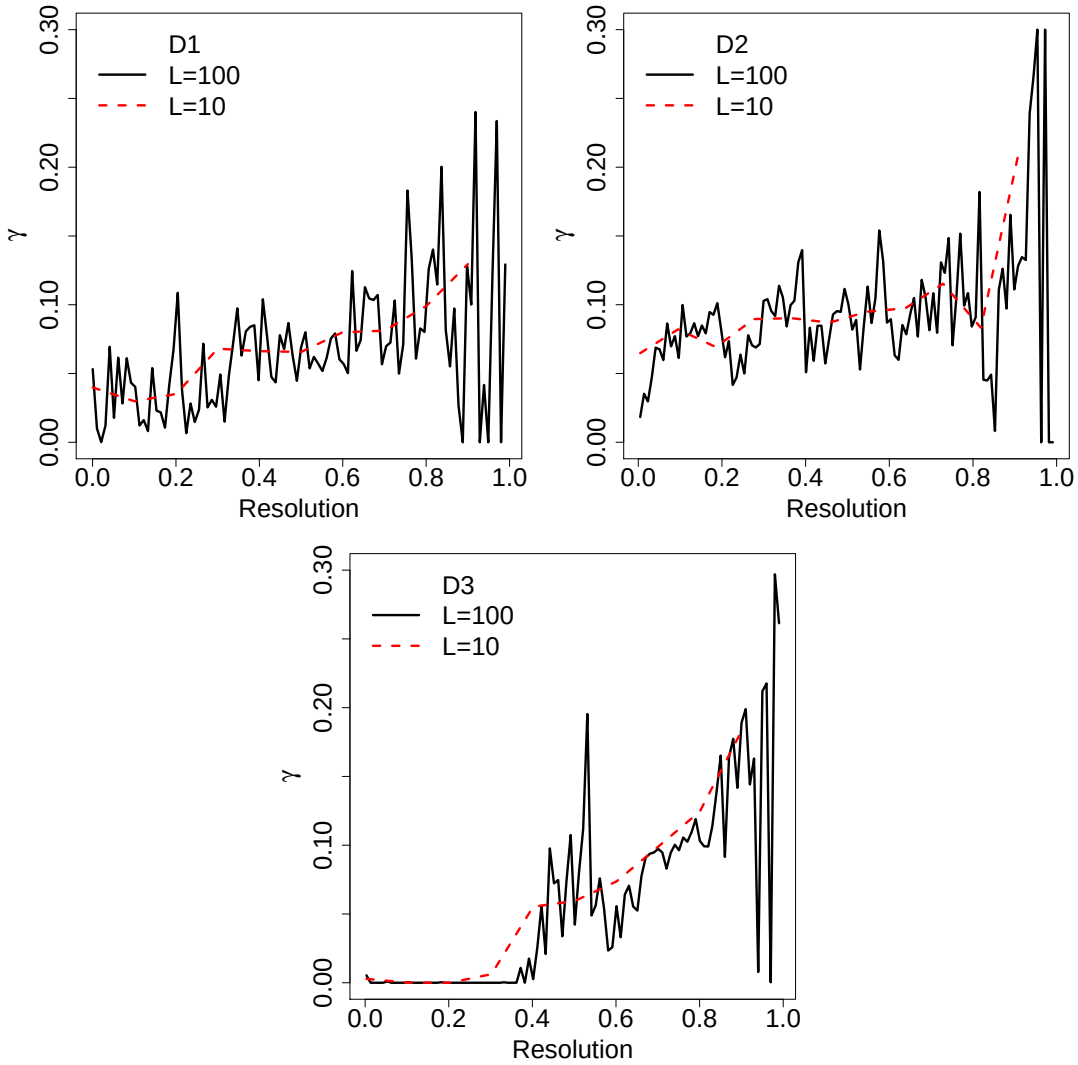

Figure 3.6: The within-class variances γ for datasets $D1$, $D2$ and $D3$. The average within-class variance was calculated for each node of the phylogenetic tree. We break the interval $(0, \max \text{ distance to the root})$ into L subintervals. Each subinterval corresponds to a specific resolution level. For each subinterval we calculate the average within-class variance for all nodes whose distance to the tree root falls within the subinterval.

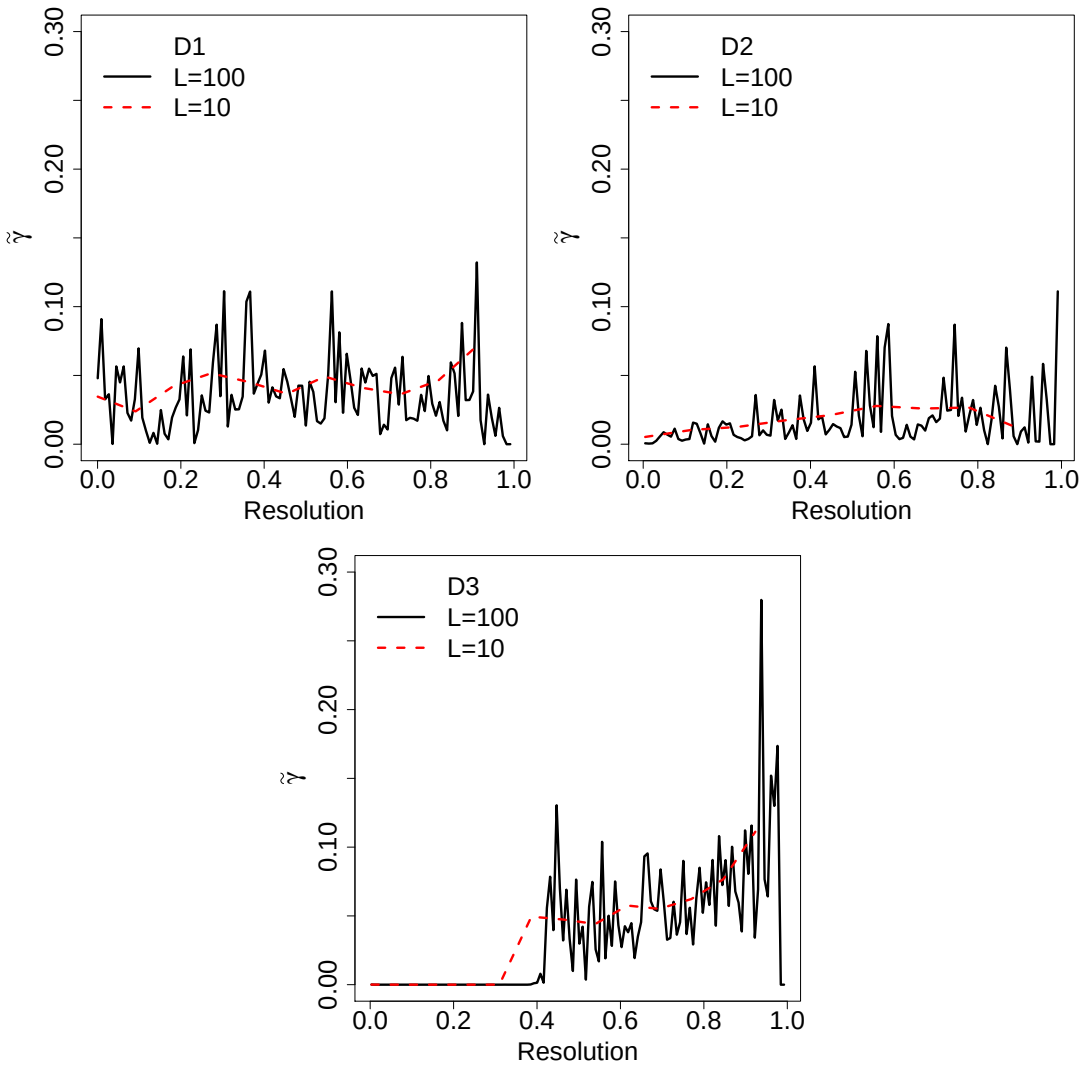

Figure 3.7: The between-class variances $\tilde{\gamma}$ for datasets $D1$, $D2$ and $D3$. The average between-class variance was calculated for each node of the phylogenetic tree. We break the interval $(0, \max \text{ distance to the root})$ into L subintervals. Each subinterval corresponds to a specific resolution level. For each subinterval we calculate the average between-class variance for all nodes whose distance to the tree root falls within the subinterval.

3.4 Conclusion

We proposed a new classification method for 16S rDNA sequence samples that employs the natural structure of microbial community data encoded by a phylogenetic tree. We showed that using the phylogenetic information leads to an improved classification accuracy compared to the state-of-the-art classification algorithms. Unlike many popular classification methods, which consider features (or OTU frequencies) in isolation, our method takes advantage of the similarities between OTUs encoded by the phylogenetic tree. We applied the algorithm to classify samples obtained from 16S rDNA studies, but the approach can also be used to classify metagenomic samples obtained by whole genome sequencing. The algorithm only requires frequencies of taxonomic groups in each sample and a phylogenetic tree that relates these groups.

Chapter 4

An Efficient Hierarchical Clustering Algorithm for Large Datasets

4.1 Introduction

Clustering is a popular unsupervised learning technique used to identify object groups within a given dataset, where intra-group objects tend to be more similar than inter-group objects. There are many different clustering algorithms [54] with applications in bio-/chem- informatics and other data mining fields [55, 19], including studies on protein families [59], functional genomics, [55], chemical scaffolds [36], *etc.* In particular, clustering algorithms have been widely adopted in the bioinformatics field after TreeView [39], a user-friendly visualization program, was made available following early studies on gene expression datasets.

Among all clustering methods, hierarchical clustering and k-means clustering are arguably the two most popular algorithms used due to their simplicity in result interpretation. In the cheminformatics field, Wards clustering [111] and Jarvis-Patrick clustering [88] are corresponding algorithms similar in spirit to hierarchical clustering and k-mean clustering, respectively. Although there is no definitive answer as to which algorithm is more accurate, hierarchical clustering has been applied more often in cheminformatics research because of its deterministic property and flexibility in flattening the resultant tree at different cutoff levels.

As discussed in Section 1.4, applying hierarchical clustering to large datasets is rather challenging, and there exists a significant need to develop a hierarchical clustering algorithm for large datasets. Approximating hierarchical clustering in subquadratic time and memory has been previously attempted [63, 64, 65, 83, 82]. However, these methods either rely on embedding into spaces that are not biologically sensible, or they produce very low resolution hierarchical structures. Our goal is to produce hierarchical results with the same resolution as the exact hierarchical method, although with less accuracy, while maintaining the bio/chemically meaningful distance metrics. For a dataset over 20 000 objects, we are limited by both $O(n^2)$ memory and time. Therefore, a reasonable approximation needs to be introduced. We observe that if an exact hierarchical tree has been constructed, one can set a similarity cutoff such that tree branches above the cutoff are distant enough from each other and represent the coarse clusters of the dataset. The branches and leaves below the cutoff represent hierarchical structures within small-scale local vicinities. For a large dataset, we are often initially interested in a "zoomed out" view of the coarse clusters, then "zoom in" to neighborhoods of interest for a finer view of the intra-group structures. The two views are often considered to be the most beneficial properties of the hierarchical clustering. For example, in the aforementioned

compound requisition problem, one would cherry pick vendor compounds from the coarse neighborhood if only a small number of compounds can be selected for purchasing. When budget and logistics permit, one could then lower the cutoff to pick more compounds within interesting coarse clusters.

To capture both distant and close views of the hierarchical structure for a large dataset, we propose a hybrid hierarchical clustering algorithm. Initially, the n objects are clustered by a k-means clustering algorithm, where k is chosen to be reasonably large, into roughly k coarse neighborhoods. We then apply the exact hierarchical clustering algorithm to cluster the k centroids into a coarse tree, as well as to the objects within each of the k clusters into k detailed trees. By replacing the k centroids in the coarse tree by the corresponding detailed trees, this two-step hybrid algorithm assembles a complete tree of n objects that can be cut, *i.e.*, zoomed in and zoomed out, at various levels. The number k can be selected by the user and controls the cutoff reflecting the average similarities of objects within each coarse neighborhood. If k is chosen so that the cutoff approximately equals the intrinsic noise in the data, the tree from the hybrid method is essentially as accurate as the tree generated by the exact hierarchical clustering algorithm. If optimized for clustering speed, $k \sim \sqrt{n}$ can be chosen to yield an approximate running time of $O(n\sqrt{n})$ and storage of $O(n\sqrt{n})$ as discussed later in detail.

4.2 The Hybrid Algorithm

In this section, we introduce a hybrid algorithm for hierarchical clustering of large datasets. Our approach combines the advantages of the partitioning and agglomerative hierarchical clustering algorithms.

Hierarchical clustering organizes the data into a dendrogram that represents the clustering structure of the data. We only consider the bottom-up clustering approach here due to its ability to capture the local clustering structure of the data. The classic agglomerative hierarchical clustering (AHC) method [32] requires computation of all pairwise distances, which has a quadratic complexity. Therefore, the construction of the distance matrix creates a bottleneck, especially for high dimensional data and expensive distance functions. Since AHC algorithms greedily merge pairs of nearest data points (clusters) into tree nodes, the exact computation of pairwise distances is important for data points that are close enough to each other, while the computation of distances between remote points is unlikely to contribute and should be avoided whenever possible. Therefore, it makes sense to partition the dataset to avoid the full distance matrix computation.

In the first step of the algorithm, we partition the data with k-means [74], a simple and effective clustering algorithm that generates a locally optimal partitioning of the data. The number of components k is predefined. The choice of k and performance of our algorithm with respect to k are discussed later in the chapter. We apply the optimized version of the exact k-means algorithm, which utilizes a triangle inequality to avoid unnecessary distance computations [40]. The clusters are initialized uniformly at random from the data points. In the second step, at the first level, AHC is applied to cluster each individual component P_i obtained by k-means into an individual detailed tree T_i. At the second level, each T_i is treated as a leaf and clustered by AHC into a coarse tree T. T, therefore, reflects both the coarse relationships among components as well as detailed relationships among members of each component.

A few questions arise in the above procedure and require careful consideration: (1) How are distances defined between the components for the second level of clustering?

(2) What should be done when the distance between a pair of component centroids is smaller than the radii of associated components?

Regarding the first question, the naive idea of taking the distance between the centroids of components as a pairwise distance between these components is undesirable. Consider two pairs of components, where the distance between the two centroids within each pair is the same. Additionally, assuming that the first pair of components have small radii while the other two components have large radii and may overlap. Clearly, the above naive approach would not capture the intuition that the second pair of components should be considered closer. We adopt the idea of data bubbles [18] and define the distance of two components P_1 and P_2 as follows:

$$Dist(P_1, P_2) = \begin{cases} dist(C_1, C_2) - (R_1 + R_2) & \text{if } dist(C_1, C_2) - \\ +1NN_1 + 1NN_2 & (R_1 + R_2) \geq 0, \\ \\ Max(1NN_1, 1NN_2) & \text{otherwise} \end{cases}$$

Here, C_i is the centroid of the partition P_i, and R_i is the radius of the component (most of the objects are located within the radius R_i around the centroid C_i). $1NN_i$ is the average 1-nearest neighbor distance within the component.

Regarding the second question, for each component P_i, we define the distance threshold r_i so that all points that are farther than r_i away from the centroid are considered outliers and removed from the component. Outliers are added as individual points and used in the second level of hierarchical clustering.

In addition, due to the nonuniform distribution of objects within a real dataset, the k-means clustering might result in components that exceed the size limit of AHC. Therefore, the hybrid algorithm might need to be recursively applied in a divide-and-

Algorithm 6: Hybrid clustering of N data points. Given k, the algorithm partitions the dataset and performs two-level hierarchical clustering to construct a tree T. (The maximum size of the input for the agglomerative hierarchical clustering (AHC) algorithm is n. It can be supplied by the user, or estimated automatically).

```
begin
    P ← ∅
    Perform optimized k-means clustering to partition the data into
    components P_i.
    for each component P_i do
        r_i = arg min_j Dist(P_i, P_j)
        Compute centroid C_i
        for each point p in P_i do
            if dist(p, C_i) > r_i then
                Remove p from P_i. Add p to P.
    for each component P_i do
        if Size(P_i) > n then
            Recursively apply hybrid clustering to generate a tree T_i.
        else
            Apply AHC to generate a tree T_i.
    Compute a combined distance matrix for all P_i and all points in P.
    Perform AHC to generate a tree T.
    return T
```

conquer manner. Occasionally, when the height of a detailed tree T_i exceeds its corresponding level-two centroid distance, its height should be propagated up to its ancestral nodes along the tree branches during the assembly of T.

Our hybrid algorithm is outlined in Algorithm 6.

4.3 The Implementation of the Exact Hierarchical Clustering Algorithm

We have been using a non-trivial assumption that AHC requires an $O(n^2)$ running time. The complexity of an AHC implementations actually varies significantly.

Cluster 3.0 [33] provides a popular AHC implementation that is used extensively in the bioinformatics field. For the average-linkage configuration, Cluster 3.0 implementation takes $O(n^3)$ time, as shown in Figure 4.6. For this study, we adopt the Murtagh reciprocal nearest neighbor idea [81], which offers a much improved $O(n^2)$ time. To test this, both Cluster 3.0 and Murtagh algorithms were implemented in Java and were applied to sample datasets sizing between 1 000 and 20 000 (40 000 for the Murtagh implementation), where each data object consisted of double vectors of length 80. As shown in Figure 4.6, the Murtagh method indeed performed at the scale of $O(n^2)$ and Cluster 3.0 at $O(n^3)$. These results are in agreement with the recent study [80]. It is worth mentioning that our Java implementation of Cluster 3.0 is two fold faster than the original C implementation, and the observation in Figure 4.6 is not an over-estimation. Note that although the Murtagh method has been used in the JKlustor program in the cheminformatics field [`http://www.chemaxon.com`], it is not widely adopted in bioinformatics. Therefore, bioinformatics researchers not using an $O(n^2)$ implementation of AHC could benefit from the release of our package.

4.4 Experimental Results

4.4.1 Data Sets

As our aim is to develop an algorithm for practical biomedical research applications, three real datasets encountered in our routine analyses were chosen. Dataset $D1$ is an activity matrix consisting of 2117 compounds profiled across 398 cancer cell lines. A subset of this matrix was previously published as the Cancer Cell Line Encyclopedia project and was described in detail by Barretina *et al.* [7]. This dataset provides

an example of a typical medium-size clustering problem involved in bioinformatics and cheminformatics research.

Dataset $D2$ is a larger high-throughput screening activity matrix of 45 000 compounds across 178 assays. This is a subset of the larger matrix described in a published HTS frequent hit study [28]. A total of 45 000 compounds that hit the most number of assays were selected, because this size approaches the upper limit of what an exact hierarchical clustering algorithm can handle on a typical desktop computer. This large dataset provides a test case to compare the speed of clustering and the qualities of resultant trees, when both the exact hierarchical clustering algorithm and the proposed hybrid algorithm are applied.

Dataset $D3$ consists of one million compounds randomly selected from our in-house compound collection, where the average Tanimoto structure similarity determined by ChemAxon two-dimensional fingerprinting is merely 0.3 [`http://www.chemaxon.com`]. As structural redundancy of the collection is low, these compounds are expected to form numerous clusters of fairly small sizes. This set is chosen to represent the more challenging problem of identifying structurally diversified compounds from a large vendor catalog as well as to enable us to study the robustness of the hybrid algorithm.

4.4.2 The Running Time and Memory Analysis

We theoretically and experimentally evaluate the running time of the hybrid algorithm. First, let us show that with a reasonable choice of the partitioning parameter k, the algorithm runs in $O(N\sqrt{N})$ time for datasets of randomly distributed objects. The running time of the algorithm is affected by (1) the time to partition the data in the k-means phase and (2) the running time of the hierarchical clustering phase. The traditional k-means algorithm requires computing kNL distances, where L is the number

of iterations. However, in the optimized version of k-means, only the first few iterations require distance computations from all the data points to all the centroids. The time needed for subsequent iterations drops significantly, because most of the distances are not computed. Thus the overall number of distance computations becomes closer to kN than to kNL. The k-means phase runs in $O(kNL')$, where $L' < L$ and can be approximately estimated experimentally. In our experiments, L' was in the range of 2 to 5. The running time of the hierarchical clustering phase includes the time required to hierarchically cluster k subsets of approximate sizes N/k and to cluster k centroids. Assuming quadratic time complexity for the AHC algorithm, the overall running time of the hybrid algorithm is $O(k^2 + N^2/k + kNL')$. As the first term k^2 is dominated by kNL', our algorithm runs in $O(N^2/k + kNL')$ time. The best performance is achieved when k is set to $\sqrt{N/L'}$, leading to an $O(N\sqrt{N})$ running time. The same analysis applies to the memory complexity which is also bounded by $O(N\sqrt{N})$.

We measured the experimental running time of the hybrid algorithm for different values of k, for both the partitioning phase and the hierarchical clustering phase. The results are shown in Figure 4.1. Even though the real data is not uniformly distributed, trends in the experimental results agree with the theory. Note, that the exact algorithm matches the cases of $k = 1$ and $k = N$. Clearly, the larger the data size, the more we gain in clustering speed compared to the exact algorithm. For example, when the parameters are optimized, the hybrid algorithm is only 5 times faster on $D1$ while it is 370 times faster on $D2$, running in 22 seconds compared to 8117 seconds for the exact algorithm.

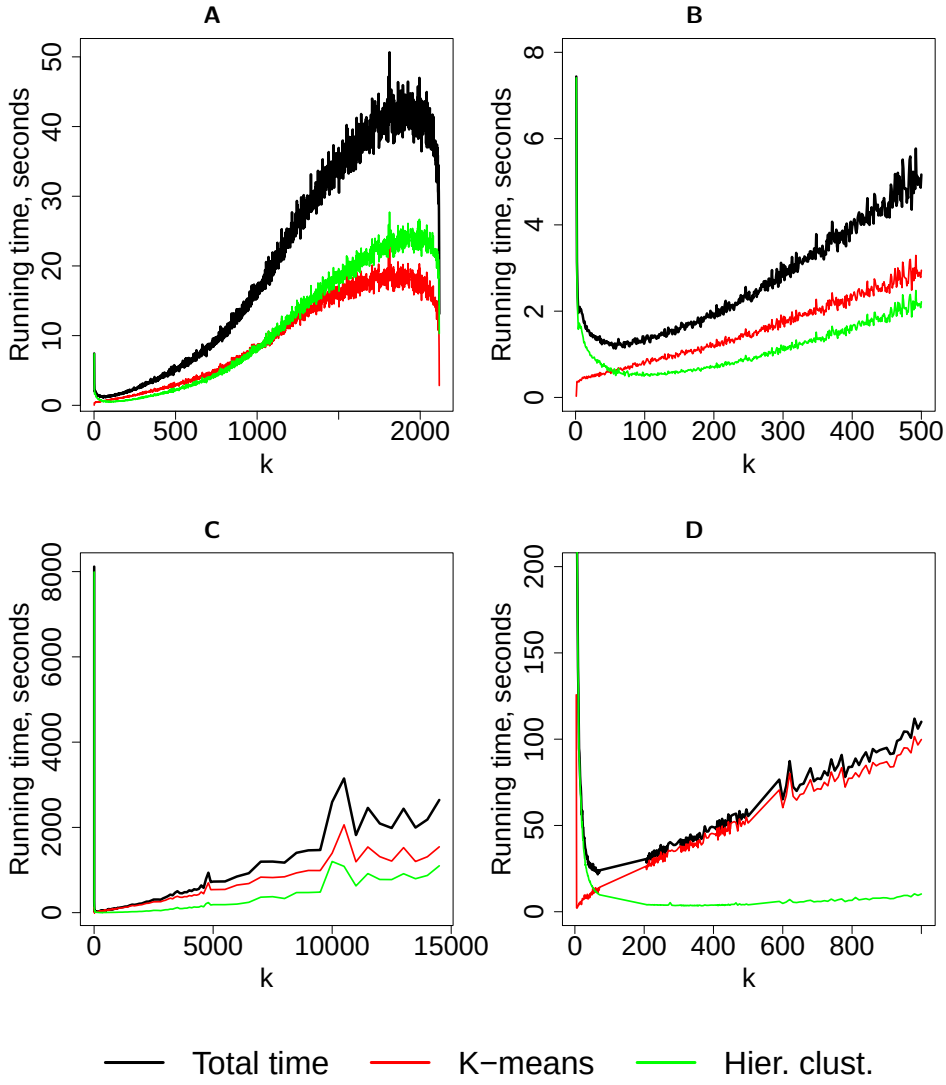

Figure 4.1: Running time of the hybrid algorithm for datasets $D1$ and $D2$. (A) Dataset $D1$, for all values of k. (B) Dataset $D1$, zoomed in for $k < 500$. (C) Dataset $D2$. (D) Dataset $D2$, zoomed in for $k < 1000$.

4.4.3 Performance Analysis

There is no universal agreement on how clustering should be performed. Therefore, methods for validating clustering results vary significantly [48]. Since our primary goal is to accelerate AHC, the hierarchical tree T produced by the AHC algorithm is taken as the gold standard and is referred to as the exact tree. The tree produced by the hybrid algorithm is referred to as a hybrid tree or an approximate tree. Quantitative comparison between the exact tree and a hybrid tree remains an open problem and few results exist in the literature. One approach is to use a well-known tree edit distance [13], but it is computationally expensive and may produce counter-intuitive results [127]. Another popular approach is to cut trees at certain heights and measure similarity between the resultant clusters. The latter was chosen for this study, as it provides visualization that can be cross-examined by biological and chemical domain knowledge. Various similarity measurements are applicable to two sets of clusters resulting from tree cuts, *e.g.*, Jaccard index [46], Rand index [89], Fowlkes-Mallows index [41], information theoretic measures [107], *etc.* Each method has its own advantages and weaknesses [108]. For example, the Rand index has an undesirable property of converging to 1 as the number of clusters increases, while the Fowlkes-Mallows index makes strong assumptions about the underlying distribution of data, making it hard to interpret the results. The information-theoretic approaches are promising for clustering validation, but require a more extensive evaluation. In our study, we chose the Jaccard index, one of the most common similarity measures for clustering.

For each of the two given datasets, we first cut the exact tree T at some height g, which was selected based on the combination of our domain knowledge of the bio- and cheminformatics problems and our visual inspection of the exact hierarchical tree

T. This resulted in a set of clusters $C(g) = \{C_1, C_2, ..., C_{|C(g)|}\}$. The corresponding hybrid tree was then cut at different cutoff values h. For each h, the Jaccard similarity index between $C(g)$ and the hybrid clusters $\widetilde{C}(h)$ was calculated according to:

$$J(C(g), \widetilde{C}(h)) = \frac{N_{11}}{N_{11} + N_{10} + N_{01}},$$

where N_{11} is the number of object pairs consisting of objects clustered together into the same cluster in both C and $\widetilde{C}$. $N_{10} + N_{01}$ is the number of object pairs consisting of objects clustered together in either C or $\widetilde{C}$ but not both. The h value that led to the highest Jaccard index was retained and used for the similarity score $S_g(T, \widetilde{T})$:

$$S_g(T, \widetilde{T}) = \arg\max_{h \in H} J(C(g), \widetilde{C}(h)).$$

The set of cutoff values H was chosen to evenly cover different granularity levels of the resulting clusterings, where granularity is defined as a percent of object pairs that cluster together.

We are particularly interested in the approximation quality for biologically meaningful clusters with pronounced activity patterns. Therefore, in the computation of the similarity score, we disregarded clusters with low average Pearson correlation of the activity profiles (below 0.2) as well as small clusters that contain less than 0.1% of the data. The selected clusters for datasets $D1$ and $D2$ are highlighted in Figures 4.2A and 4.3A, respectively. For the dataset $D1$, we additionally excluded a large cluster of low-activity compounds. Even though this cluster is well approximated by the hybrid algorithm, it dominates the resulting Jaccard index leading to an overall high similarity score. The results of the proposed similarity measures S_g on datasets $D1$ and

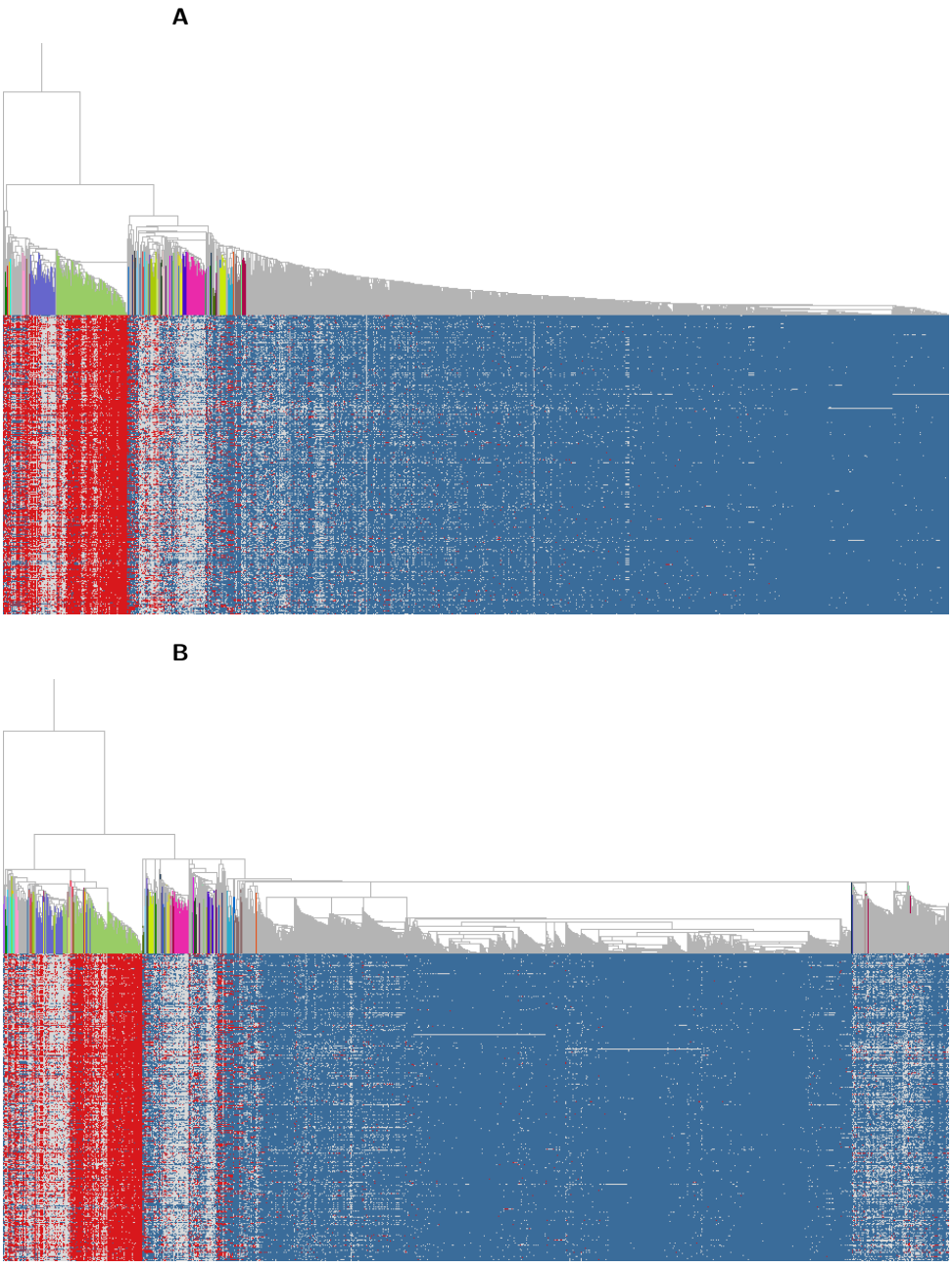

Figure 4.2: The hierarchical trees for the dataset $D1$ produced by (A) the exact algorithm and (B) the hybrid algorithm with $k = 25$. Highlighted are the biologically meaningful clusters selected for the evaluation of the approximation quality of the hybrid algorithm. The heat map illustrates the activity of compounds: red and green indicate active and inactive compounds, respectively.

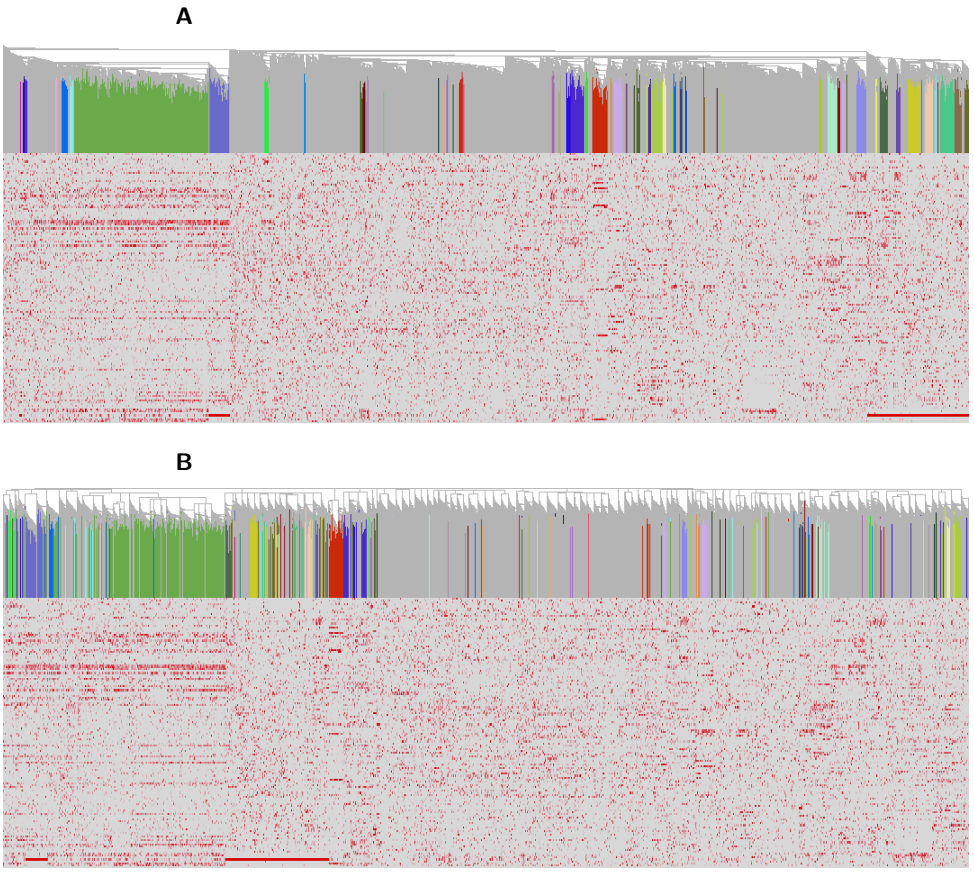

Figure 4.3: The hierarchical trees for the dataset $D2$ produced by (A) the exact algorithm and (B) the hybrid algorithm with $k = 130$. Highlighted are the biologically meaningful clusters selected for the evaluation of the approximation quality of the hybrid algorithm. The heat map illustrates the activity of compounds: the intensity of red is proportional to the compound's activity.

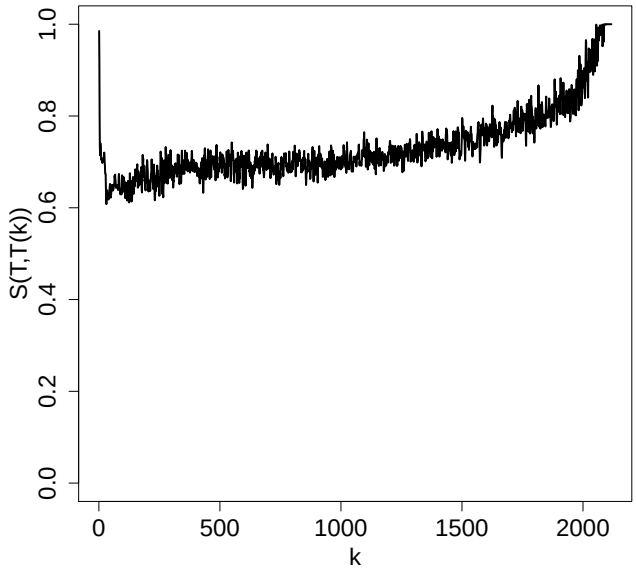

Figure 4.4: Approximation quality $S_g(T, \widetilde{T}(k))$ of the exact tree T with the hybrid trees $\widetilde{T}(k)$ for different values of the parameter k for dataset $D1$.

$D2$ for the selected clusters are shown in Figures 4.4 and 4.5. It was observed that quality measurements for both datasets are rather insensitive to the choice of k over a wide range. Since the hybrid tree $\widetilde{T}$ retains both the coarse and detailed structures within a dataset and provides approximate results for interpretations in-between, it is not surprising that $\widetilde{T}$ reasonably approximates the exact tree. Since high quality trees are produced for a wide range of the parameter values, it makes sense to optimize the parameter k mainly for improved running time in practice.

4.4.4 Performance on a Large Dataset and Robustness Analysis

A major goal in proposing our algorithm is to provide a hierarchical method that is capable of clustering datasets that contain more than 40 000 objects. Here, we

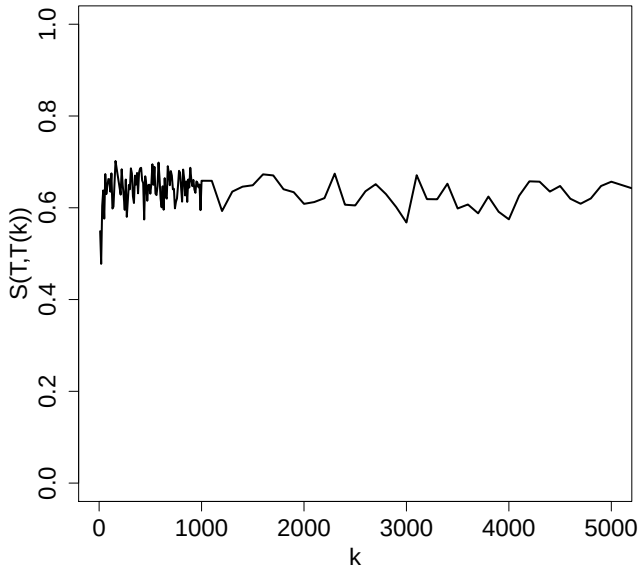

Figure 4.5: Approximation quality $S_g(T, \widetilde{T}(k))$ of the exact tree T with the hybrid trees $\widetilde{T}(k)$ for different values of the parameter k for dataset $D2$.

studied dataset $D3$, which consists of one million compounds randomly selected from our in-house compound collection. Running AHC on such a large dataset is infeasible and cheminformaticians have relied on greedy algorithms such as Sphere Exclusion (SE) [45] to partition the compounds into clusters. SE requires a fixed similarity cutoff value as its input. It randomly selects a query compound and extracts all remaining compounds, where their structural similarities to the query compound are above the predefined threshold. The extraction and exclusion process is iterated until the collection is exhausted. Because the exact tree is not available for a dataset as large as $D3$, performance comparisons between SE and hybrid algorithms cannot be conducted in a manner similar to what we presented in Sections 4.4.2 and 4.4.3. Nevertheless, we speculate that the hybrid method provides a result closer to the exact AHC tree than to SE. This is because no super-sized compound cluster is expected in $D3$ based on our

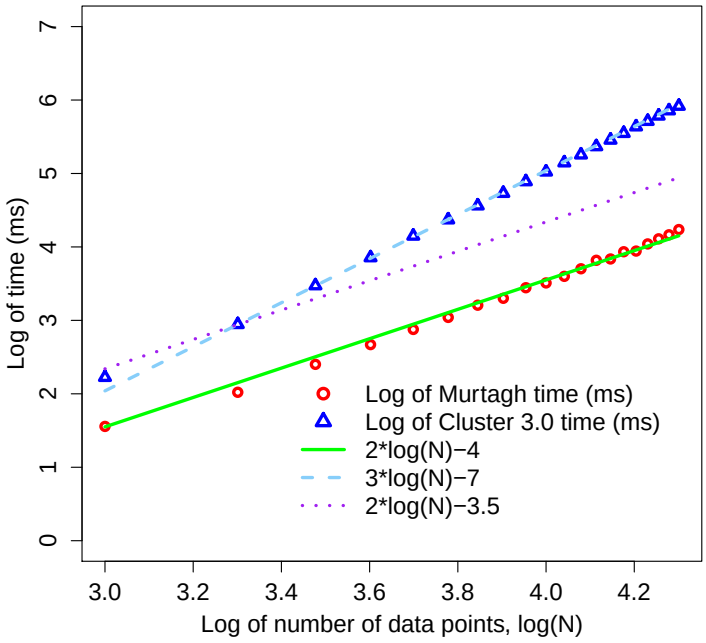

Figure 4.6: The performance comparison between the Murtagh method and the Java implementation of the Cluster 3.0 method. The running time of the Murtagh method matches a linear curve of slope 2 while the running time of the Cluster 3.0 method matches a linear curve of slope 3, showing that their running time are of $O(n^2)$ and $O(n^3)$, respectively. The green curve is a linear curve of slope 2 that crosses the curve of Cluster 3.0 running time included to make the comparison easier.

domain knowledge, *i.e.*, the sizes of chemically interesting clusters are small. The first k-means clustering step is expected to produce only large components of structurally diverse compounds and is unlikely to break down small groups of highly similar compounds. The SE algorithm, on the other hand, produced flattened clusters based on a rather subjective similarity threshold, which may not match the average similarities in small clusters.

A main criticism on SE is its greediness, which led to different clustering results in different runs in our experiment. As the hybrid algorithm also has a random component in the k-means stage, it would be interesting to compare the two methods for robustness in the results. We shuffled records in the one million compound dataset ten times and applied both algorithms. We then measured how well each method was able to reproduce its own results. In particular, we first applied a cutoff value to flatten hybrid trees into a similar number of clusters as in the output of the SE algorithm. Then, through random sampling of compound pairs in the output clusters, we estimated the probability that a pair of compounds will cluster together in consecutive runs to be 37.1% with a standard deviation of 0.9% for the hybrid method, and 27.8% with a standard deviation of 1.6% for the SE methods (p-value is $1 \times e^{-10}$). Similarly, we also estimated the probability that a pair of compounds will not cluster together in consecutive runs to be 99.8% and 99.9%, respectively. These results indicate the superior robustness of the hybrid algorithms across multiple runs.

4.5 Conclusion

In the chapter, we have introduced a hybrid hierarchical clustering algorithm that requires approximately $O(n\sqrt{n})$ running time and $O(n\sqrt{n})$ memory, producing hi-

erarchical trees similar to what the exact hierarchical algorithm offers but applicable to much larger datasets. With three example datasets, the hybrid algorithm was demonstrated to be much faster, reasonably accurate and robust for clustering large datasets encountered in bioinformatics and cheminformatics research. The software package has been made available to the informatics community and should prove very useful when applied to a wide range of data mining problems.

Chapter 5

Conclusions

Recent advances in biotechnology have created new opportunities for the life sciences. High-throughput methods (NGS, HTS/uHTS) are increasingly applied in various areas of biological and chemical research. For example, metagenomics approach combined with NGS has opened a door into the previously hidden world of microorganisms, while HTS/uHTS technologies have created new opportunities in drug discovery. There is a need for new methods and tools to efficiently analyze the increasing amounts of data which would also handle challenges and take into account natural properties of the data.

In this dissertation, we addressed three problems that arise in metagenomics and cheminformatics. First, we presented an algorithm for separating short paired-end NGS reads from different bacterial genomes of similar abundance levels, and extended it with a new abundance-based binning method to handle arbitrary abundance ratios. Second, we proposed a novel supervised classification method for metagenomic samples that takes advantage of the natural structure in microbial community data encoded by a phylogenetic tree. This model allows us to take advantage of environment-specific

compositional patterns that may contain features at multiple granularity levels. Our method is based on the multinomial logistic regression model with a tree-guided penalty function. We provide an efficient optimization algorithm to learn the model parameters. Finally, we presented a hybrid hierarchical clustering algorithm that combines the k-means clustering and agglomerative hierarchical clustering to efficiently cluster large datasets encountered in bioinformatics and cheminformatics research.

Bibliography

[1] Davide Albanese, Roberto Visintainer, Stefano Merler, Samantha Riccadonna, Giuseppe Jurman, and Cesare Furlanello. Mlpy: Machine learning python, 2012.

[2] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, October 1990.

[3] R. I. Amann, W. Ludwig, and K. H. Schleifer. Phylogenetic identification and in situ detection of individual microbial cells without cultivation. *Microbiological reviews*, 59(1):143–169, March 1995.

[4] R. I. Amann, W. Ludwig, and K. H. Schleifer. Phylogenetic identification and in situ detection of individual microbial cells without cultivation. *Microbiological reviews*, 59(1):143–169, 1995.

[5] Manimozhiyan Arumugam, Jeroen Raes, Eric Pelletier, Denis Le Paslier, Takuji Yamada, Daniel R. Mende, Gabriel R. Fernandes, Julien Tap, Thomas Bruls, Jean-Michel Batto, Marcelo Bertalan, Natalia Borruel, Francesc Casellas, Leyden Fernandez, Laurent Gautier, Torben Hansen, Masahira Hattori, Tetsuya Hayashi, Michiel Kleerebezem, Ken Kurokawa, Marion Leclerc, Florence Levenez, Chaysavanh Manichanh, H. Bjorn Nielsen, Trine Nielsen, Nicolas Pons, Julie Poulain, Junjie Qin, Thomas Sicheritz-Ponten, Sebastian Tims, David Torrents, Edgardo Ugarte, Erwin G. Zoetendal, Jun Wang, Francisco Guarner, Oluf Pedersen, Willem M. de Vos, Soren Brunak, Joel Dore, Jean Weissenbach, S. Dusko Ehrlich, and Peer Bork. Enterotypes of the human gut microbiome. *Nature*, 473(7346):174–180, May 2011.

[6] Musa H. Asyali, Dilek Colak, Omer Demirkaya, and Mehmet S. Inan. Gene expression profile classification: a review. *Current Bioinformatics*, pages 55–73, January 2006.

[7] Jordi Barretina, Giordano Caponigro, Nicolas Stransky, Kavitha Venkatesan, Adam A. Margolin, Sungjoon Kim, Christopher J. Wilson, Joseph Lehár, Gregory V. Kryukov, Dmitriy Sonkin, Anupama Reddy, Manway Liu, Lauren Murray, Michael F. Berger, John E. Monahan, Paula Morais, Jodi Meltzer, Adam Korejwa, Judit Jané-Valbuena, Felipa A. Mapa, Joseph Thibault, Eva Bric-Furlong, Pichai Raman, Aaron Shipway, Ingo H. Engels, Jill Cheng, Guoying K. Yu, Jianjun Yu, Peter Aspesi, Melanie de Silva, Kalpana Jagtap, Michael D. Jones, Li Wang, Charles Hatton, Emanuele Palescandolo, Supriya Gupta, Scott Mahan, Carrie Sougnez, Robert C. Onofrio, Ted Liefeld, Laura MacConaill, Wendy

Winckler, Michael Reich, Nanxin Li, Jill P. Mesirov, Stacey B. Gabriel, Gad Getz, Kristin Ardlie, Vivien Chan, Vic E. Myer, Barbara L. Weber, Jeff Porter, Markus Warmuth, Peter Finan, Jennifer L. Harris, Matthew Meyerson, Todd R. Golub, Michael P. Morrissey, William R. Sellers, Robert Schlegel, and Levi A. Garraway. The Cancer Cell Line Encyclopedia enables predictive modelling of anticancer drug sensitivity. *Nature*, 483(7391):603–607, March 2012.

[8] O. Béjà, M. T. Suzuki, E. V. Koonin, L. Aravind, A. Hadd, L. P. Nguyen, R. Villacorta, M. Amjadi, C. Garrigues, S. B. Jovanovich, R. A. Feldman, and E. F. DeLong. Construction and analysis of bacterial artificial chromosome libraries from a marine microbial assemblage. *Environmental Microbiology*, 2(5):516–529, 2000.

[9] Asa Ben-Hur, Cheng S. Ong, Soren Sonnenburg, Bernhard Scholkopf, and Gunnar Ratsch. Support vector machines and kernels for computational biology. *PLoS computational biology*, 4(10):e1000173+, October 2008.

[10] Dennis A. Benson, Ilene Karsch-Mizrachi, David J. Lipman, James Ostell, and Eric W. Sayers. GenBank. *Nucleic acids research*, 37(Database issue):D26–31, January 2009.

[11] David R. Bentley. Whole-genome re-sequencing. *Current opinion in genetics & development*, 16(6):545–552, December 2006.

[12] Stephen D. Bentley and Julian Parkhill. Comparative genomic structure of prokaryotes. *Annual Review of Genetics*, 38:771–791, December 2004.

[13] Philip Bille. A survey on tree edit distance and related problems. *Theor. Comput. Sci.*, 337(1-3):217–239, 2005.

[14] Balls In Bins. http://www.mathpages.com/home/kmath199.htm.

[15] Anne-Laure Boulesteix, Silke Janitza, Jochen Kruppa, and Inke R. König. Overview of random forest methodology and practical guidance with emphasis on computational biology and bioinformatics. *WIREs Data Mining Knowl Discov*, 2(6):493–507, 2012.

[16] Michael Boutros, Amy A. Kiger, Susan Armknecht, Kim Kerr, Marc Hild, Britta Koch, Stefan A. Haas, Heidelberg F. Consortium, Renato Paro, and Norbert Perrimon. Genome-Wide RNAi Analysis of Growth and Viability in Drosophila Cells. *Science*, 303(5659):832–835, February 2004.

[17] Arthur Brady and Steven L. Salzberg. Phymm and PhymmBL: metagenomic phylogenetic classification with interpolated Markov models. *Nat Meth*, 6(9):673–676, 2009.

[18] Markus M. Breunig, Hans P. Kriegel, Peer Kröger, and Jörg Sander. Data bubbles: quality preserving performance boosting for hierarchical clustering. In *SIGMOD '01: Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, pages 79–90, New York, NY, USA, 2001. ACM.

[19] Sylvain Brohee and Jacques van Helden. Evaluation of clustering algorithms for protein-protein interaction networks. *BMC Bioinformatics*, 7(1):488+, November 2006.

[20] J. Gregory Caporaso, Kyle Bittinger, Frederic D. Bushman, Todd Z. DeSantis, Gary L. Andersen, and Rob Knight. PyNAST: a flexible tool for aligning sequences to a template alignment. *Bioinformatics (Oxford, England)*, 26(2):266–267, January 2010.

[21] J. Gregory Caporaso, Justin Kuczynski, Jesse Stombaugh, Kyle Bittinger, Frederic D. Bushman, Elizabeth K. Costello, Noah Fierer, Antonio G. Pena, Julia K. Goodrich, Jeffrey I. Gordon, Gavin A. Huttley, Scott T. Kelley, Dan Knights, Jeremy E. Koenig, Ruth E. Ley, Catherine A. Lozupone, Daniel McDonald, Brian D. Muegge, Meg Pirrung, Jens Reeder, Joel R. Sevinsky, Peter J. Turnbaugh, William A. Walters, Jeremy Widmann, Tanya Yatsunenko, Jesse Zaneveld, and Rob Knight. QIIME allows analysis of high-throughput community sequencing data. *Nature methods*, 7(5):335–336, May 2010.

[22] Mark J. Chaisson and Pavel A. Pevzner. Short read fragment assembly of bacterial genomes. *Genome research*, 18(2):324–330, February 2008.

[23] Soumitesh Chakravorty, Danica Helb, Michele Burday, Nancy Connell, and David Alland. A detailed analysis of 16s ribosomal RNA gene segments for the diagnosis of pathogenic bacteria. *J Microbiol Methods*, 69(2), 2007.

[24] Chon-Kit Chan, Arthur Hsu, Saman Halgamuge, and Sen-Lin Tang. Binning sequences using very sparse labels within a metagenome. *BMC Bioinformatics*, 9(1), 2008.

[25] Qin Chang, Yihui Luan, and Fengzhu Sun. Variance adjusted weighted UniFrac: a powerful beta diversity measure for comparing communities based on phylogeny. *BMC Bioinformatics*, 12(1):118+, 2011.

[26] Anveshi Charuvaka and Huzefa Rangwala. Evaluation of short read metagenomic assembly. *BMC Genomics*, 12(Suppl 2):S8+, 2011.

[27] Sourav Chatterji, Ichitaro Yamazaki, Zhaojun Bai, and Jonathan A. Eisen. Compostbin: a dna composition-based algorithm for binning environmental shotgun reads. In *Proceedings of the 12th annual international conference on Research in computational molecular biology*, RECOMB'08, pages 17–28. Springer, 2008.

[28] Jianwei Che, Frederick J. King, Bin Zhou, and Yingyao Zhou. Chemical and Biological Properties of Frequent Screening Hits. *J. Chem. Inf. Model.*, 52(4):913–926, March 2012.

[29] The Human Microbiome Project Consortium. Structure, function and diversity of the healthy human microbiome. *Nature*, 486(7402):207–214, June 2012.

[30] E. K. Costello, C. L. Lauber, M. Hamady, N. Fierer, J. I. Gordon, and R. Knight. Bacterial community variation in human body habitats across space and time. *Science*, 326(5960):1694–1697, December 2009.

[31] E. K. Costello, C. L. Lauber, M. Hamady, N. Fierer, J. I. Gordon, and R. Knight. Bacterial Community Variation in Human Body Habitats Across Space and Time. *Science*, 326(5960):1694–1697, December 2009.

[32] William H. Day and Herbert Edelsbrunner. Efficient algorithms for agglomerative hierarchical clustering methods. *Journal of Classification*, 1(1):7–24, December 1984.

[33] M. J. L. de Hoon, S. Imoto, J. Nolan, and S. Miyano. Open source clustering software. *Bioinformatics*, 20(9):1453–1454, June 2004.

[34] Naryttza Diaz, Lutz Krause, Alexander Goesmann, Karsten Niehaus, and Tim Nattkemper. TACOA - Taxonomic classification of environmental genomic fragments using a kernelized nearest neighbor approach. *BMC Bioinformatics*, 10(1):56+, 2009.

[35] Juliane C. Dohm, Claudio Lottaz, Tatiana Borodina, and Heinz Himmelbauer. SHARCGS, a fast and highly accurate short-read assembly algorithm for de novo genomic sequencing. *Genome Research*, 17(11):1697–1706, November 2007.

[36] Geoff M. Downs and John M. Barnard. *Clustering Methods and Their Uses in Computational Chemistry*, pages 1–40. John Wiley and Sons, Inc., 2003.

[37] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[38] Robert C. Edgar. Search and clustering orders of magnitude faster than BLAST. *Bioinformatics (Oxford, England)*, 26(19):2460–2461, October 2010.

[39] Michael B. Eisen, Paul T. Spellman, Patrick O. Brown, and David Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences*, 95(25):14863–14868, December 1998.

[40] C. Elkan. Using the triangle inequality to accelerate kMeans, 2003.

[41] E. B. Fowlkes and C. L. Mallows. A Method for Comparing Two Hierarchical Clusterings. *Journal of the American Statistical Association*, 78(383):553–569, 1983.

[42] Tarini Ghosh, M. Monzoorul Haque, and Sharmila Mande. DiScRIBinATE: a rapid method for accurate taxonomic classification of metagenomic sequences. *BMC Bioinformatics*, 11(Suppl 7):S14+, 2010.

[43] Steven R. Gill, Mihai Pop, Robert T. DeBoy, Paul B. Eckburg, Peter J. Turnbaugh, Buck S. Samuel, Jeffrey I. Gordon, David A. Relman, Claire M. Fraser-Liggett, and Karen E. Nelson. Metagenomic Analysis of the Human Distal Gut Microbiome. *Science*, 312(5778):1355–1359, June 2006.

[44] Enrico Glaab, Jonathan M. Garibaldi, and Natalio Krasnogor. Learning pathway-based decision rules to classify microarray cancer samples. In *German Conference on Bioinformatics 2010*, volume 173 of *Lecture Notes in Informatics*, pages 123–134. Gesellschaft fuer Informatik, 2010.

[45] A. Gobbi and M. L. Lee. DISE: Directed Sphere Exclusion. *J. Chem. Inf. Comput. Sci.*, 43(1):317–323, January 2003.

[46] Lieve Hamers, Yves Hemeryck, Guido Herweyers, Marc Janssen, Hans Keters, Ronald Rousseau, and André Vanhoutte. Similarity measures in scientometric research: The Jaccard index versus Salton's cosine formula. *Information Processing & Management*, 25(3):315–318, January 1989.

[47] J. Handelsman, M. R. Rondon, S. F. Brady, J. Clardy, and R. M. Goodman. Molecular biological access to the chemistry of unknown soil microbes: a new frontier for natural products. *Chemistry & biology*, 5(10), October 1998.

[48] J. Handl, J. Knowles, and D. B. Kell. Computational cluster validation in post-genomic data analysis. *Bioinformatics*, 21(15):3201–3212, August 2005.

[49] S. J. Helyar, J. Hemmer-Hansen, D. Bekkevold, M. I. Taylor, R. Ogden, M. T. Limborg, A. Cariani, G. E. Maes, E. Diopere, G. R. Carvalho, and E. E. Nielsen. Application of SNPs for population genetics of nonmodel organisms: new opportunities and challenges. *Molecular Ecology Resources*, 11:123–136, 2011.

[50] R. P. Hertzberg and A. J. Pope. High-throughput screening: new technology for the 21st century. *Curr Opin Chem Biol*, 4(4):445–451, August 2000.

[51] Matthias Hess, Alexander Sczyrba, Rob Egan, and et al. Metagenomic discovery of biomass-degrading genes and genomes from cow rumen. *Science*, 331(6016):463–467, 2011.

[52] Daniel H. Huson, Alexander F. Auch, Ji Qi, and Stephan C. Schuster. MEGAN analysis of metagenomic data. *Genome research*, 17(3):377–386, March 2007.

[53] Laurent Jacob, Guillaume Obozinski, and Jean-Philippe Vert. Group lasso with overlap and graph lasso. In *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*, pages 433–440, New York, NY, USA, 2009. ACM.

[54] A. K. Jain, M. N. Murty, and P. J. Flynn. Data Clustering: A Review. *ACM Comput. Surv.*, 31(3):264–323, September 1999.

[55] Daxin Jiang, Chun Tang, and Aidong Zhang. Cluster analysis for gene expression data: a survey. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1370–1386, November 2004.

[56] William Jones. High-Throughput Sequencing and Metagenomics. *Estuaries and Coasts*, pages 944–952, July 2010.

[57] Steven W. Kembel, Jonathan A. Eisen, Katherine S. Pollard, and Jessica L. Green. The phylogenetic diversity of metagenomes. *PLoS ONE*, 6(8):e23214+, August 2011.

[58] Seyoung Kim and Eric P. Xing. Tree-guided group lasso for multi-task regression with structured sparsity. In *Proceedings of the 27th International Conference on Machine Learning*, 2010.

[59] William Klimke, Richa Agarwala, Azat Badretdin, Slava Chetvernin, Stacy Ciufo, Boris Fedorov, Boris Kiryutin, Kathleen O'Neill, Wolfgang Resch, Sergei Resenchuk, Susan Schafer, Igor Tolstoy, and Tatiana Tatusova. The National Center

for Biotechnology Information's Protein Clusters Database. *Nucleic Acids Research*, 37(suppl 1):D216–D223, January 2009.

[60] Dan Knights, Elizabeth K. Costello, and Rob Knight. Supervised classification of human microbiota. *FEMS Microbiology Reviews*, 35(2):343–359, 2011.

[61] Dan Knights, Laura W. Parfrey, Jesse Zaneveld, Catherine Lozupone, and Rob Knight. Human-associated microbial signatures: examining their predictive value. *Cell Host Microbe*, 10(4):292–296, October 2011.

[62] Lutz Krause, Naryttza N. Diaz, Alexander Goesmann, Scott Kelley, Tim W. Nattkemper, Forest Rohwer, Robert A. Edwards, and Jens Stoye. Phylogenetic classification of short environmental DNA fragments. *Nucleic Acids Research*, 36(7):2230–2239, April 2008.

[63] Drago Krznaric and Christos Levcopoulos. The first subquadratic algorithm for complete linkage clustering. In John Staples, Peter Eades, Naoki Katoh, and Alistair Moffat, editors, *Algorithms and Computation, 6th International Symposium, ISAAC 95, Cairns, Australia, December 4-6, 1995, Proceedings*, volume 1004 of *Lecture Notes in Computer Science*, pages 392–401. Springer, 1995.

[64] Drago Krznaric and Christos Levcopoulos. Optimal algorithms for complete linkage clustering in d dimensions. In Igor Prvara and Peter Ruzicka, editors, *Mathematical Foundations of Computer Science 1997, 22nd International Symposium, MFCS 97, Bratislava, Slovakia, August 25-29, 1997, Proceedings*, volume 1295 of *Lecture Notes in Computer Science*, pages 368–377. Springer, 1997.

[65] Meelis Kull and Jaak Vilo. Fast approximate hierarchical clustering using similarity heuristics. *BioData Mining*, 1(1):9+, 2008.

[66] E. S. Lander and M. S. Waterman. Genomic mapping by fingerprinting random clones: a mathematical analysis. *Genomics*, 2(3):231–239, April 1988.

[67] Henry C. M. Leung, S. M. Yiu, Bin Yang, and et al. A robust and accurate binning algorithm for metagenomic sequences with arbitrary species abundance ratio. *Bioinformatics*, 27(11):1489–1495, June 2011.

[68] Xiaoman Li and Michael S. Waterman. Estimating the Repeat Structure and Length of DNA Sequences Using l-Tuples. *Genome Research*, 13(8):1916–1922, August 2003.

[69] Zhenqiu Liu, William Hsiao, Brandi L. Cantarel, Elliott F. Drábek, and Claire Fraser-Liggett. Sparse distance-based learning for simultaneous multiclass classification and feature selection of metagenomic data. *Bioinformatics*, 27(23):3242–3249, December 2011.

[70] Catherine Lozupone and Rob Knight. UniFrac: a new phylogenetic method for comparing microbial communities. *Applied and environmental microbiology*, 71(12):8228–8235, December 2005.

[71] Catherine A. Lozupone and Rob Knight. Global patterns in bacterial diversity. *Proceedings of the National Academy of Sciences*, 104(27):11436–11440, July 2007.

[72] Catherine A. Lozupone and Rob Knight. Species divergence and the measurement of microbial diversity. *FEMS Microbiology Reviews*, 32(4):557–578, July 2008.

[73] Rachel Mackelprang, Mark P. Waldrop, Kristen M. DeAngelis, and et al. Metagenomic analysis of a permafrost microbial community reveals a rapid response to thaw. *Nature*, 480(7377):368–371, 2011.

[74] J. B. MacQueen. Some Methods for classification and analysis of multivariate observations. In *Procedings of the Fifth Berkeley Symposium on Math, Statistics, and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[75] David Madigan, Alexander Genkin, David D. Lewis, Dmitriy Fradkin, and David D. Lewis Consulting. Bayesian multinomial logistic regression for author identification. In *In Maxent Conference*, pages 509–516, 2005.

[76] Marcel Margulies, Michael Egholm, William E. Altman, and et al. Genome sequencing in microfabricated high-density picolitre reactors. *Nature*, 437(7057):376–380, 2005.

[77] Alice C. McHardy, Hector G. Martin, Aristotelis Tsirigos, and et al. Accurate phylogenetic classification of variable-length DNA fragments. *Nature Methods*, 4(1):63–72, 2006.

[78] Lukas Meier, Sara van de Geer, and Peter Buhlmann. The group lasso for logistic regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(1):53–71, February 2008.

[79] M. Monzoorul Haque, Tarini Shankar S. Ghosh, Dinakar Komanduri, and Sharmila S. Mande. SOrt-ITEMS: Sequence orthology based approach for improved taxonomic estimation of metagenomic sequences. *Bioinformatics (Oxford, England)*, 25(14):1722–1730, 2009.

[80] Daniel Müllner. fastcluster: Fast hierarchical, agglomerative clustering routines for R and Python. *Journal of Statistical Software*, 53(9):1–18, 2013.

[81] F. Murtagh. Complexities of hierarchic clustering algorithms: state of the art. *Jomputational Statistic Quarterly*, 1(2):101–113, 1984.

[82] F. Murtagh and P. Contreras. Fast, linear time, m-adic hierarchical clustering for search and retrieval using the baire metric, with linkages to generalized ultrametrics, hashing, formal concept analysis, and precision of data measurement. *P-Adic Numbers, Ultrametric Analysis, and Applications*, 4(1):46–56, 2012.

[83] Fionn Murtagh, Geoff Downs, and Pedro Contreras. Hierarchical clustering of massive, high dimensional data sets by exploiting ultrametric embedding. *SIAM J. Scientific Computing*, 30(2):707–730, 2008.

[84] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[85] Shruthi Prabhakara and Raj Acharya. A two-way multi-dimensional mixture model for clustering metagenomic sequences. In *Proceedings of the 2nd ACM Conference on Bioinformatics, Computational Biology and Biomedicine*, BCB '11, pages 191–200. ACM, 2011.

[86] Morgan N. Price, Paramvir S. Dehal, and Adam P. Arkin. FastTree 2 Approximately Maximum-Likelihood Trees for Large Alignments. *PLoS ONE*, 5(3):e9490+, March 2010.

[87] Fengzhu Sun Qin Chang, Yihui Luan. Variance adjusted weighted UniFrac: a powerful beta diversity measure for comparing communities based on phylogeny. *BMC Bioinformatics*, 12(1):118+, 2011.

[88] E. A. Patrick R. A. Jarvis. Clustering Using a Similarity Measure Based on Shared Near Neighbors . *JComputers, IEEE Transactions on*, C-22(11), 1973.

[89] William M. Rand. Objective Criteria for the Evaluation of Clustering Methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

[90] Michael S. Rappé and Stephen J. Giovannoni. The uncultured microbial majority. *Annual review of microbiology*, 57(1):369–394, 2003.

[91] Daniel C. Richter, Felix Ott, Alexander F. Auch, Ramona Schmid, and Daniel H. Huson. MetaSim: a Sequencing Simulator for Genomics and Metagenomics. *PLoS ONE*, 3(10):e3373+, October 2008.

[92] Patrick D. Schloss and Jo Handelsman. Introducing TreeClimber, a test to compare microbial community structures. *Appl. Environ. Microbiol.*, 72(4):2379–2384, April 2006.

[93] Patrick D. Schloss, Sarah L. Westcott, Thomas Ryabin, Justine R. Hall, Martin Hartmann, Emily B. Hollister, Ryan A. Lesniewski, Brian B. Oakley, Donovan H. Parks, Courtney J. Robinson, Jason W. Sahl, Blaz Stres, Gerhard G. Thallinger, David J. Van Horn, and Carolyn F. Weber. Introducing mothur: open-source, platform-independent, community-supported software for describing and comparing microbial communities. *Applied and Environmental Microbiology*, 75(23):7537–7541, December 2009.

[94] Neethu Shah, Haixu Tang, Thomas G. Doak, and Yuzhen Ye. Comparing bacterial communities inferred from 16S rRNA gene sequencing and shotgun metagenomics. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, pages 165–176, 2011.

[95] R. Sibson. SLINK: An optimally efficient algorithm for the single-link cluster method. *The Computer Journal*, 16(1):30–34, January 1973.

[96] Jared T. Simpson, Kim Wong, Shaun D. Jackman, Jacqueline E. Schein, Steven J. M. Jones, and İnanç Birol. ABySS: A parallel assembler for short read sequence data. *Genome Research*, 19(6):1117–1123, June 2009.

[97] Amoolya H. Singh, Tobias Doerks, Ivica Letunic, and et al. Discovering Functional Novelty in Metagenomes: Examples from Light-Mediated Processes. *J. Bacteriol.*, 191(1):32–41, 2009.

[98] Xiaoquan Su, Jian Xu, and Kang Ning. Meta-Storms: efficient search for similar microbial communities based on a novel indexing scheme and similarity score for metagenomic data. *Bioinformatics*, 28(19):2493–2501, October 2012.

[99] Hanno Teeling, Jost Waldmann, Thierry Lombardot, Margarete Bauer, and Frank Glockner. TETRA: a web-service and a stand-alone program for the analysis and comparison of tetranucleotide usage patterns in DNA sequences. *BMC Bioinformatics*, 5(1):163+, October 2004.

[100] Susannah G. Tringe, Christian von Mering, Arthur Kobayashi, and et al. Comparative Metagenomics of Microbial Communities. *Science*, 308(5721):554–557, 2005.

[101] Peter J. Turnbaugh, Micah Hamady, Tanya Yatsunenko, Brandi L. Cantarel, Alexis Duncan, Ruth E. Ley, Mitchell L. Sogin, William J. Jones, Bruce A. Roe, Jason P. Affourtit, Michael Egholm, Bernard Henrissat, Andrew C. Heath, Rob Knight, and Jeffrey I. Gordon. A core gut microbiome in obese and lean twins. *Nature*, 457(7228):480–484, January 2009.

[102] Peter J. Turnbaugh, Ruth E. Ley, Micah Hamady, Claire M. Fraser-Liggett, Rob Knight, and Jeffrey I. Gordon. The Human Microbiome Project. *Nature*, 449(7164):804–810, October 2007.

[103] Peter J. Turnbaugh, Ruth E. Ley, Michael A. Mahowald, Vincent Magrini, Elaine R. Mardis, and Jeffrey I. Gordon. An obesity-associated gut microbiome with increased capacity for energy harvest. *Nature*, 444(7122):1027–1031, December 2006.

[104] Gene W. Tyson, Jarrod Chapman, Philip Hugenholtz, Eric E. Allen, Rachna J. Ram, Paul M. Richardson, Victor V. Solovyev, Edward M. Rubin, Daniel S. Rokhsar, and Jillian F. Banfield. Community structure and metabolism through reconstruction of microbial genomes from the environment. *Nature*, 428(6978):37–43, March 2004.

[105] Stijn van Dongen. *Graph Clustering by Flow Simulation*. PhD thesis, University of Utrecht, May 2000.

[106] J. Craig Venter, Karin Remington, John F. Heidelberg, Aaron L. Halpern, Doug Rusch, Jonathan A. Eisen, Dongying Wu, Ian Paulsen, Karen E. Nelson, William Nelson, Derrick E. Fouts, Samuel Levy, Anthony H. Knap, Michael W. Lomas, Ken Nealson, Owen White, Jeremy Peterson, Jeff Hoffman, Rachel Parsons, Holly Baden-Tillson, Cynthia Pfannkoch, Yu-Hui Rogers, and Hamilton O. Smith. Environmental Genome Shotgun Sequencing of the Sargasso Sea. *Science*, 304(5667):66–74, April 2004.

[107] Nguyen Xuan Vinh et al. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *J. Mach. Learn. Res.*, 11:2837–2854, December 2010.

[108] Silke Wagner and Dorothea Wagner. Comparing clusterings- an overview. Technical report, Universität Karlsruhe (TH), 2007.

[109] Yi Wang, Henry Leung, S.M. Yiu, and Francis Chin. Metacluster 5.0: A two-round binning approach for metagenomic data for low-abundance species in a noisy sample. In *Proceedings of the ECCB*, 2012. To appear.

[110] Yi Wang, Henry C. Leung, S. M. Yiu, and Francis Y. Chin. MetaCluster 4.0: A Novel Binning Algorithm for NGS Reads and Huge Number of Species. *Journal of computational biology : a journal of computational molecular cell biology*, 19(2):241–249, 2012.

[111] Joe H. Ward. Hierarchical Grouping to Optimize an Objective Function. *Journal of the American Statistical Association*, 58(301):236–244, March 1963.

[112] Rene L. Warren, Granger G. Sutton, Steven J. M. Jones, and Robert A. Holt. Assembling millions of short DNA sequences using SSAKE. *Bioinformatics*, 23(4):500–501, February 2007.

[113] M.C. Wendl and R.H. Waterston. Generalized gap model for bacterial artificial chromosome clone fingerprint mapping and shotgun sequencing. *Genome Res*, 12(1):19431949, 2002.

[114] D. L. Wheeler, T. Barrett, D. A. Benson, S. H. Bryant, K. Canese, V. Chetvernin, D. M. Church, M. Dicuccio, R. Edgar, S. Federhen, L. Y. Geer, Y. Kapustin, O. Khovayko, D. Landsman, D. J. Lipman, T. L. Madden, D. R. Maglott, J. Ostell, V. Miller, K. D. Pruitt, G. D. Schuler, E. Sequeira, S. T. Sherry, K. Sirotkin, A. Souvorov, G. Starchenko, R. L. Tatusov, T. A. Tatusova, L. Wagner, and E. Yaschenko. Database resources of the National Center for Biotechnology Information. *Nucleic Acids Research*, 35(Database issue), January 2007.

[115] James Robert White, Niranjan Nagarajan, and Mihai Pop. Statistical methods for detecting differentially abundant features in clinical metagenomic samples. *PLoS computational biology*, 5(4):e1000352+, April 2009.

[116] Tanja Woyke, Hanno Teeling, Natalia N. Ivanova, and et al. Symbiosis insights through metagenomic analysis of a microbial consortium. *Nature*, 443(7114):950–955, 2006.

[117] Dongying Wu, Sean C. Daugherty, Susan E. Van Aken, Grace H. Pai, Kisha L. Watkins, Hoda Khouri, Luke J. Tallon, Jennifer M. Zaborsky, Helen E. Dunbar, Phat L. Tran, Nancy A. Moran, and Jonathan A. Eisen. Metabolic Complementarity and Genomics of the Dual Bacterial Symbiosis of Sharpshooters. *PLoS Biol*, 4(6):e188+, June 2006.

[118] Tong T. Wu, Yi F. Chen, Trevor Hastie, Eric Sobel, and Kenneth Lange. Genome-wide association analysis by lasso penalized logistic regression. *Bioinformatics*, 25(6):714–721, March 2009.

[119] Yu-Wei Wu and Yuzhen Ye. A novel abundance-based algorithm for binning metagenomic sequences using *l*-tuples. In *Proceedings of the 14th annual international conference on Research in computational molecular biology*, RECOMB'10, pages 535–549. Springer, 2010.

[120] Bin Yang, Yu Peng, Henry Leung, and et al. Unsupervised binning of environmental genomic fragments based on an error robust selection of l-mers. *BMC Bioinformatics*, 11(Suppl 2):S5+, 2010.

[121] C. Yang, D. Mills, K. Mathee, Y. Wang, K. Jayachandran, M. Sikaroodi, P. Gillevet, J. Entry, and G. Narasimhan. An ecoinformatics tool for microbial community studies: supervised classification of amplicon length heterogeneity (ALH) profiles of 16S rRNA. *Journal of Microbiological Methods*, 65(1):49–62, 2006.

[122] Fang Yang, Xiaowei Zeng, Kang Ning, and et al. Saliva microbiomes distinguish caries-active from healthy human populations. *The ISME Journal*, 6(1):1–10, 2011.

[123] Yuzhen Ye. Identification and quantification of abundant species from pyrosequences of 16S rRNA by consensus alignment. *Proceedings. IEEE International Conference on Bioinformatics and Biomedicine*, 2010:153–157, February 2011.

[124] Gangman Yi, Michael R. Thon, and Sing-Hoi H. Sze. Supervised protein family classification and new family construction. *Journal of computational biology : a journal of computational molecular cell biology*, 19(8):957–967, August 2012.

[125] Daniel R. Zerbino and Ewan Birney. Velvet: Algorithms for de novo short read assembly using de Bruijn graphs. *Genome Research*, 18(5):821–829, May 2008.

[126] Aibin Zhan, Martin Hulk, Francisco Sylvester, Xiaoting Huang, Abisola A. Adebayo, Cathryn L. Abbott, Sarah J. Adamowicz, Daniel D. Heath, Melania E. Cristescu, and Hugh J. MacIsaac. High sensitivity of 454 pyrosequencing for detection of rare species in aquatic communities. *Methods in Ecology and Evolution*, 4(6):558–565, 2013.

[127] Qi Zhang, Eric Yi Liu, Abhishek Sarkar, and Wei Wang. Split-order distance for clustering and classification hierarchies. In *In proceeding of: Scientific and Statistical Database Management, 21st International Conference, SSDBM 2009*, 2009.

[128] Tong Zhang and Frank J. Oles. Text categorization based on regularized linear classification methods. *Information Retrieval*, 4:5–31, 2000.

[129] Peng Zhao, Guilherme Rocha, and Bin Yu. The composite absolute penalties family for grouped and hierarchical variable selection. *Annals of Statistic*, 37(6A):3468–3497, 2009.

[130] Fengfeng Zhou, Victor Olman, and Ying Xu. Barcodes for genomes and applications. *BMC Bioinformatics*, 9(1):546+, 2008.