Czech Technical University in Prague

Faculty of Electrical Engineering

Department of Computer Science

Doctoral thesis

Efficient genome similarity estimation for learning from sequencing data

Supervisor:	prof. Ing. Filip Železný, Ph.D.
Programme:	Electrical Engineering and Information Technology (P2612)
Branch:	Artificial Intelligence and Biocybernetics (3902V035)

Friday 10th March, 2023 **Petr Ryšavý**

Abstract

Sequence assembly is one of the most common tasks in bioinformatics. Its goal is to produce estimates of the DNA sequence of an organism only by knowing short substrings of this sequence. The sequence assembly field is mature; therefore, it is possible to get reasonable estimates quickly. However, the problem itself is known to be NP-hard, and some instances are not efficiently solvable; some are unsolvable at all.
This work presents several methods for capturing similarities in sequencing data. We will avoid the difficulties of sequence assembly in the problems of phylogeny and sequence alignment. Firstly, we avoid the assembly step completely. Secondly, we assemble only the easy parts. Finally, we merge those two approaches into a single measure. We will provide many theoretical insights into the proposed measure, including a lower-bound view and the *p*-value, as well as experimental evaluation.
Sequence similarity can be used as an input in many machine-learning algorithms. In this thesis, we focus on unsupervised learning and namely on hierarchical clustering, which allows us to estimate evolutionary trees.
Next, we apply similar ideas to the problem of annotation of circular RNA with annotation terms, such as, for example, gene ontology terms. Further, we discuss the possibility of incorporating the RNA-Seq data into the annotation problem and, thus, provide further methods for learning on different sequencing data.
Keywords: genomic sequences, read bags, sequence assembly, sequence similarity, circular RNA, annotation, Monge-Elkan distance, Levenshtein distance

Abstrakt

Skládání sekvencí je jeden z nejběžnějších úkonů v bioinformatice. Cílem je získat odhady DNA sekvence organismu v situaci, kdy známe pouze krátké podřetězce této sekvence. Obor skládání sekvencí je vyspělý, takže je možné získat rozumné odhady brzy. Nicméně problém jako takový je NP-těžký a některé jeho instance nelze řešit efektivně, některé jsou dokonce neřešitelné.
Tato práce prezentuje několik metod pro zachycení podobností v sekvenačních datech. Budeme se snažit vyhnout problémům se skládání sekvencí v kontextu fylogenetiky a zarovnávání sekvencí. Nejprve vynecháme skládání sekvencí zcela. V druhém kroku budeme skládat pouze jednoduché části sekvencí a nakonec tyto dvě motody spojíme do jedné. Poskytneme teoretické náhledy do řešených problémů, včetně dolní meze, p-hodnoty a experimentální evaluace.
Podobnost sekvencí může být použita jako vstup v mnoha algoritmech strojového učení. V této práci se zaměříme na nesupervizované učení, jmenovitě na hierarchické shlukování, které umožňuje odhadnout evoluční stromy.
V konci práce aplikujeme podobné myšlenky na problém anotace cirkulárních RNA termy, jako jsou například termy genových ontologií. Dále budeme diskutovat možnosti zapojení RNA-Seq dat do problému anotace, což nám pak umožní vytvořit další metodu pro učení na různých sekvenačních datech.
Klíčová slova: genomické sekvence, multimnožina čtení, skládání sekvencí, podobnost sekvencí, cirkulární RNA, annotatace, Monge-Elkanova vzdálenost, Levenshteinova vzdálenost

Acknowledgements

I am very thankful to three major groups who helped me throughout my doctoral studies and with this dissertation — those who helped me directly or indirectly.

Those who helped me directly were especially those who supervised me any time in the past, especially Filip Železný, who led the thesis, consulted with me, and made the IDA group such a nice and comfortable place to work in. I am also thankful to Jiří Kléma, who supervised the research in Chapter 14. But I had the possibility to learn from others - Radomír Černoch motivated me to go for a doctoral program when I was writing a bachelor's thesis under his supervision. Also, Greg Hamerly, with whom I could work during my master's studies, and Karel Drbal for insights into cytometry problems. I am thankful to my current and former colleagues, especially Gustav Šír, Martin Svatoš, and Karel Horák, for having a pleasant atmosphere in the lab and for the possibility to talk about the challenges. My thanks go to Ondřej Kuželka for pointing me to the idea that I should try to find an approach to calculate the p-value.

The second group of people helped me indirectly. This includes especially my parents, Petr a Marie Ryšavých. I am very thankful for their support and for having the possibility to think about ”what I can do to improve my future” instead of ”what I need to do to improve my future,” and for teaching me that there is no work that one should not try doing himself. Very few things teach patience as much as situations when one says for the tenth time that ”that darned pipe does not fit.” I am also helpful for the support my girlfriend, Kačka, gave me. Thank you for being such a nice person, always helpful and understanding.

Among those who helped me indirectly, I also count many friends. Far too many ideas in this thesis popped out to me either while I was canoeing and exercising in USK KR in Malá Chuchle, while I was cross-country skiing in the Giant Mountains region, or while I was on a train to the cottage of Lokomotiva Nymburk in Vrchlabí. It sounds like a little, but I have no idea how I would clear my head otherwise, and I doubt that any time spent in front of a computer screen would be a sufficient substitute. Therefore, my thanks go to Petr K., Ivo B., Zuzana K., Michal P., Martina V., and others. It would be much less fun without y'all. And thank you for asking me what I am working on.

The last group is those who helped me, but it is far too easy to forget about them. My thanks go to the anonymous reviewers of the papers we wrote, to people who gave me questions and talked to me at conferences, or who gave talks and lectures which had some connection to this work. Thank you!

The support of the following grants is highly appreciated: CZ.02.1.01/0.0/0.0/16_019/0000765 “Research Center for Informatics”, CESNET LM2015042, GA CR 14-21421S, GA CR 20-19162S, SGS17/189/OHK3/3T/13, and SGS20/178/OHK3/3T/13.

Contents

Glossary

bp base pair, a pair of two nucleotides connected by a hydrogen bound. 5, 7, 15, 18

pH potential of hydrogen, used to measure acidity or basicity of a substance. 3

A adenine. 1, 9, 22

C cytosine. 1, 9, 22

G guanine. 1, 9, 22

T thymine. 1, 9, 22

BLAST Basic Local Alignment Search Tool. 12, 22, 85

BLOSUM Blocks Substitution Matrix. 22

circGPA circRNA generating-polynomial annotator. vii, 113, 115, 117, 119, 122–124, 126, 130–133, 135, 138, 139, 158

circRNA circular RNA. 7, 18, 113–117, 120–124, 126–128, 130–133, 135, 145, 158

COVID-19 Coronavirus 2019. 7, 8, 137, 160

CRT cyclic reversible termination. 3, 5

DBG de Bruijn graph. 32, 33

DNA deoxyribonucleic acid. iii, 1–4, 6, 7, 9, 10, 14–17, 25, 27, 33, 37, 38, 69, 74, 91, 94, 107, 112, 120, 136, 137, 141, 143–147, 150, 153, 155, 156, 158, 160–162

EBI European Bioinformatics Institute. 6

EMBL European Molecular Biology Laboratory. 6

ENA European Nucleotide Archive. 6, 69, 93, 107

FSWM Filtered Spaced Word Matches. 28, 30

GO Gene Ontology. 120, 122, 130

i.i.d. independent and identically distributed. 14, 17, 29, 39, 46, 53, 62, 67, 93, 107

miRNA micro RNA. 1, 113–122, 125, 128, 130–132, 153

mRNA messenger RNA. 1, 113–122, 125, 128, 130, 132, 133, 150

NCBI National Center for Biotechnology Information. 6, 124

NGS next-generation sequencing. 2, 4, 6, 7, 13, 14, 21, 24, 29, 30, 32, 141, 144, 154

OLC overlap layout consesnus. 32, 33

PAM Point Accepted Mutation. 22

PASDP Partially-Assembled-Sequences Distance Problem. 35, 36

PCR Polymerase Chain Reduction. 4, 160, 161

RNA ribonucleic acid. ix, x, 1, 4, 6, 111, 113–115, 120, 122, 124–126, 146, 151, 153, 160

RNA-seq RNA sequencing. 4, 8, 18, 31, 113, 135, 144, 146, 153, 156, 158, 162

SBS sequencing by synthesis. 3, 5

scRNA-seq single-cell RNA sequencing. 4

SNA single nucleotide addition. 3, 5

SNP Single Nucleotide Polymorphism. 6, 156

STAR Spliced Transcripts Alignment to a Reference. 4, 146

TSP traveling salesman problem. 32

UPGMA Unweighted Pair Group Method with Arithmetic Mean. v, 7, 24–27, 92

WGS whole-genome sequencing. 3

WPGMA Weighted Pair Group Method with Arithmetic Mean. 24

Frequently Used Symbols

Symbol	Explanation	Reference
$\mathbb{N}$	Natural numbers	
$\mathbb{R}$	Real numbers	
E	Expected value	
Σ	Alphabet	Chapter 2
A, B	Sequence (string)	Chapter 2
A_i	Nucleotide at position i in A	Chapter 2
A_i^j	Substring of A that starts at position i and ends at position j	Chapter 2
$\|A\|$	Length of sequence A	Chapter 2
$\mathcal{R}_A$	Set of all possible reads for a sequence	Definition 2.12
a, b	Read	Definition 2.12
R_A, R_B	Read bag	Definition 2.13
α, β	Contig	Definition 2.16
C_A, C_B	Contig set	Definition 2.17
T_A, T_B	Tuple of a read bag and a contig set	Definition 2.18
$\tilde{A}$, $\tilde{B}$	Assembled estimate of a sequence	Chapter 4
$\overline{A}$	Sequence complementary to A	Definition 2.2
$\underleftarrow{A}$	Reversed A	Definition 2.2
$\underleftarrow{\overline{A}}$	Reversed complement to A	Definition 2.2
$\alpha_{:}$, $\beta_{:}$	A substring of a contig	Section 8.1
l	Read length	Definition 2.12
c	Coverage (average over the sequence)	2.14
$c(i)$	Coverage at position i	2.14
c_m	Maximum per-base coverage	Definition 7.3
dist	Levenshtein distance	Definition 2.3
dist_q	q-gram distance	Definition 2.10
$\overline{\mathrm{dist}}$	Post-normalized Levenshtein distance	Definition 2.5
d	A distance measure or a distance value, depending on the context	
Dist	A distance on sets/bags	
$\mathsf{Dist}_{\mathsf{ME}}$	Monge-Elkan distance	Equation (5.1)
$\mathsf{dist}_{\mathsf{rS}}$	Distance used to search for a read in a string	Equation (7.1)
$\mathsf{dist}_{\mathsf{cR}}$	Distance between a read and a contig set. Other symbols might be used as well: r, c, R, C, T stand for a read, a contig, a read bag, a contig set, a tuple, respectively.	
U	Universe of elements	Definition 2.7
ω	An event	Chapter 14.5
Ω	The set of all events	Chapter 14.5

Symbol	**Explanation**	**Reference**
Pref	Set of all prefixes	Definition 2.19
Suff	Set of all suffixes	Definition 2.19
Subseq	Set of all substrings	Definition 2.19
$\mathsf{Overlaps}$	Set of all possible overlaps	Definition 2.20
$\mathsf{overlap}$	Overlap between two contigs	Equation (8.1)
$\mathbf{Q}_q$	q-gram profile	Definition 2.9
B_a	The closest substring of read b in sequence B	Equation (7.1)
t	Number of penalty-free gaps at a margin	Equation (5.4)
genpoly	Generating polynomial	Definition 6.3
$\lvert\mu\rvert$	Number of miRNAs	Section 14.1
$\lvert m\rvert$	Number of mRNAs	Section 14.1
$\mathbf{a}^{\mu,c}$	Edges vector between a circRNA and miRNAs	Section 14.1
$\mathbf{A}^{m,\mu}$	Adjacency matrix between miRNAs and mRNAs	Section 14.1
$\mathbf{g}^{m}$	Annotation of mRNAs	Section 14.1
$\mathbf{g}^{\mu}$	Annotation of miRNAs	Section 14.1
s	Statistic	Equation (14.1)

Chapter 1 Introduction

In living organisms, the molecule that stores the cookbook for life is the *deoxyribonucleic acid*, DNA in short. This molecule contains all information needed for cell replication, development, and function. Any DNA molecule consists of three primary ingredients. Deoxyribose and fosfate group form a chain on which nucleotides bound. The nucleotides carry genetic information. There are four of them - adenine (A), cytosine (C), guanine (G), and thymine (T).

Each DNA molecule consists of two complementary strands. The orientation of deoxyribose in the strand determines its orientation. We call the respective ends of the strand 3' end and 5' end based on the bonds of the deoxyribose. The complementary strands are oriented in reverse directions and are connected with peptide bonds between the nucleotides. The nucleotides in both strands are not connected arbitrarily. Adenine pairs with thymine, and cytosine has a peptide bond with guanine. Other combinations are unlikely but not impossible [131]. A complete DNA structure is visualized in Figure 1.1.

In 1869, *Friedrich Miescher* detected nucleotides for the first time [59]. He extracted nuclein from leucocytes obtained from pus in bandages. Later, Miescher was able to produce nuclein in larger amounts and quality from salmon sperm. However, it took another 75 years to prove that DNA is responsible for the transmission of genetic information [13]. A well-known discovery was made in 1953 when Watson and Crick published a three-dimensional model of DNA double helix structure [302]. This discovery was later rewarded with a Nobel prize.

Further research has shown that the DNA molecule is not used directly to synthesize proteins. Instead, another molecule, the *ribonucleic acid* (or RNA), is used as a *messenger* that carries the information stored in DNA to ribosomes where the proteins are synthesized according to the order of nucleotides. The nucleotides are read by triplets, known as *codons*, where each of the 64 possible triplets (called codons) codes one of the 21 *amino-acids*, a start codon, or a stop codon. The first step when a part of DNA, called *gene*, is used as a template for RNA synthesis is called *transcription*, the ribosomal formation of proteins is called *translation*. This whole process is sometimes called *the central dogma of molecular biology* [58].

The DNA can be understood as hardware that defines what can be synthesized in a cell. Similarly to computers, the core of the cell capabilities is not the hardware but the software, in the cell terminology called the *gene expression*. The DNA molecule is structured and split into many genes, some coding for proteins. Not all proteins are synthesized at every moment. Some proteins regulate the translation of genes. Other regulation mechanisms include micro RNA (miRNA) silencing when a short miRNAs marks a messenger RNA (mRNA) for degradation [34]. The miRNAs are, on the contrary, *sponged* [215] by a circular RNA (a product of mRNA splicing [80]).

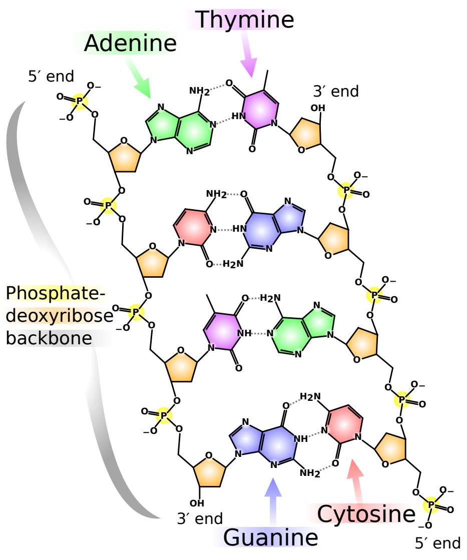


Figure 1.1: Structure of a DNA molecule. [By Madprime [CC0], via Wikimedia Commons, `https://commons.wikimedia.org/wiki/File:DNA_chemical_structure.svg`]

1.1 Genome Sequencing

It became clear that knowledge and understanding of the DNA sequence may give the human power to understand life, improve health care [74], and prevent some diseases before they start to develop [262, 219]. Therefore, a great effort was put into the development of sequencing. *Frederick Sanger* was the first one who was able to read the genetic code indirectly. Sanger identified the insulin protein sequence in 1955 [250].

The first methods to directly sequence DNA were developed in the 1970s. In 1977, Frederick Sanger developed a chain-terminating method [251]. During replication, a modified dideoxynucleotide is inserted into DNA. This modified dideoxynucleotide prevents DNA polymerase from replicating the sequence further. As a result, we obtain a collection of sequences of different lengths terminated by a known symbol. Modifications to the original Sanger's method rely on fluorescent chain terminators. The electrophoresis process then separates the sequences by length, allowing us to determine the correspondence between nucleotides and positions in the sequence [114].

Sanger sequencing was the leading method between the 1980s and mid-2000s. Its advantages include high accuracy and the ability to read sequences of lengths up to one thousand nucleotides. For those reasons, it is still used in smaller projects and to validate the results of newer methods. The higher cost per nucleotide, however, outweighs the advantages.

A genome is usually much longer than the limited sequence length that can be read by Sanger sequencing. Therefore, the idea of *shotgun sequencing* was proposed [274, 12]. Instead of reading the whole sequence at once, the DNA is broken randomly into small pieces. After sequencing, we call those small pieces *reads*. To reconstruct the original sequence, we have to clone DNA several times to provide some overlaps between reads. The reconstruction process of the original sequence is called the *assembly* and was first proposed in 1979 [274]. Due to the scale of the data, assembly needs to be done in computers.

The shotgun sequencing idea persisted into newer methods called *next-generation sequencing* (NGS) [107]. This group is characterized by shorter reads, a slightly higher error rate than Sanger sequencing, and a much lower cost at the order of five magnitudes.

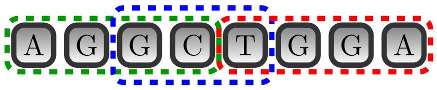


Figure 1.2: An illustration of shotgun sequencing. Long sequence AGGCTGGA is represented by three reads AGGC, TGGA, and GCT. The original sequence is reconstructed in silico.

The reads are sequenced in parallel, and most of the work is done *in silico*.

The first commercially successful system was *Roche 454* [184]. This sequencer was based on parallel pyrosequencing and was discontinued in 2013. The most popular technology nowadays is owned by *Illumina* (formerly Solexa). The Illumina sequencers are based on *sequencing by synthesis* (SBS).

The SBS methods rely on attaching known nucleotides to a single strand during the synthesis of the second DNA strand by the DNA polymerase. In 454 sequencing, we add only one nucleotide to the sample. The nucleotide is modified so that it emits light when it bonds to the strand. Because we know which nucleotide is used in the current cycle, we know which nucleotide was added to the strand. By repeating this procedure, we can synthesize the second strand and, at the same time, detect all nucleotides. *Ion Torrent* technology uses a similar single-nucleotide addition (SNA) approach; however, instead of light emission, the changes in pH are measured.

Illumina's approach belongs to the *cyclic reversible termination* CRT group of SBS approaches. During the synthesis of the second strand, the nucleotides are marked and modified so that no new nucleotide can attach to them. In each cycle, one nucleotide hybridizes to the complementary base. Next, imagining is done with a laser. By this procedure, the markers on the newly added nucleotides are read. The last step of the cycle consists of the removal of the markers blocking further synthesis.

Sequencing by ligation approaches are competing to SBS. In each step, probes of one or two nucleotides together with a degenerate bases anchor are added to the chain. They hybridize to the strand that is read. Then the ligation and imagining steps follow. After several cycles, we know nucleotides (or dinucleotides) with gaps of a known size between them. Therefore, the anchors are removed, and a new cycle is started with a different offset anchor. Sequencing by ligation includes SOLiD technology [291] and Complete Genomics technology [75].

To make the in silico genome reconstruction possible, especially in the whole-genome sequencing (WGS) projects, there is a need for longer reads. Therefore, the *long-read sequencing* methods were proposed. *Pacific Biosciences* introduced a real-time long-read sequencing that can read a cyclic molecule. This technology has a relatively high error of 13 %, which can be decreased by reading a read several times. *Oxford Nanopore* proposed a competing method [46, 81] where a single molecule is passed through a pore in a protein molecule. By measuring the changes in the electric current flow, the machine detects nucleotide k-mers as they flow through the pore.

Illumina's response to the long-read sequencing was the synthetic long-read sequencing approach. The long reads are separated one from the other. A short barcode is appended to the fragments when a long read is sheared into fragments. The classical sequencing technologies then read short reads and group them by the barcode. As a result, the long reads can be assembled in silico. The advantage comes from the fact that no new equipment is needed, only chemicals that can append the barcodes. However, the reconstruction in a computer cannot overcome some disadvantages of the classical short-read approach. A brief overview of the sequencing technologies we mentioned is in Table 1.1.

There are other methods that undergo active development in order to increase read length, decrease the time needed to provide results, and decrease costs. The nanopore and PacBio methods are the first methods from the group called *third-generation sequencing* [255, 160]. Other methods exploit electric properties of DNA molecules [312], spectrometry [79], electron microscopy [19], or extend the existing methods.

1.2 Gene Expression Analysis and Other Alternatives to the Next-Generation Sequencing

Besides NGS, there are several other methods that either compete with or complement NGS. The most relevant to us are DNA microarrays [283] and RNA sequencing (RNA-seq) [300]. Both are used for the *gene expression analysis*. The gene expression analysis aims to find out which genes are transcribed to RNA. This is usually done by counting the quantities of RNA molecules in a biological sample. By knowing the counts of each gene, it is possible to find out which genes are actively used in a cell to synthesize proteins and which are suppressed. We call the active genes expressed. Reconstruction of the original gene sequences from the RNA transcripts allows detection of mutations [146], single nucleotide polymorphism [228], alternative gene splicing [142, 247, 174], and others. It can be said that DNA encodes what is possible to happen in a cell and the gene expression shows what really happens.

The *DNA microarrays* [283] consist of several thousand probes on a chip. The probes are selected to cover a great variety of sequences typical for particular genes, mutations, or other subsequences that we are interested in. If an RNA sample contains a sequence that matches a probe, it hybridizes to the probe. The hybridization is connected with a signal that can be measured automatically, for example, by the emission of light.

The *RNA-seq* [300] is related to the NGS methods. The RNA molecules are reverse transcribed to DNA. Then the DNA is sequenced using the standard approaches. Further, reads can be mapped to the genes to obtain expression levels. One of the typical pipelines includes STAR [71] for read mapping, HTSeq-count [11] for abundance quantification, and DeSeq2 [182] for differentially expressed genes identification. Nowadays, the resolution of RNA sequencing is being improved up to single cells. With *single-cell RNA sequencing* (scRNA-seq) [41], the current technology can read expression levels of individual cells, which are then clustered in silico to increase the coverage.

Cytometry is an older alternative to the RNA-seq. In principle, we are interested in the same question, i.e., which genes are expressed. However, the technology is entirely different. Instead of broad transcript sequencing, we measure levels of proteins directly in cytometry. The cells *flow* through a machine that is able to detect amounts for approximately 20 selected proteins, producing a data matrix with tens to hundreds of thousand or more rows, each with the expression profile of a cell. On such a matrix, data reduction techniques are applied so that it is possible to identify different cell populations in blood sample [4] and improve our understanding of diseases [47] or cancer [16, 6].

Besides DNA microarrays, there is another method for testing whether a sample contains a pre-selected genetic sequence. The name of the method is *polymerase chain reaction* (PCR in short) [203], and it became well known in the recent covid pandemics [280]. By an iterative procedure, a sample with initial DNA sequence is amplified so that the sequence is copied in large quantities. Next, the sample can be tested for an infectious disease [292], cancerous mutations [23], or whether a potential parent is a carrier of an inheritable disease [248]. The invention of PCR by Kary Mullis was rewarded with a Nobel Prize in Chemistry in 1993. The main advantage of PCR is its cost-efficiency and speed of the process. The cloning is often done in 30 to 40 cycles, causing an exponential growth in the number of samples, which allows the results to be available in just a few hours. The method also allows abundance quantification by

Platform	Technology	Number of reads	Read length	Runtime	Error rate	Cost per Gbp
Sanger	chain termination	–	400 to 900bp	$\leq$ 3 hours	$\leq 0.1\,\%$	\$2,400,000
SOLiD	sequencing by ligation	up to $1.4 \cdot 10^9$	up to 75	6 to 10 days	$\leq 0.1\,\%$	cca \$100
Illumina	SBS: CRT	millions to billions	up to 300	by read length	$\leq 1\,\% - 0.1\,\%$	mostly bellow \$100
454	SBS : SNA	up to 1 million	up to 1000	$\leq$ 1 day	$1\,\%$	\$10,000 to \$40,000
Ion	SBS : SNA	up to 80 million	up to 400	hours	$1\,\%$	around \$1,000
PacBio	single-molecule	hundreds of thousands	up to 20,000	hours	$13\,\%$	\$1,000
Oxford Nanopore	single-molecule	$>$ 100,000	up to 200,000	up to 2 days	$12\,\%$	\$750

Table 1.1: A brief overview of the sequencing technologies. This table summarizes a complete list that can be found in [107]. The manufacturers usually provide several variants of their technology that differ in cost and throughput. The methods usually have a different target group of projects and sequenced organisms.

counting the number of cycles needed for the test to show positive [32]. One disadvantage is that the possibilities to detect multiple probes are not limitless [191].

There are many other related methods for studying cell processes. For example, ChIP-Seq [217] is a method to measure how proteins interact with DNA, which in turn influences gene expression. Similarly, CLIP-Seq [286] and RIP-Seq [135] are used to find interactions between RNA and proteins.

1.3 Sequence Databases

There are huge amounts of sequencing data produced each day. Genome sequencing is one of the sources producing most of the big data in the world [276]. The cost of sequencing per base pair decreases faster than expected from Moore's Law [117]. Many researchers publish sequenced DNA data into public databases, where the amount of data grows exponentially [48].

One of the most important sequence databases is the *European Nucleotide Archive* (ENA) [166], which is operated by the European Molecular Biology Laboratory's European Bioinformatics Institute (EMBL-EBI) in the United Kingdom. The data can be downloaded online from `https://www.ebi.ac.uk/ena`. The American counterpart to the ENA archive is operated by the *National Center for Biotechnology Information* (NCBI) [306]. The NCBI database is available on `https://www.ncbi.nlm.nih.gov/`.

Development of the NGS technologies is connected with the *Human Genome Project* [159]. The project started in 1990 in the United States of America. The aim of the project was to identify the whole sequence of the human genome. A first draft was published in 2000, and almost the entire genome was published in 2003. The goal was to sequence only the euchromatic regions of the genome that are lightly packed parts of the chromosomes. The euchromatic regions are easily accessible by the RNA polymerase. The remaining 8 % of the genome was not sequenced. Although the project is finished, the human genome sequencing process continues to fill missing regions and improve accuracy.

The knowledge of a consensus human genome cannot help in some applications, including single nucleotide polymorphisms (SNPs) [279], variant calling [311], and others. Therefore, a more ambitious project was proposed. The goal of the *1000 genomes project* [1] is to sequence more than one thousand individuals from different populations. The goal is to provide some guidance on differences between human genomes based on ethnic and geographic location. The authors of the project made the data available online on `http://www.internationalgenome.org/`.

The Human Genome Project is the most remarkable of the sequencing projects since it was the first one. Nevertheless, further, more ambitious projects, such as the aforementioned 1000 genomes project, arise and will arise. The goal of the 100,000 genomes project [102] was to sequence patients in the British health care system with emphasis on the medical history of the sequenced patients. The number of samples to sequence was reached after five years in 2018. Concurrently, an even more broad project, named Earth BioGenome [169], was started. Its goal is to sequence as many living organisms DNA as possible to understand biodiversity better.

The DNA sequence databases are not the only sources that are available. For example, protein data are available in the UniProt [284] database. For DNA data, the most appropriate file format used is FASTA. For read data, the FASTQ format is used.

Not only the sequence in the database is important. Therefore, other databases annotate the sequences, genes, and molecules with function. The most important are ontology-based databases. An *ontology* is a directed acyclic graph that stores a hierarchy of annotations from the most general to specific ones. For example, the genes are annotated by the Gene Ontology database [56]; proteins are annotated by the Protein Ontology database [206].

1.4 Phylogenetics

The knowledge of a DNA sequence may be helpful in many applications. For example, many inheritable diseases are encoded in the genome, namely several cancer types [99]. A similar example is connected with the p53 gene, which acts as a tumor suppressor, and contains a mutation in more than 50 % cancerous cells [232]. Therefore, the knowledge and understanding of the genome may allow us to predict diseases even before the first symptoms are visible [305].

Another popular application is the reconstruction of evolutionary trees. Many species split in two during evolution due to the different environmental conditions influencing the separated populations. If we pick several organisms, it may be helpful to draw a hypothetical tree that shows the history of such events. Such a tree is called a *phylogenetic tree*. It is usually a rooted tree, where the root depicts a hypothetical common ancestor of the species. The leaves of the tree belong to clustered organisms, and the lengths of the edges represent time.

Traditionally, such trees were built manually by observing simple characteristics of the organisms, such as the number of legs. However, the knowledge of DNA sequences makes the task much more exact. From DNA sequences, it is possible to estimate the time that has passed since the common ancestor of the organisms. If we apply a hierarchical clustering algorithm on such distances, we get a tree called a dendrogram, which in our case, serves as a phylogenetic tree. The standard approaches include the neighbor-joining algorithm [249] and UPGMA [269].

One of the most famous applications of phylogenetic trees arose with the Ebola virus outbreak in 2014. Due to the high mortality and a rapid outbreak in Africa, it was not possible to sequence the DNA of the virus in the early stages of the epidemic. Also, due to the rapid outbreak, it was not clear where the virus originated from. Therefore, researchers sequenced 99 virus genomes and used an inferred phylogenetic tree to discover that the outbreak was caused by a single transmission from a non-human source to a human [104]. A similar application is mentioned with respect to the Salmonella bacteria in [157].

Another famous example of the usage of sequencing in phylogeny is connected with the panda bear. The panda is an animal visually similar to the bear. However, the panda misses some essential features of bears. For example, the panda does not hibernate in winter, similar to the raccoon. Biologists had disagreed on whether the panda is a bear, a raccoon, or a separate species family for more than one hundred years. In 1985, authors of paper [211] have shown by sequencing 500, 000bp of the genome that the panda is, in fact, a bear and not a raccoon.

Other usages of phylogenetic trees include the study of the migration of human tribes in the past [112] or forensic applications [263]. Phylogenetic studies were irreplaceable over the COVID-19 pandemics. A common source for media and other researchers was based on the Gisaid initiative data [261] and provided on address `https://nextstrain.org/ncov/gisaid/global` [113]. The provided data were used to generate the phylogenetic tree in Figure 1.3.

1.5 Summary of Contributions

The main contribution of the thesis is a collection of methods that facilitate unsupervised learning from NGS data. At the heart of the methodology are novel algorithms for

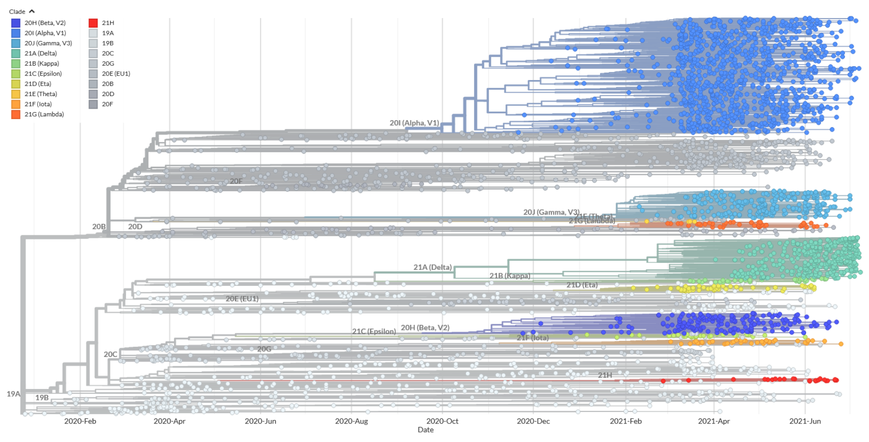


Figure 1.3: A recent example of a phylogenetic tree built after the COVID-19 outbreak. The tree is used to study mutation, evolution, and spread of the virus. The image was produced by the nextstrain tool [113].

efficient estimation of similarities between genomes from sequencing data formalized as distance functions on sequences. The estimated distances are readily valuable in tasks such as clustering and phylogenetic tree reconstruction. The main challenge in this endeavor is to avoid the problems entailed by the traditional approach consisting of de novo genome reconstruction and subsequent sequence alignment. We tackle this by proposing a novel approach for calculating the distance directly from read data or contig data if the latter is available. As a next logical step, we combine both read-level and contig-level methods into a unified method. We will compare the proposed method with the mentioned conventional approaches. The proposed method will be analyzed not only experimentally, but some theoretical insights will also be provided; these include the formal derivation of the p-value of the proposed distance and an algorithm for its calculation using an original approach based on generating polynomials.

Towards the end of the study, we focus on similarity calculations in a different context, particularly circular circRNAs. Here, the distances will be captured by the circRNA — annotation term relationship. Later, we will show how to extend the annotations by incorporating sequencing data, which will, in this case, originate from the RNA-seq. We provide a theoretical analysis in a fashion analogical to the first contribution and adopt some of the core concepts of the latter, such as the p-value derivation using generating polynomials.

Chapter 2 Basic Concepts

We will study sequences that describe the DNA code. Therefore, we restrict ourselves only to sequences[1] over alphabet $\Sigma = \{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{T}\}$. Each of the symbols in the alphabet represents one of the possible nucleotides - adenine, cytosine, guanine, and thymine, respectively. We consider only sequences with a finite, however unbounded, length, i.e., if A is a sequence, then

$$A \in \bigcup_{i=1}^{\infty} \Sigma^i.$$

By $|A|$, we denote the length of sequence A. By A_i we denote the i-th character in sequence A for $i \in \{1, 2, \dots, |A|\}$, i.e., $A = A_1 A_2 \cdots A_{|A|}$. By A_i^j, we will denote a substring of A that starts at i-th position and ends at j-th position. Formally, $A_i^j = A_i A_{i+1} \cdots A_j$.

We have already mentioned that not all pairings between nucleotides on the complementary strands are common — adenine bonds to thymine and cytosine bonds to guanine. Therefore, we define the concept of complementary nucleotides.

Definition 2.1 (Complementary nucleotide) *Let $\Sigma = \{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{T}\}$. The **complement** of $x \in \Sigma$ (denoted $\overline{x}$) is:*

$$\overline{\mathsf{A}} = \mathsf{T}, \qquad \overline{\mathsf{C}} = \mathsf{G}, \qquad \overline{\mathsf{G}} = \mathsf{C}, \qquad \overline{\mathsf{T}} = \mathsf{A}.$$

Having this pairing between nucleotides, we can define a complementary sequence.

Definition 2.2 (Complementary and reversed sequence) *Let $A = A_1 A_2 \cdots A_{|A|}$ be a sequence over alphabet $\Sigma = \{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{T}\}$. We define its **complement** as*

$$\overline{A} = \overline{A}_1 \overline{A}_2 \cdots \overline{A}_{|A|}. \tag{2.1}$$

*Further, we define the **reversed** sequence as*

$$\underleftarrow{A} = A_{|A|} \cdots A_2 A_1. \tag{2.2}$$

*The combination of those two operations is the **reversed complement**, which is*

$$\underleftarrow{\overline{A}} = \overline{A}_{|A|} \cdots \overline{A}_2 \overline{A}_1. \tag{2.3}$$

[1] In this work, we will use terms string and sequence for the same thing. The difference is only in the context - term sequence originates from the bioinformatics field. In contrast, term string comes from mathematics, automata, and related fields.

2.1 Distance between Two Sequences

If we consider two sequences A and B, we may be interested in knowing their distance. For our purposes, we will use the *Levenshtein distance* [167] (sometimes called the *edit distance*), which counts the minimum number of elementary operations needed to transform one sequence to the other.

Definition 2.3 (Levenshtein distance) *Let A and B be two sequences. Then the **Levenshtein distance** of A and B (denoted* $\text{dist}(A, B)$*) is the minimum number of single symbol edit operations* insert*,* delete*, and* substitute *needed to transform A to B.*

The Levenshtein distance approximates well the evolutionary distance of two organisms under the *molecular clock hypothesis* [329]. This hypothesis states that the rate of evolutionary changes is constant over time and for different species. In other words, the probability of mutation within a time unit is the same for all organisms studied in an experiment. For smaller-scale sequences (i.e., viruses or single genes in eucaryota), there are three common mutations in genes:

- *insertions*, when a new nucleotide is inserted into the sequence;
- *deletions*, when a single nucleotide is deleted from the sequence; and
- *substitutions*, when a single nucleotide is replaced with another one.

Those mutations are at a single point and correspond with the edit operations considered in the Levenshtein distance. Substitutions are more common than insertions and deletions (together called *indels*) because some substitutions lead to the synthesis of precisely the same protein, i.e., do not change the meaning of the genetic code. Even if a substitution leads to a different amino acid, the protein may still keep its function. On the contrary, an indel may shift the genetic code (which is based on triplets of nucleotides, called *codons*). As a result, the indels are much less often viable. However, [30] states that the indels are the mutations that cause most of the differences between organisms, assuming they do not cause death.

For larger-scale genomes, we have to consider two additional edit operations:

- *transpositions*, when a long block of DNA moves from one position to another; and
- *inversions*, when a piece of DNA is replaced by its reversed complement, which is taken from the complementary strand.

Insertions and deletions may arise not only at a single point but for longer blocks as well.

The Levenshtein distance is a metric, i.e., it fulfills the properties stated in the following definition.

Definition 2.4 (Metric) *A **metric** on set S is function $f : S \times S \mapsto \mathbb{R}$ such that for all $x, y, z \in S$ holds:*

1. $f(x, y) \geq 0$ *(non-negativity),*
2. $f(x, y) = f(y, x)$ *(symmetry),*
3. $f(x, y) = 0 \Leftrightarrow x = y$ *(identity condition),*
4. $f(x, z) \leq f(x, y) + f(y, z)$ *(triangle inequality).*

There are many alternatives to the Levenshtein distance. In computational biology, costs for indels and substitutions are often different, and their values are proportional to the probability of the corresponding changes to the genome [8]. In the model used in Definition 2.3, we assume that the cost per one gap is the same regardless of the gap length. Intuitively, we might say that there is some fixed cost for opening a gap, and then the cost of making the gap longer is much smaller. This idea brings us to the *affine gap penalty* [17], which can be calculated by a sum of the opening cost and a factor that grows linearly with the gap length. We will discuss modifications to the Levenshtein distance in Section 3.1.1.

A simple extension of the Levenshtein distance is the *Damerau-Levenshtein distance* [61], where transpositions of two adjacent characters are allowed. Damerau stated that those four edit operations correspond to more than 80 % of human spelling errors. If we allow transpositions of longer substrings, we obtain the *string block edit distance* [181], which better matches the similarity for longer genomes. Among the simpler distance measures, we mention the *Hamming distance* [115], which allows only substitutions.

Often, we require a distance measure to be normalized to the $[0, 1]$ interval. There are several ways how to normalize the Levenshtein distance. The normalization should be done so that the final measure is still a metric, as proposed in [193]. However, for our purposes, we will mention only the *post-normalized Levenshtein distance*, which is not a metric.

Definition 2.5 (Post-normalized Levenshtein distance) *Let A and B be two sequences. The* ***post-normalized Levenshtein distance*** *of A and B is*

$$\overline{\operatorname{dist}}(A, B) = \frac{\operatorname{dist}(A, B)}{\max\{|A|, |B|\}}. \tag{2.4}$$

This normalization is based on the following proposition, which guarantees that the resulting number is from the $[0, 1]$ interval. In this case, the resulting measure is not a metric. There are several ways to normalize the Levenshtein distance, so that the metric properties are met, for example, in [193]. In that paper, the Levenshtein distance is defined as a minimum of $W(A, B)/L(A, B)$, where $W(A, B)$ is the cost for an alignment of A and B, and $L(A, B)$ is the length of this alignment. Nevertheless, this approach has a cubic runtime, making it inapplicable to our settings.

Proposition 2.6 (Upper bound on the Levenshtein distance) *For any two sequences A and B holds that*

$$\operatorname{dist}(A, B) \leq \max\{|A|, |B|\}. \tag{2.5}$$

Proof WLOG assume that $|A| \geq |B|$. Then by $|B|$ substitutions and $|A| - |B|$ deletions, we can transform A to B. Therefore, the distance has to be smaller or equal $|B| + |A| - |B| = |A| = \max\{|A|, |B|\}$. Moreover, this bound is tight for $|\Sigma| \geq 2$. We can pick the sequences so that $A = x^{|A|}$ and $B = y^{|B|}$, where x and y are two different symbols from Σ. ■

2.2 Other Relevant Distance Measures

In this section, we mention other distance measures related to the thesis. Namely, we will mention the Monge-Elkan distance [200] and the q-gram distance [290].

2.2.1 Monge-Elkan Similarity

The Monge-Elkan similarity [200] was proposed to be used in the field of databases to search for duplicates. Imagine that we have a similarity function s, which compares two strings. When humans enter non-atomic fields, two strings may represent the same object. For example, „John Doe“ and „Doe, John“ are very likely to represent the same person, which may not be reflected by measure s.

Therefore, both inputs are split into two sets, S_1 and S_2, with the space as a delimiter. For each string in set S_1, we find the most similar string in S_2 and its similarity. The Monge-Elkan similarity is defined as an average of those similarities over all strings of S_1.

Definition 2.7 (Monge-Elkan similarity) *Let S_1, S_2 be two sets on universum U and s be a similarity measure on U, i.e., $s : U \times U \mapsto \mathbb{R}_0^+$. The **Monge-Elkan similarity** is defined as*

$$MES(S_1, S_2) = \frac{1}{|S_1|} \sum_{x \in S_1} \max_{y \in S_2} s(x, y). \tag{2.6}$$

Should we need to calculate a distance using Formula (2.6), we need to replace similarity measure s with a dissimilarity measure and maximum operator with minimum.

2.2.2 q-gram Distance

We will use the q-gram distance for faster calculations in an algorithm proposed in Chapter 5. The q-gram distance was formalized in [290], where it was used for the *approximate string matching problem* [207], where we look for approximate occurrences of a pattern in a long text. The distance stems from the following definitions.

Definition 2.8 (q-gram) *Let Σ be an alphabet. Then a **q-gram** is any string from Σ^q.*

Definition 2.9 (q-gram profile) *Let Σ be an alphabet, let A be a string over Σ, and let $q \in \mathbb{N}$. Let f be a bijection from all q-grams to the set $\{1, 2, \ldots, |\Sigma|^q\}$ (i.e., $f : \Sigma^q \mapsto \{1, 2, \ldots, |\Sigma|^q\}$). Then the **$q$-gram profile of** A, $\mathbf{Q}_q(A) \in (\mathbb{N}_0)^{|\Sigma|^q}$, is a vector, where for any q-gram a, $f(a)$-th field contains the number of occurrences of a in A as a substring.*

We assume that bijection f is fixed throughout the thesis. It is not important which of the possible $|\Sigma|^q!$ bijections we choose, but it needs to be the same whenever the q-gram distance is used.

Definition 2.10 (q-gram distance) *Let A, B be two strings over alphabet Σ and let $q \in \mathbb{N}$. The **q-gram distance** is the Manhattan distance of the sequence's q-gram profiles, i.e.,*

$$\text{dist}_q(A, B) = \|\mathbf{Q}_q(A) - \mathbf{Q}_q(B)\|_1. \tag{2.7}$$

The term q-gram comes from the approximate string matching field [207]. However, in bioinformatics, the term *k-mer* is often used, for example, in [316]. They both have the same meaning; the difference comes from the fact that the fields developed independently.

As stated in the following theorem, the q-gram distance is a lower bound on the Levenshtein distance. Paper [278] shows that two sequences sharing a large number of q-grams are likely to be close in the Levenshtein distance. The observation is largely supported by the success of the BLAST algorithm [9], which uses high scoring q-grams as seeds for the alignment. The choice of q depends on the expected sequence length; we follow recommendation of [207], where

$$q = \log_{|\Sigma|}(\mathsf{E}(|A|)). \tag{2.8}$$

The result in Formula (2.8) corresponds with the situation when each field in the q-gram profile contains one on average. I.e., the length of the q-gram profile ($|\Sigma|^q$) is equal to the expected sequence length.

Because the read length varies from tens to hundreds of symbols, we choose $q = 3$ when working with reads.

Proposition 2.11 ([239]) *For any two sequences A and B and $q = 3$, holds*

$$\operatorname{dist}(A, B) \geq \frac{1}{6} \operatorname{dist}_q(A, B). \tag{2.9}$$

Proof ([239]) Theorem 2.11 can be proven through mathematical induction on the Levenshtein distance between A and B. Suppose that $\operatorname{dist}(A, B) = d$.

- If $d = 0$, then both $\operatorname{dist}(A, B) = \operatorname{dist}_q(A, B) = 0$ and the inequality holds.
- Now suppose the bound holds for any two sequences with distance at most $d - 1$. There exists a sequence of d operations *insert*, *delete*, and *replace* that transforms A into B. Denote B' the result we obtain after applying the first $d - 1$ operations. Then $\operatorname{dist}(A, B') = \operatorname{dist}(A, B) - 1$. Consider the last operation and calculate the maximal value of $\operatorname{dist}_q(A, B) - \operatorname{dist}_q(A, B')$.

  In the case of insertion (or, vice versa, deletion), we obtain, in the worst case, instead of q-grams *abc*,*bcd*, q-grams abX,bXc,Xcd (or vice versa). The q-gram distance grows by at most 5. In the case of a mismatch, q-grams abX,bXc,Xcd are replaced by q-grams abY,bYc,Ycd. The q-gram distance grows at most by 6. Therefore, $\operatorname{dist}_q(A, B) - \operatorname{dist}_q(A, B') \leq 6$. Making the induction step results in

$$\begin{aligned} \operatorname{dist}(A, B) = 1 + \operatorname{dist}(A, B') &\geq 1 + \frac{1}{6} \operatorname{dist}_q(A, B') \\ &\geq 1 + \frac{1}{6} (\operatorname{dist}_q(A, B) - 6) = \frac{1}{6} \operatorname{dist}_q(A, B). \end{aligned}$$

The proof is then finished by the standard mathematical induction axiom. ■

2.3 Sequences in Sequencing

As we mentioned in the introduction, the NGS methods are not able to read long sequences. Instead, they read short substrings called *reads*. In this section, we define the key terms that are connected with the process of sequencing and further genome reconstruction.

Definition 2.12 (Read) *Let A be a sequence and $l \in \mathbb{N}$, such that*

$$l \ll |A|. \tag{2.10}$$

***Read** a is any approximate substring of A, $\overline{A}$, $\underleftarrow{A}$ or $\underleftarrow{\overline{A}}$ that has length l, where we allow a small number of insertions, deletions, and substitutions. By $\mathcal{R}_A$, we denote the **set of all reads** that can be obtained from sequence A.*

Throughout the text, we will assume that the length of reads is constant and the same for all reads. We will denote this constant l and call it the *read length*. Not for all sequencing technologies, the constant read length assumption is valid. However, the most common technology provided by Illumina follows this pattern.

Many sequencing technologies read sequences from the $5'$ end to the $3'$ end of a DNA strand.[2] If the direction of strands is known to us, we can restrict the reads to be substrings of A and $\underleftarrow{\overline{A}}$ only.

In many technologies, a read is sequenced from the two opposite ends of a DNA fragment. Those reads are called *paired-end* reads. During the wet lab work, only fragments of an approximately known small length are filtered out. Then the fragments are read from both ends. A common trick is based on sequencing fragments that are only a few base pairs shorter than $2l$. If we calculate the overlap of the two reads obtained from the same fragment, we can join them into a single one, which is longer.

A sequencing machine does not read a single read but many of them. A collection of reads will be called a *read bag*.

Definition 2.13 (Read bag) ***Read bag*** *R_A generated from sequence A is a multiset of $|R_A|$ reads drawn from a probabilistic distribution on $\mathcal{R}_A$.*

This definition of a read bag is too broad, so it matches the actual sequencing procedure. In theory, we usually restrict the read bag by some additional constraints so that the reads are easier to analyze.

- A common assumption is that the reads are drawn, i.i.d. from a distribution on $\mathcal{R}_A$. This assumption is not strictly valid; for example, when enzymes break a DNA molecule into fragments, one fragment likely starts where the other ends.
- The second usual assumption is that the reads are approximately uniformly distributed on the sequence. In the real world, the reads are not uniformly distributed, and this assumption causes a bias due to the wet-lab work noise [233]. However, many NGS algorithms work with this assumption, for example, to test for the existence of long repeating blocks, called *repeats*, or to assess the quality of an assembly [73, 190].

Besides the read length l, another critical term is connected with sequencing. The *per-base coverage* (alternatively called the *sequencing depth*)[3] for a particular base pair is the number of reads that span over this base pair. For position i, we will denote the *per-base coverage* $c(i)$. However, usually, we are not interested in the coverage at a single position but in the average *coverage*.

Definition 2.14 (Coverage) *Assume that A is a sequence, R_A a read bag, and l the read length. Then* ***coverage*** *c is*

$$c = \frac{|R_A| \cdot l}{|A|}. \tag{2.11}$$

Under the uniformly i.i.d. generated reads assumption, the coverage is a random variable, with the same expected value c for all base pairs except a few at the ends of the sequence. For simplicity, we adopt the assumption that coverage c is the same constant for all datasets in one experiment. The coverage is usually around 3 or 5 for low-quality sequencing. However, coverage over 30 is usually required for high-quality assemblies [267].

[2]This direction is not random; it is the direction in which the DNA polymerase synthesizes the second strand over the process of DNA replication.

[3]The term coverage is often used with many different meanings. Besides the per-base coverage and average coverage, it is sometimes used to quantify the percentage of a reference genome covered by sequenced reads [267].

2.4 Sequences in Assembly

The goal of *assembly* is to produce a putative sequence from the original reads. However, this is usually impossible for the reasons explained later in Section 2.6. Therefore, the sequence assembly is a multistep process. Following Occam's razor, the assembly can be formalized as a search for the *shortest common superstring*.

Definition 2.15 (Shortest common superstring problem) *Let* $\{S_1, S_2, \ldots, S_n\}$ *be a set of strings. Find the shortest string* S*, such that* $\forall i \in 1, 2, \ldots, n : S_i$ *is a substring of* S*.*

The complication is that the shortest common superstring problem is known to be NP-complete [185]. Because of the amounts of data and the fact that the reads are only approximate, we have to use heuristic approaches for assembly. Moreover, the exact assembly is often not possible. As a result, the assembly algorithms produce several putative sequences, which are eventually subsequences of the actual DNA sequence. Those output sequences are called *contigs*. They usually represent the parts that are easily reconstructed from a read bag.

Definition 2.16 (Contig) *The* ***contigs*** *are maximal, mutually non-intersecting sequences that are assembled from reads.*

The contigs are only approximate substrings of the original sequence because they may include some errors caused by the sequencing process. The assembly process is another source of errors. The contigs should be non-intersecting in the meaning of location in the original sequence. If two contigs intersect, a proper assembly algorithm should be able to detect this as an overlap and join the contigs into a single sequence.

For completeness, we include the original definition of a contig, which was stated in paper [275]:

> *A contig is a set of gel readings that are related to one another by overlap of their sequences. All gel readings belong to one and only one contig, and each contig contains at least one gel reading. The gel readings in a contig can be summed to form a contiguous consensus sequence and the length of this sequence is the length of the contig. [275]*

It is usually unknown for any pair of contigs, whether they come from the same strand. We also usually do not know their relative order and locations in the original sequence. Conversely, the information about the fact that the contigs are oriented from the $5'$ end to the $3'$ is preserved from the orientation of the reads.

Definition 2.17 (Contig set) *A* ***contig set*** *of sequence* A*, denoted* C_A*, is a set of all contigs assembled by an assembly algorithm from read bag* R_A*.*

For whole-genome studies, more than the contig data is needed. Therefore, the contigs have to be assembled into a single sequence. This is done via a process called *scaffolding*. Using long-read data or *mate-pairs*, it is possible to build a scaffold, order contigs, and identify gaps between them. So-called mate-pairs are a particular type of paired-end reads that are separated by a gap of an approximately known length, usually around 10 kbp. To sequence the mate-pairs, a different approach for sequencing has to be used, as described in [132]. Once the scaffold is built, the process of *gap filling* is used to fill as many blanks between the contigs with unmapped reads as possible. If the mate-pair data (or the long read data) are available at the assembly time, the assembly algorithms may do both assembly and scaffolding at once [221].

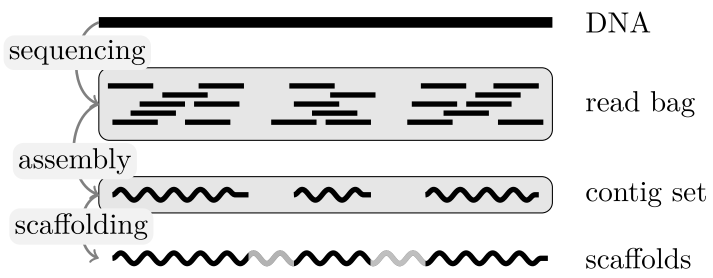


Figure 2.1: The types of strings that are connected with the sequencing and de-novo assembly process.

So far, we have talked about the *de novo* assembly. This is the case when a completely unknown genome is sequenced. However, in many cases, it is possible to reuse a genome of a related species or even another individual of the same species. In this case, we talk about the *mapping* assembly. We use a reference genome to order the reads [71]. From the sequence similarity, it is possible to match the reads with a location in the reference genome. From the counts of symbols at a particular position, it is possible to build a consensus sequence by considering only the most common symbol at each position. The mapping assembly is much easier than the de novo process. However, the reference genome does not need to be known. Also, knowledge of a reference genome allows us to do the sequencing with a smaller coverage, which allows us to save some costly wet lab work. On the contrary, when used improperly, the reference sequence might introduce a bias into the assembly process [42].

In Chapter 9, we will need to work with both read and contig data simultaneously. As a shorthand, we define a tuple of the latter.

Definition 2.18 (Read tuple) *Let A be a sequence, R_A its read bag, and C_A a contig set assembled from R_A by an assembly algorithm. We define a **read tuple**, or tuple for short, as*

$$T_A = (R_A, C_A). \tag{2.12}$$

If we write that a sequence is from T_A, we automatically assume that it is either a read from R_A or a contig from C_A.

2.5 Duality of Subsequence and Interval

All reads (and all contigs in an ideal situation) can be understood not only as subsequences of the DNA sequence but also as subintervals of the nucleotide indices. Throughout the work, we will often work with some prefix-suffix overlaps of two contigs (or reads, sometimes). However, in some cases, the overlapping subsequences will be from the same sequence, and in some cases, they will be from different sequences. To ensure clarity in the future, we need to distinguish those two cases. For this reason, we include the following definitions.

Definition 2.19 (Prefix-set, suffix-set, subsequence-set) *Let A be a sequence. We denote*

- *the set of all **prefixes** of A as* $\mathsf{Pref}(A)$,
- *the set of all **suffixes** of A as* $\mathsf{Suff}(A)$,
- *the set of all **subsequences** of A as* $\mathsf{Subseq}(A)$.

Definition 2.20 (Possible overlap set) *Let A and B be two sequences. We denote the set of all* ***possible overlaps*** *of A and B as*

$$\begin{aligned}\mathsf{Overlaps}(A,B) = \mathsf{Suff}(A) \times \mathsf{Pref}(B) \cup \mathsf{Pref}(A) \times \mathsf{Suff}(B)\\ \cup \mathsf{Subseq}(A) \times \{B\} \cup \{A\} \times \mathsf{Subseq}(B).\end{aligned} \quad (2.13)$$

Definition 2.21 (Sequence overlap) *Let A, B be two sequences. We say that A and B* ***overlap*** *with respect to an algorithm if this algorithm predicts an element from set $\mathsf{Overlaps}(A,B)$ to be a high similarity region of A and B.*

Informally, the overlap is a calculated prefix-suffix (or substring) similarity of two subsequences (of two possibly different sequences). If two sequences overlap, it means that some of the nucleotides of one sequence are contained in the other sequence as well. On the contrary, the intersection will be defined for two subsequences of the same sequence. While overlap is inexact, the intersection is an exact match.

Definition 2.22 (Subsequence intersection) *Let $A = A_1A_2 \cdots A_{|A|}$ be a sequence. Let $A_i^j = A_iA_{i+1} \cdots A_j \in \mathsf{Subseq}(A)$ and $A_k^l = A_kA_{k+1} \cdots A_l \in \mathsf{Subseq}(A)$ be two subsequences of A. We say that x and y* ***intersect*** *if one of the following holds*

- *$i \leq k$ and $j \geq k$, or*
- *$i > k$ and $l \geq i$.*

2.6 Problems Connected with Assembly

The process of de-novo assembly is often difficult or even impossible. There are several sources of obstacles. There may be sequencing errors, the selection of reads from $\mathcal{R}_A$ is random, and many problems come from the complexity of the underlying data.

- During the sequencing process, some errors may be introduced. The sequencing machines include in the FASTQ format a line that shows the expected error rate at each position of the read. Because the fragments are located in clusters on a chip, sometimes, two clusters merge. As a result, the sequencing machine gets contradictory signals about the nucleotide present in each cluster [82].

  Other errors come from the fact that the sequencing process is quite lengthy. As there is a mixture of chemicals on the chip, the quality of data degrades with time. For example, during the process of phasing, the blocker that prevents the hybridization of multiple nucleotides at once might not be removed from some of the fragments in the cluster. The respective fragment is then read one nucleotide behind the rest of the cluster. Other types of errors can happen as well. As a result, there is usually a lower error rate at the beginning of each read than at the end [98].

- The randomness of read locations causes that, despite a high coverage, there may be some parts of the genome that were not covered by any of the reads. The uniform i.i.d. assumption is idealized and is not valid in the real world, as stated in [233]. The enzymes may be unlikely to fragment the DNA molecule at some positions [225]. In a better case, an assembly algorithm reports two contigs; in a worse case, two non-intersecting contigs are connected due to a random prefix-suffix overlap.

  On the contrary, some of the simplest ways to check the assembly quality are based on the coverage [73]. For example, when the estimated per-base coverage is approximately two times higher than elsewhere for a part of a contig, the assembly algorithm might have used reads from a paralog gene. The paralogs, which are duplicated genes in a single species, are not only important evolutionary but are very useful in classifying organisms [154].

- Suppose that sequence A contains a repeated subsequence, called a *repeat*. If this subsequence is longer than l and is repeated at least 3-times, then no assembly algorithm has a guide to order the blocks between the repeats. As a result, multiple contigs are usually generated. For random sequences, the repeats are extremely unlikely; however, in real-world data, they are common, especially for plants [95]. Similarly, if there is a single-nucleotide subsequence longer than l, no assembly algorithm can reliably identify its exact length.

  Telomeres are a perfect example of a repeat that cannot be sequenced by the shotgun sequencing method. Telomeres are protective caps at the ends of chromosomes long 5 to 15 kbp containing repeated `TTAGGG`. Each time the cell replicates, the chromosome ends shorten. The telomeres, therefore, protect valuable genetic information from being lost in the cell replication process. The shotgun sequencing produces many similar reads, however, with only limited information about the telomere length (and cell's potential for future replications) [70].

2.7 Avoiding the Assembly Step?

In the following chapters, we will propose a way to use read data in unsupervised learning. Mainly, we will focus on phylogeny as one of the most popular applications of unsupervised learning in bioinformatics. To do so, we will use read bags directly.

A typical property of distance-based algorithms such as the neighbor-joining algorithm [249] or the UPGMA algorithm [269] is that they rely on a single input - the distance matrix between the clustered objects. Therefore, a distance measure will be developed as a crucial requirement for phylogeny. The idea is sketched in Figure 2.2. Instead of calculating the estimates of the genomic sequences to estimate the distance, we will estimate the distance directly from read bags. A theoretical discussion will follow with a calculation of the p-value for the proposed distance measure. Later, we will show that the proposed measure is connected to the evolutionary distance as a lower bound.

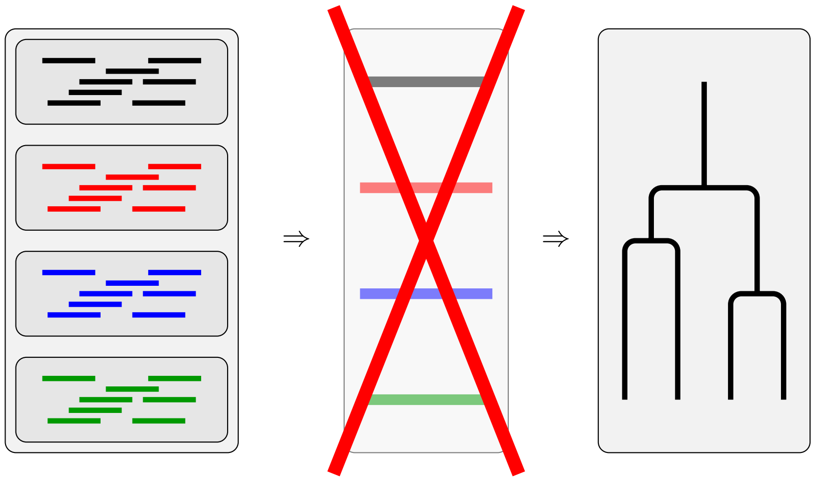


Figure 2.2: A common approach for phylogeny consists of a two-step procedure. First, the read bags are assembled into long sequences, and then those sequences are used for phylogeny. In this proposal, we either avoid the assembly at all or assemble only the easy parts.

Chapter 3 Related Work

In this chapter, we will first focus on the related work describing the sequence distance calculations. Then, we will briefly describe the most common algorithms for inferring phylogenetic trees. We will overview approaches for hierarchical clustering. As a next step, we will mention the alignment-free approaches, out of which several apply to NGS data, and therefore, they are trying to solve the same problem as our method. The chapter is closed with an overview of current sequence assembly algorithms.

3.1 Sequence Distance

The most common approach for calculating the Levenshtein distance [167] defined in Definition 2.3 is the *Wagner-Fischer algorithm* [296]. It is a simple dynamic programming algorithm based on the recursive formula

$$\operatorname{dist}(A,B) = \min \begin{cases} \operatorname{dist}(A_1^{|A|-1}, B) + 1, \\ \operatorname{dist}(A, B_1^{|B|-1}) + 1, \\ \operatorname{dist}(A_1^{|A|-1}, B_1^{|B|-1}) + [\![A_{|A|} \neq B_{|B|}]\!] \,, \end{cases} \tag{3.1}$$

where $[\![\cdot]\!]$ represents 1 if the condition inside is true, 0 otherwise. The first case corresponds to a deletion of a symbol in A (or inversely, an insertion in B if we switch the direction in which the operations are applied), the second case corresponds to an insertion in A (a deletion in B), and the third case represents a substitution or a match.

A dynamic programming algorithm based on Formula (3.1) was invented multiple times, and the list of inventors includes several names. In the context of the Levenshtein distance calculation, the *Wagner-Fischer algorithm* [296] is usually referenced. However, in the context of bioinformatics and sequence alignment, the algorithm is sometimes called the *Needleman-Wunsch algorithm* [208]. In that case, Formula (3.1) is used in a second pass to backtrack the alignment of the sequences. The algorithm then uses information which of the cases considered in the min operator in (3.1) was used to construct the alignment. The first case corresponds to a gap in sequence B, the second to a gap in A, and the third to either a match or a mismatch.

There are several variants of the Wagner-Fischer algorithm. A formula similar to (3.1) is used for calculating the local alignment using the *Smith-Waterman algorithm* [268]. The only difference is that we do not enforce the whole sequence to be used in the alignment, and therefore, we search for the most similar substring of both sequences. This search is implemented by an additional 0 option in the set of the minimum operator arguments in Formula (3.1).

3.1.1 Substitution Matrix and Gap Penalty

The bioinformaticians often replace the uniform cost for substitutions, deletions, and insertions with a scoring matrix [8] that provides different costs for different nucleotide or amino-acid mutations. In the case of amino acids, the BLOSUM matrix [120] and the PAM matrix [64] are commonly used. The BLOSUM matrix was designed to identify conserved regions of divergent proteins; hence, it is appropriate for local alignment. The PAM matrix, on the contrary, is used to calculate the similarity between related proteins using the global alignment. Both matrices have different versions based on the expected distance of the sequences compared.

For nucleotide substitutions, several models exist. Transitions (A-G and C-T substitutions) and transversions (A-C and G-T substitutions) have different probability, and those probabilities are reflected in the matrices. The models differ by the number of free parameters and form a hierarchy. The most important substitution models include the Tamura-Ney (TN93) model [281], the Kimura 2-parameter (K80) model [151], the Jukes-Cantor (JC69) model [140] and the F81 model [89]. For this thesis, the most relevant is the Jukes-Cantor model, which assumes that all substitutions are equiprobable and that all basis have probability $\frac{1}{4}$.

Not only the substitution penalty might be changed, but we may also introduce a different gap penalty. The biological motivation is that the long gaps are usually more likely than several consecutive single-nucleotide gaps [7]. To account for this phenomenon, we might introduce the *affine gap penalty*. For a gap of length g, the cost is

$$(\text{gap open penalty}) + g \cdot (\text{gap extension openalty}). \tag{3.2}$$

As a result, we need more memory to calculate the alignment. Nevertheless, the asymptotic complexity remains unchanged. Instead of the affine function, we might introduce any cost function, which, however, might affect the asymptotic complexity.

3.1.2 Sub-quadratic Approaches

The problem with the previous algorithms is that they require a quadratic amount of work (i.e., $\mathcal{O}(|A||B|)$ for $\text{dist}(A, B)$ calculation). Therefore, much effort was put into the development of faster algorithms. The first group relies on a faster exact calculation of the Levenshtein distance. We can mention the *Ukkonen's cutoff* [287, 288]. The algorithm runs in $\mathcal{O}(\text{dist}(A, B) \cdot \min(|A|, |B|))$ time. The speedup is based on the observation that not all fields in the dynamic programming table need to be calculated in order to determine the Levenshtein distance. If we have some upper bound k on the Levenshtein distance, we do not need to look further than on k-th diagonal from the middle one. This restriction comes from the Levenshtein distance being monotonic along each diagonal of the dynamic programming table. The approach of Ukkonen was further improved by [22]. A competing approach was proposed by [194], where the runtime is equal to $\mathcal{O}\left(|A| \cdot \max\left\{1, \frac{|B|}{\log |A|}\right\}\right)$, under the assumption that $|A| \geq |B|$.

Besides the exact approaches, there are approximate methods for calculating the similarity (or distance) of two sequences. The most important is the BLAST algorithm [9]. This algorithm is used mainly for searching large databases. The algorithm finds high-scoring subsequences (of length 3 in the case of amino acids and 11 in the case of nucleotides). Those subsequences are found in the database using a fast index. The alignment is expanded only locally around those matches using the classical dynamic programming in the original version of the algorithm. Newer variants include optimizations. For example, two nearby hits can be required to initiate the alignment expansion [10]. When the accuracy of the match starts to drop below a threshold, the algorithm stops and reports the sequence.

3.2 Embeddings for the Edit Distance

One of the ways to get below the quadratic runtime of the edit distance calculation is a usage of an *embedding*. An embedding is a mapping that preserves some structure of the data. In the case of the Levenshtein distance, we would like to map strings to some other space (of strings, numbers, vectors, . . .) so that the relative distance is preserved. Ideally, the embedded strings are close under a distance measure if and only if the original strings are close. The distance in the embedded space is supposed to be easier to calculate than the distance in the original space.

Work [214] proposes to embed $\{0,1\}^l$ to the ℓ_1 space with distortion $2^{\mathcal{O}\left(\sqrt{\log l \log \log l}\right)}$. This means that the distances are preserved up to the factor $2^{\mathcal{O}\left(\sqrt{\log l \log \log l}\right)}$ and uniform scaling.

Work [39] proposes a randomized embedding to the space of strings with the Hamming distance [115]. The Hamming distance between the projections of A and B is in $\mathcal{O}\left((\operatorname{dist}(A,B))^2\right)$ with a high probability. The embedding, therefore, does not increase the distance more than quadratically. The embedding from [39] can be calculated online, and only local knowledge of the sequences A and B is needed. This streaming approach was further extended by [318] on genomic data and similarity joins that we will present in the next section.

3.3 Approximate String Matching

The work of Ukkonen [287, 288] has a strong connection to the field of *approximate string matching*. A classic overview of the approximate string matching field is in [207]. The motivation for this field is the ability to fix typos in human writing. When a human makes several typos in a word, a computer program may try to find the closest match in a dictionary. Therefore, the goal is to find words that differ by at most k edit operations under some metric. A good example of a metric is the Damerau-Levenshtein distance [61], which we mentioned in Section 2.1. In this context, one often has to solve the *similarity join* and the *dictionary search* problems.[1]

Definition 3.1 (Dictionary search) *Let x be a string, S be a set of strings, k a fixed threshold, and d a distance measure on strings. Find*

$$\{x' \in S \mid d(x,x') \leq k\}.$$

Definition 3.2 (Similarity join) *Let S_1, S_2 be two sets of strings, k a fixed threshold, and d a distance measure on strings. Find*

$$\{(x_1,x_2) \in S_1 \times S_2 \mid d(x_1,x_2) \leq k\}. \tag{3.3}$$

The evaluation of (3.3) is similar to the evaluation of the Monge-Elkan distance in (2.6). The main difference is that we are interested only in the most similar string instead of all that are closer than the threshold.

There are several ways to evaluate the similarity searches and joins more effectively than using a naïve implementation. The approaches follow several directions, namely automata (proposed by Ukkonen in [289]), the bit-parallelism approach (proposed in the Ph.D. thesis of Baeza-Yates [14]), filtering (for example by q-grams [290]), and modifications of the dynamic programming algorithms [307].

One of the most straightforward approaches to improve dictionary search is with a *trie*. A trie [65] is a data structure that can store sets of strings by exploiting common

[1]Sometimes, an equivalent term *similarity search* is used.

prefixes. It is efficient, especially for small alphabet Σ. Each node has up to $|\Sigma|$ children, each representing one single character prefix. Each leaf stores a string that starts with characters that annotate the path from the root to the leaf. Because common prefixes are represented by the same path, a trie forms a compact representation of a set of strings. [243]

Paper [258] proposed a way to use PATRICIA trie [202], an even more compact representation of trie, for *dictionary search*. Suppose that the query string is ATCA, $S = \{\mathsf{AGCT}, \mathsf{AGAA}\}$, and assume the usage of the standard Wagner-Fischer [296] dynamic programming algorithm. Once the distance of ATCA and AGCT is calculated, the algorithm moves to calculate the distance of ATCA and AGAA. However, strings AGCT and AGAA share a common prefix — AG. The first three rows of the dynamic programming table will be equal. Therefore, [258] proposes to build a trie on set S and then traverse the trie in a way that avoids repeated work. Each time a leaf node of a trie is reached, the algorithm backtracks only to the deepest node lying on a path to a not-yet-explored string from set S. [243]

3.4 Clustering in Bioinformatics

There are several approaches to cluster reads to speed up the assembly process and other related NGS algorithms. Paper [15] uses clustering of reads to remove redundancy in data. This allows many NGS data analysis algorithms to run faster because it is possible to replace clusters only by selected representatives. Paper [187] uses clustering of reads in context of mapping reads to a reference genome. Paper [147] uses clustering to speed up the assembly process. Having the clusters, the authors can align the reads effectively and provide a de novo assembly. Paper [303] proposes an alignment-free measure of the similarity between reads based on k-mer counting.

Commonly, clustering is used in bioinformatics for phylogeny. A hierarchical clustering can be represented as a dendrogram representing an evolutionary tree, where the length of an edge is proportional to the evolutionary distance. To calculate such a tree, usually, the agglomerative approaches are used, mostly *UPGMA* [269] and the *neighbor-joining* algorithm [249]. Another simple approach is WPGMA [269], which is a weighted variant of UPGMA.

The agglomerative algorithms follow the same schema. First, there is one cluster created for each clustered object, in our case for each organism. The distances between the clusters are held in a distance matrix. As long as there is not a single cluster, two clusters are merged under some criteria, and the distance matrix is recalculated to reflect the new setting.

UPGMA and the neighbor-joining algorithm belong to the group of algorithms called *distance based.* Those algorithms rely on a distance matrix. This matrix is traditionally built from a multiple-sequence alignment, where we align all sequences at once. The alignment can be calculated by a simple generalization of the Wagner-Fischer algorithm in $\mathcal{O}(L^n)$, where L is the length of the sequences and n is the number of the sequences. Other, more effective approaches exist, such as CLUSTALW [123] or STAR [111]. In the simplest model, the evolutionary distance can be obtained from the multiple sequence alignment by the Jukes-Cantor estimate [140] as

$$\operatorname*{dist}_{JC}(A, B) = -\frac{3}{4} \ln \left(1 - \frac{4}{3}p\right), \tag{3.4}$$

where p is the number of mismatched nucleotides in the alignment of A and B divided by the length of this alignment. Formula (3.4) is justified by the fact that for random sequences and uniform nucleotide probability, one-quarter of nucleotides match.

For similar sequences, the number of mismatched nucleotides is small compared to the length of the alignment. Hence, p being small allows us to use the first-order Taylor approximation as follows

$$\underset{JC}{\operatorname{dist}}(A, B) \approx -\frac{3}{4}\left(-\frac{4}{3}p\right) = p. \tag{3.5}$$

We see that value p can be used directly for similar sequences.

Another big group of phylogeny algorithms consists of *parsimony-based approaches*. The main idea is to find a tree that best explains the observed sequences. Similarly to the maximum likelihood approach and Occam's razor principle, the tree is reconstructed so that the minimum number of mutations had to occur during the evolution. This procedure does not only give a phylogenetic tree but provides an estimate of the DNA sequences of the extinct species. The *Fitch algorithm* [93] can be used to find labeling (i.e., the sequences) for non-leaf nodes of a known tree under the assumption that the cost of each mutation is the same. A weighted generalization is the *Sankoff algorithm* [252] that uses dynamic programming to find labeling for non-uniform mutation costs. The runtime of this algorithm is $\mathcal{O}(|\Sigma|nm)$, where n is the number of species to be clustered, and m is the number of bases in the sequences. The more general problem of finding the most parsimonious tree, i.e., when the tree structure is not known, is NP-hard [63]. Exact methods are based on a *branch and bound* search [119]. However, due to the complexity of the problem, they cannot be used for whole genome studies, and heuristical approaches have to be used.

The last group of methods is based on *maximum likelihood* [90]. The *Felsenstein algorithm* [90] is a method to calculate the likelihood of a tree that has labels in its leaves. The algorithm is based on dynamic programming. The likelihood of a node is calculated from the likelihood of the node's children for all possible annotations of the node. In this way, it is possible to compare dendrograms produced by several algorithms. Also, the approach might be turned into an iterative procedure when a tree is calculated using a multiple-sequence alignment. Then this tree is used to calculate an alternative multiple sequence alignment [91]. Further, those two steps are iterated.

3.4.1 UPGMA Algorithm

The UPGMA algorithm [269] restricts the calculation to the *ultrametric* distance matrices. An ultrametric tree is a rooted tree where all paths from the root to any leaf have the same length. An ultrametric matrix is a matrix that can be obtained from an ultrametric tree by summing the lengths of the edges connecting a pair of leaves. As a result, we can find the distance to a common ancestor for any two leaf vertices by halving their distance. Ultrametricity can be tested by the following theorem and matches the real world under the *molecular clock hypothesis* assumption [329].

Theorem 3.3 (Ultrametric matrix) *Symmetric matrix* $\mathbf{D}$ *is* ***ultrametric*** *if and only if, for every three indices i, j, and k, the maximum of $D(i,j)$, $D(j,k)$, and $D(i,k)$ is not unique.*

In UPGMA, the cluster distance is defined as the arithmetic average of all distances between the objects in the clusters, i.e.,

$$D(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{c_i \in C_i, c_j \in C_j} d(c_i, c_j),$$

where $C_{\bullet}$ represents a cluster, D is a distance measure between the clusters, and d is the distance between the clustered objects. It is possible to develop a recursive formula for updating the cluster distances

$$D(C_{(ij)}, C_k) = \frac{|C_i|D(C_i, C_k) + |C_j|D(C_j, C_k)}{|C_i| + |C_j|},$$

where $C_{(ij)}$ is the cluster formed by merging clusters C_i and C_j. C_k denotes an arbitrary cluster different from C_i and C_j. In each step of the algorithm, the two closest clusters are merged.

3.4.2 Neighbor-joining Algorithm

The neighbor-joining algorithm [249] avoids the molecular clock hypothesis assumption. The claim is that the evolutionary changes do not have the same speed on all branches of the evolutionary tree. Therefore, it is only sometimes correct to merge the two closest clusters. The neighbor-joining algorithm merges two clusters that are close to each other and, at the same time, far away from the others. The cost of avoiding ultrametricity is that the neighbor-joining algorithm produces only unrooted trees.

The neighbor-joining algorithm can optimally reconstruct a phylogenetic tree for *additive* matrices.

Definition 3.4 (Additive matrix) *Symmetric matrix* $\mathbf{D}$ *is* ***additive*** *if and only if there exists a tree such that for any indices* i, j, $D(i, j)$ *is equal to the sum of the lengths of the edges connecting* i *and* j.

The additivity of a matrix can be tested by the *four-node condition* [33]. This condition is stronger than the *triangle inequality*, and it implies that the triangle inequality is true but not vice versa.

Theorem 3.5 (Buneman, 1974, four-point condition) *Symmetric matrix* $\mathbf{D}$ *is additive if and only if, for every four indices* i, j, k, *and* l, *two sums of*

$$D(i,j) + D(k,l), \qquad D(i,l) + D(j,k), \qquad D(i,k) + D(j,l)$$

are the same, and the other is smaller.

The neighbor-joining algorithm merges in each step two clusters C_i, C_j minimizing

$$D(C_i, C_j) - \frac{1}{n-2} \sum_{C_k} \left(D(C_i, C_k) + D(C_j, C_k) \right), \tag{3.6}$$

where n is the number of clusters. The first part of the sum tells that clusters C_i and C_j should be close to each other, and the second part says that C_i and C_j should be far apart from the remaining clusters. Equation (3.6) guarantees that clusters C_i and C_j share a common parent. By solving a simple set of equations for three leaves, it is possible to calculate the update rule

$$D(C_{(ij)}, C_k) = \frac{1}{2} \left(D(C_i, C_k) + D(C_j, C_k) - D(C_i, C_j) \right).$$

3.4.3 Notes on Clustering Algorithms

Both UPGMA and the neighbor-joining algorithm make assumptions about the underlying distance matrix. The problem is that for many common distances that follow the metric properties, the conditions from Theorem 3.3 (or Theorem 3.5) are not met. This issue would make those algorithms often unusable in praxis. Therefore, they are used anyway as a form of a heuristic, although they may produce counterintuitive results as, for example, negative edge lengths. When a matrix is additive, the neighbor-joining is, however, guaranteed to generate the optimal tree. Similarly, for the ultrametric matrices, UPGMA is guaranteed to generate the optimal tree.

Regarding the neighbor-joining algorithm, we may be interested in finding a rooted tree. A rooted tree allows some additional analyses which are not possible in unrooted trees [87]. However, the actual root may be located on any edge of the tree. To remove this uncertainty, a concept of an *outgroup* [87] is used. Researchers include in the tree some organism which is not interesting to them and which is different from the others. This organism is the one that has the highest evolutionary distance from all other clustered organisms. Therefore, the root lies somewhere on the edge connecting the outgroup with the rest of the tree. If an outgroup is not in the data, a good choice for placing the root can be the longest edge connecting a leaf with the rest of the tree.

3.5 Alignment-free Measures

The community that proposed the alignment-free measures reasons that the traditional methods for phylogeny based on the sequence alignment are slow. The alignment-free measures usually avoid the alignment step by various statistics on the sequences representing the clustered organisms.

As we will see in the following subsections, many alignment-free methods are applicable to raw read data because they rely mostly on k-mer counts, which can be approximated from the read bags. A brief overview of the methods mentioned in this section is in Table 3.1.

3.5.1 Statistical Measures

One of the first papers that proposed an alignment-free measure was [25]. The work proposed the D_2 statistic to compare the correlation between two sequences. The D_2 statistic is a dot product of vectors containing the number of occurrences for all possible k-mers. Using the notation from Section 2.2.2 we can write that

$$D_2(A, B) = \mathbf{Q}_k^T(A)\mathbf{Q}_k(B) = \langle \mathbf{Q}_k(A), \mathbf{Q}_k(B) \rangle. \tag{3.7}$$

Further, a new statistic $D2z$ was proposed by [143], and D_2^S and D_S^* were proposed by [231]. $D2z$ is motivated by the fact that the background models for DNA sequences for various organisms may be different. Therefore, the D_2 statistic should be normalized as follows

$$D2z(A, B) = \frac{D_2(A, B) - \mathsf{E}(D_2(A, B))}{\sigma(D_2(A, B))}, \tag{3.8}$$

where σ denotes the standard deviation. To introduce the D_2^S statistic and the D_2^* statistic, we need to define a few symbols. Let $\text{count}(a, A)$ denote the number of occurrences of k-mer a in A. Let $p(a) = \prod_{i=1}^{k} p(a_i)$ denote the probability that the subsequence a is observed by chance. Under the equiprobable nucleotides model (Jukes-Cantor, proposed in [140]) this probability is equal to $\left(\frac{1}{4}\right)^{|a|}$. Define

$$\widetilde{\text{count}}(a, A) = \text{count}(a, A) - (|A| - k + 1) \cdot p(a).$$

In the formula, we see that the true count of k-mer a is decreased by the expected count when it is observed by chance. Paper [231] then defines

$$D_2^S(A, B) = \sum_{w \in \Sigma^k} \frac{\widetilde{\text{count}}(w, A) \cdot \widetilde{\text{count}}(w, B)}{\sqrt{\widetilde{\text{count}}(w, A)^2 + \widetilde{\text{count}}(w, B)^2}}. \tag{3.9}$$

Method	Paper	Assembled	Read bags	Main idea
D_2	[25]	✓	✗	k-mer counts
$D2z$	[143]	✓	✗	k-mer counts
D_2^S	[231]	✓	✗	k-mer counts
D_2^*	[231]	✓	✗	k-mer counts
d_2	[271]	✗	✓	k-mer counts
d_2^S	[271]	✗	✓	k-mer counts
d_2^*	[271]	✗	✓	k-mer counts
CVTree, Hao	[226]	✓	✗	k-mer counts
S_2	[60]	✓	✗	k-mer counts
$\mathcal{U}nder_2$	[53]	✓	✗	underlying patterns
$\overline{\mathcal{U}nder_2}$	[51]	✗	✓	underlying patterns
QCluster, D^q	[54]	✗	✓	quality values
d^q	[55]	✗	✓	quality values
	[26]	✓	✗	spaced k-mers
FSWM	[165]	✓	✗	filtering, spaced k-mers
co-phylog	[315]	✓	✓	C-grams and O-grams
	[285]	✓	✓	compression-based distance
Mash	[213]	✓	✓	dimensionality reduction
NexABP	[234]	✗	✓	anchors
Cnidaria	[3]	✓	✓	k-mer counts
	[195]	✗	✓	k-mer counts
`kWIP`	[204]	✓	✓	inner product, hashing
	[254]	✓	✓	fuzzy integral similarity
	[177]	✓	✗	conversion to images
andi	[116]	✓	✗	small exact matches
Afann	[282]	✗	✓	bias adjustment
	[44]	✓	✓	Bloom filters
	[196]	✗	✓	Earth mover's distance
	[148]	✗	✓	Earth mover's distance

Table 3.1: Selected alignment-free methods that can be used for genome comparison and their intended use.

In praxis, the probability of each nucleotide is estimated by the relative counts of each nucleotide in both sequences under the null hypothesis that both sequences were generated i.i.d. from the same distribution. Denote this empirical probability as $\hat{p}(a) = \prod_{i=1}^{k} \hat{p}(a_i)$. Then [231] defines

$$D_2^*(A,B) = \frac{1}{\sqrt{(|A|-k+1)(|B|-k+1)}} \sum_{w\in\Sigma^k} \frac{\widetilde{\text{count}}(w,A)\cdot\widetilde{\text{count}}(w,B)}{\hat{p}(w)}. \tag{3.10}$$

The measures D_2^S and D_2^* proposed in [231] were further studied in [298]. The authors provided a detailed theoretical analysis of the measures.

The D_2, D_2^S, and D_2^* statistics are used for comparing long sequences. Work [270] modify them so that they can be used for short read NGS data. This conference paper was further extended in journal paper [271]. The main motivation is the fact that the magnitudes of the original statistic depend on the sequence length, nucleotide frequencies, and the number of reads. Therefore, the values are normalized to the unit interval. The new statistics are defined as

$$\begin{aligned}
d_2 &= \frac{1}{2}\left(1 - \frac{D_2(A,B)}{\|\mathbf{Q}_k(A)\|_2\|\mathbf{Q}_k(B)\|_2}\right),\\
d_2^S &= \frac{1}{2}\left(1 - \frac{D_2^S(A,B)}{\sqrt{\sum_{w\in\Sigma^k}\frac{\widetilde{\text{count}}(w,A)^2}{\sqrt{\widetilde{\text{count}}(w,A)^2+\widetilde{\text{count}}(w,B)^2}}}\sqrt{\sum_{w\in\Sigma^k}\frac{\widetilde{\text{count}}(w,B)^2}{\sqrt{\widetilde{\text{count}}(w,A)^2+\widetilde{\text{count}}(w,B)^2}}}}\right),\\
d_2^* &= \frac{1}{2}\left(1 - \frac{D_2^*(A,B)}{\sqrt{\sum_{w\in\Sigma^k}\frac{\widetilde{\text{count}}(w,A)^2}{(|A|-k+1)\hat{p}(w)}}\sqrt{\sum_{w\in\Sigma^k}\frac{\widetilde{\text{count}}(w,B)^2}{(|B|-k+1)\hat{p}(w)}}}\right).
\end{aligned} \tag{3.11}$$

In the formula above, we can approximate $\mathbf{Q}_k(A)$ either from reads by $\sum_{a\in R_A}\mathbf{Q}_k(a)$ or from contigs by $\sum_{\alpha\in C_A}\mathbf{Q}_k(\alpha)$. This approximation is common for all other alignment-free measures that can process raw read data.

Among the previously listed statistics, there have been several others developed. Namely, we can mention the *Hao* statistic proposed in [226, 313, 227], which is applicable to whole genomes. The method uses Markov models to eliminate random noise from the original k-mer counts. As a result, the weight of the evolutionary changes in the model is increased. Another measure based on a synergy of Markov models and k-mer counts is $S2$, which was proposed in [60].

A review of other alignment-free measures can be found in [272]. This paper claims that the provided measures were used several times in praxis. However, due to the randomness of read bags, they are less powerful with read data than with whole genomes. Other review papers include [27, 28, 327].

3.5.2 Work of Matteo Comin

Work [53] proposed measure $\mathcal{U}nder_2$ based on a concept named the *underlying subword*. The authors decided to filter out from the statistic words that are located in large non-coding regions. Those regions tend to be highly repetitive and may skew the similarity score. The paper shows that it is possible to define a distance measure on the irredundant common subwords. Work [53] is based on [50].

Work [51] extends this approach to NGS data. After finding the underlying patterns from read bags, the authors propose a measure $\overline{\mathcal{U}nder_2}$ and compare it with all d-type statistics from (3.11). The measure performs better on most of the experimental data allowing a better phylogenetic tree reconstruction. One of the main advantages is that the proposed method does not rely on a fixed pattern length which is commonly used as it was in the case of k-mers. The proposed method is parameter-free. The method was further described in [52].

Comin et al. proposed in [55] to modify the d-type statistics by using the quality values we mentioned in Section 2.6. In FASTQ file format, the sequencing machines provide estimates of the probability that the particular nucleotide was sequenced with an error. Paper [55], therefore, proposes to use the quality values to weight each occurrence of a k-mer. Each occurrence of k-mer a is weighted by the probability that it was read without any error. Each occurrence of a word a is, therefore, counted as a real number between 0 and 1, instead of 1. Following this idea, the d-type statistics are replaced with the d^q-type statistics defined similarly to (3.11). Work [55] was based on conference paper [54], where the same authors modified just the D-type statistic to deal with the quality values. The authors also make their source code available on `http://www.dei.unipd.it/~ciompin/main/qcluster.html`. We will use their implementation for evaluation of the proposed method in Chapter 11.

3.5.3 Other Relevant Alignment-free Measures

Work [26] proposes using the *spaced k-mers* to estimate the distance. A spaced k-mer contains besides k important symbols also some number of positions where we *do not care* about the underlying nucleotide. Positions of don't care characters are fixed and predefined. Once the spaced k-mer counts are counted, the count vector is used similarly to other alignment-free measures, i.e., as an input to a numeric distance. In the study, the Jensen-Shannon distance [178] is used. The method is proposed only for whole genomes, not for read data. The authors further developed their method, for example, in works [125, 164].

Some of the authors of paper [26] continued with the development of this approach and proposed the *Filtered Spaced Word Matches* (FSWM) [165] approach that improved the previous work by filtering and allows fast phylogeny. Paper [165] was based on thesis [163]. Paper [201] is the last work using the spaced words we can track.

Outside of the main track of the statistical measures designed for NGS reads stands the *co-phylog* algorithm [315]. The authors modify the concept of k-mers by introducing C-grams and O-grams, which can simulate a local alignment with only substitutions considered. For each exact match (a C-gram), some inexact close O-grams are considered and then used in the distance calculation. The algorithm can identify well outer branches of the phylogenetic tree; however, it does not work well for distant organisms. The algorithm can calculate the similarity of unassembled read data as well as the similarity of contigs.

Work [285] proposes to use a *compression-based* distance measure. The idea is that the NGS data contain redundancy. By removing this redundancy, it is possible to compress sequences into a smaller representation that can be used by the standard methods to estimate the sequence similarity. The paper then shows that the compression-based distances are consistent with the k-mer-based distances.

The *Mash* algorithm [213] relies on k-mer counting and MinHash dimensionality reduction techniques to estimate the similarity of two read bags or whole genome sequences. Instead of full count vectors $\mathbf{Q}_k(A)$, only vectors of counts of k-mer hashes are used.

Authors of [234] modify their anchor-based measure from [293] to be applicable to raw NGS data. Work [3] presents a tool based on k-mer counting that can be used for comparing whole genomes or raw sequencing data. Work [195] proposed another measure for inferring *E. coli* phylogenetic tree. Work [85] proposes an assembly-free and alignment-free method. The proposed method was designed for low-coverage data from poorly known species.

Work [204] proposes a novel metric called the *weighted inner product* with k-mer counts obtained from read data. The measure called `kWIP` is alignment-free and assembly-free and can be used with success for calculating phylogenetic trees. Again, it is based on an idea similar to Mash — the k-mers are hashed to produce a memory-efficient hash vector. This vector, called sketch, is then used as input for a statistical measure.

Two papers [196, 148] use the Earth mover's distance [168] to compare MiSeq data [229] from viral outbreaks. The MiSeq data contain very short reads that make assembly difficult. However, it is possible to capture many different populations of a rapidly developing virus.

3.5.4 Strengths and Future of the Alignment-free Measures

This section is far from providing a complete overview of the alignment-free measures. Query *„alignment-free genome comparison“* on Google Scholar returns more than 2000 scientific articles matching the query. The range of used papers includes many new and recent ideas from Bloom filters [44], neural networks [282], finding small exact matches [116], conversion to images and Fast Fourier transform [177], to fuzzy integral similarity [254].

Also, alignment-free measures have made their way into many applications. Only in 2022, the alignment-free methods were used, for example, to study covid data [152], coral organisms [180], bacteria plasmids [150], or endangered crocus variety [38]. The main advantage of the alignment-free methods compared to the alignment-based methods is their low sensitivity to events such as genome rearrangements. Genome rearrangements cannot be captured by the standard nucleotide-to-nucleotide pairing as the order of nucleotides changes dramatically [327]. Another advantage is that the alignment-free methods are faster by order of magnitude than the alignment-based methods [327]. A disadvantage is the apparent loss of information incurred by the breakage into k-mers.

In this section, we talked only about applications of the alignment-free measures to phylogeny and genome comparison. However, it is worth mentioning the Salmon tool [218], which belongs to the alignment-free approaches for abundance quantification in RNA-seq data. The algorithm uses the k-mer counts of genes and read data. The read k-mer counts are split onto genes sharing the same k-mers. The expectation-maximization algorithm [67] is then used to calculate the proper split ratio. The mapped k-mers counts are then used to quantify the abundances.

The author of this thesis recommends [327] as the best source to start with the alignment-free measures together with a benchmarking review [328] in which the author of the aforementioned review cooperated with many other authors of alignment-free measures to provide a dataset and toolkit for benchmarking the measures on various applications, including, among other, genome comparison of unassembled reads.

3.6 Discriminative Patterns

Works [295, 134, 133] try to solve the problem of finding the *discriminative patterns* in metagenomic samples. This problem belongs to the supervised learning area. On input, there are several positive examples and several negative examples. In an idealized case, the output is the shortest string that contains all positive examples as a substring and no negative example. This may not be possible, and also, in reality, we have read bags instead of the actual sequences. Therefore, the goal is to assemble a string with the best discriminatory properties. This string should cover as much from the positive examples and as little from the negative.

The proposed approach uses the PN-value criterion to measure the discriminative qualities of a pattern. Once the initial seeds are identified, a local view on the overlap graph is built in a lazy manner from the suffix-prefix overlap. Using a beam search, the initial seeds are prolonged as long as the expansion leads to a higher objective function value.

3.7 Assembly Algorithms

The sequence assembly process was mentioned in Section 2.4. In this section, we will focus on the state-of-the-art algorithms that solve this problem. The algorithms can be grouped into three groups: the greedy algorithms, the overlap layout consensus algorithms (OLC), and the de Bruijn graph algorithms (DBG). A review of the assembly methods can be found in [199].

The *greedy algorithms* start with a seed. In each iteration, this sequence is prolonged by a read that shares the maximum overlap with the current sequence. Once the overlap quality drops below a specified threshold, a contig is produced, and a new seed is found among the remaining reads. The main advantage of the greedy approaches is their simplicity. However, this is paid by the sub-optimality of the algorithm. The greedy approaches were the first ones used for the assembly of NGS data. Examples of the greedy algorithms are SSAKE [301], SHARCGS [72], VCAKE [136].

3.7.1 Overlap-Layout-Consensus

The *overlap layout consensus* approaches use an overlap graph for sequence assembly. For each pair of reads, their overlap is identified. The assembly algorithms differ in the way how the graph is constructed. Either only exact matches are considered, or the possibility for sequencing errors is included. Some algorithms filter out the non-perspective overlaps and avoid their calculation [110].

Each vertex in an *overlap graph* represents one read. The weighted edges connecting the reads represent the quality of the prefix-suffix overlap. A possible assembly then corresponds to a path in the overlap graph that goes through all reads. Because shorter assemblies are preferred, we are interested in finding the shortest path. The shortest path through all vertices reduces to the *traveling salesman problem* (TSP), which is a well-established problem from the graph theory. The TSP problem is, however, known to be NP-hard [185, 101]. Despite its complexity, the TSP problem is well known, and many heuristic/approximate algorithms running in polynomial time were developed. We can mention the $\mathcal{O}(n^3)$ Christofides algorithm [43], which is guaranteed to find a path that is longer by less than 50 % than the optimum. Therefore, the biggest source of the computational complexity is usually connected with finding the prefix-suffix overlaps of all reads.

Once the overlap graph is calculated and the longest path is identified, the putative alignment of the reads to the unknown sequence is calculated. We call this second step *layout*. The third step, called *consensus*, is used to identify each nucleotide from the reads. Majority voting is used together with weighting by the quality data.

The advantages of the OLC approaches include a direct connection to the shortest common superstring problem. On the opposite, computational inefficiency is a disadvantage. Common OLC assembly algorithms include Celera Assembler [205], Arachne [18], CAP and PCAP [128], and Edena [122].

3.7.2 De Bruijn Graph

The *de Bruijn graph* approaches rely on a modification of de Bruijn graphs. Each read is split in k-mers. Those k-mers then form vertices of a graph. Two vertices are connected by an oriented edge if and only if the respective two k-mers are consecutive in a read. For example, the sequence ACTGCCA would be split into $\{\mathsf{ACT}, \mathsf{CTG}, \mathsf{TGC}, \mathsf{GCC}, \mathsf{CCA}\}$ for $k = 3$. The graph then contains edges $(\mathsf{ACT}, \mathsf{CTG})$, $(\mathsf{CTG}, \mathsf{TGC})$, $(\mathsf{TGC}, \mathsf{GCC})$, and $(\mathsf{GCC}, \mathsf{CCA})$. We can reconstruct the original sequence by following the path formed by the aforementioned edges.

Finding a sequence in a de Bruijn graph reduces to finding an *Eulerian path*. This problem can be easily solved in linear time by Hierholzer's algorithm (described, for example, in the textbook [141]). Therefore, the main advantage of the DBG approaches is computational efficiency. There are, however, many problems that need to be solved. First, due to the double-stranded nature of the DNA molecule, we cannot be sure whether to include an edge between two k-mers or their reverse complements. Sequencing errors may introduce into graph new vertices and edges. Therefore, many heuristic approaches were proposed to eliminate those false edges. In this case, not only the existence of an edge is important, but its multiplicity is used to simplify the graph. For example, *spurs* are short dead-end paths leaving the main graph, which are connected only by a single edge. The spurs are induced by sequencing errors near the ends of reads. Similarly, an error near the middle of a read causes the creation of an alternative path, called a *bubble*. Details of the error correction techniques might be found, for example, in review paper [198].

Another problem that is connected with the DBG approaches is the sensitivity to the setting of k. Small k might lead to a too-dense graph, and too large k leads to a graph that is not connected. In the OLC approaches, the threshold for overlap is more natural. Decomposition of reads into shorter k-mers also causes a loss of information. Despite the disadvantages of the DBG approaches, the DBG algorithms can run faster than the OLC approaches. The common DBG approaches are Velvet [316], ABySS [266], AllPaths [35], SOAPdenovo [173], SPAdes [210], and Euler [222].

Chapter 4

Distance Estimation and Complexity

In this thesis, we restrict ourselves to the distance-based phylogeny methods. Their common property is that to calculate the phylogenetic tree, we need to calculate only the distance matrix. Therefore, the problem of inferring a phylogenetic trees reduces to inferring the distance matrix between the clustered species. The idea is sketched in Figure 5.1. As we will see in Chapter 13, the applicability of a distance measure is much wider than only application to phylogeny, aka hierarchical clustering.

The problem does not provide any guarantees for an exact solution. We cannot be sure that reads cover all spots on the original DNA sequence. In the opposite extreme, the contigs can have even bigger sum of lengths than the length of the original sequence. Therefore, an approximation of the Levenshtein distance is needed. In this chapter, we will show that the problem is hard from the computational point of view. [240]

The *shortest common superstring problem* is often formalized differently from Definition 2.15. The problem might ask for existence of a superstring instead of finding it while remaining its computational complexity with respect to the NP class [153] and related classes of problems. The equivalence of optimization and decision is a more general concept from algorithm theory and the reader might be referred to Chapter 8 of book [153]. To conclude, the alternate definition is as follows.

Definition 4.1 (Shortest common superstring problem) *Given a set of strings $M = \{S_1, S_2, \dots, S_n\}$ and a positive integer K, is there a string S such that $|S| \leq K$, and for all $i \in \{1, 2, \dots, n\}$, S_i is a substring of S.*

To show the complexity of the problem solved in this thesis, we need to define the problem more formally. In the following definition, we will ask for the distance of the assembled sequences. As mentioned above, in an alternate definition, we might ask for the distance value. The yes/no problem is, however, more practical for the purposes of the following proofs.

Definition 4.2 (Partially-Assembled-Sequences Distance Problem (PASDP)) *Let A and B two sequences, and T_A, T_B their reads with partially assembled contigs, and let K be a non-negative integer. Let $\tilde{A}$ ($\tilde{B}$, respectively) be the shortest superstring of $R_A \cup C_A$ ($R_B \cup C_B$, respectively). Decide whether $\mathrm{dist}(\tilde{A}, \tilde{B}) \leq K$.* [240]

Theorem 4.3 *The partially-assembled-sequences distance problem is* NP*-hard.* [240]

Proof Let (M, K) be an instance of the shortest common superstring problem. We will show that it reduces to the PASDP.

Let $C_A = C_B = \emptyset$. Let $R_A = M$ and $R_B = \emptyset$. Then $\tilde{B} = \varepsilon$ and $\tilde{A}$ is the shortest superstring of M. As $\tilde{B}$ is empty, by the definition of the Levenshtein distance,

$$\mathrm{dist}(\tilde{A}, \tilde{B}) = \mathrm{dist}(\tilde{A}, \varepsilon) = |\tilde{A}| = |\text{shortest superstring of } M|. \tag{4.1}$$

The shortest superstring problem (M, K) has a solution if and only if $\operatorname{dist}(\tilde{A}, \tilde{B}) \leq K$. Therefore, the shortest common superstring problem cannot be harder than the PASDP, and as a result the PASDP is NP-hard. ■

The fact that our problem is NP-hard does not mean that it is not solvable. However, the complexity of the problem justifies the usage of heuristic approaches we will develop in future chapters. Also, one might argue that estimating the distance is, intuitively, a less complex problem than estimating the sequences. A small change in sequences, that in biological world coincides with transposition of two large blocks of nucleotides might lead to a drastic change in the distance of the assemblies. On the contrary, the heuristic we provide will be influenced marginally.

From the algorithmic point of view, a proof of NP-completeness is usually connected with a second proof to show that the problem is in class NP. For completeness, we include this proof in the next paragraphs.

Proposition 4.4 *The partially-assembled-sequences distance problem is in* NP.

Proof The proof will be done by providing a certificate and a polynomial time verifier. In our case, the certificate is tuple $(\tilde{A}, \tilde{B})$. In polynomial time, it is possible to verify that all strings in $R_A \cup C_A$ ($R_B \cup C_B$, respectively) are substrings of $\tilde{A}$ ($\tilde{B}$, respectively). The naïve implementation of this verification runs in

$$\mathcal{O}\left(|\tilde{A}| \cdot \left[\sum_{a \in R_A} |a| + \sum_{\alpha \in C_A} |\alpha| \right] + |\tilde{B}| \cdot \left[\sum_{b \in R_B} |b| + \sum_{\beta \in C_B} |\beta| \right] \right). \tag{4.2}$$

The complexity in (4.2) is quadratic with respect to the size of the input, as the shortest substring cannot be longer than all sequences in the set on the input.

The verifier then needs to calculate $\operatorname{dist}(\tilde{A}, \tilde{B})$, and compare it to K, which can be done in quadratic time using dynamic programming [296] and Formula (3.1).

As there exists a polynomial time verifier for the certificate, the problem is in NP. ■

Corollary 4.5 The partially-assembled-sequences distance problem is NP-complete.

Proof The statement is a direct corollary of Theorem 4.3 and Proposition 4.4. ■

Chapter 5 Estimating Similarity from Read Bags

In this chapter, we will present an approach that shows how to estimate the sequence similarity from read bags. The contents will correspond to the work presented in paper [237]. The text of this chapter will be mostly taken from its extended version [239].

For distance-based phylogeny methods, only a distance matrix between the clustered objects is needed. Therefore, our goal is to propose a distance function $\mathsf{Dist}(R_A, R_B)$ that approximates $\mathrm{dist}(A, B)$ for read bags R_A and R_B of arbitrary sequences A and B. We also want $\mathsf{Dist}(R_A, R_B)$ to be more accurate and less complex to calculate than a natural estimate $\mathrm{dist}(\tilde{A}, \tilde{B})$ in which the arguments represent putative sequences reconstructed from R_A and R_B using *assembly algorithms* described in Section 3.7. [239]

5.1 Distance Function Design

5.1.1 Base Case: Which Reads Belong Together [239]

A natural approach to instantiate $\mathsf{Dist}(R_A, R_B)$ is to exploit the $|R_A||R_B|$ pairwise Levenshtein distances between the reads in R_A and R_B. Most of those values are useless because they match reads from entirely different parts of sequences A and B. Therefore, we want to account only for those pairs which likely belong together.

If we seek a read from R_B that matches read $a \in R_A$, we make the assumption that the most similar read in R_B is the one that we look for (see Figure 5.2), i.e.,

$$\arg\min_{b \in R_B} \mathrm{dist}(a, b).$$

To calculate the distance from R_A to R_B, we average over all reads from R_A:

$$\underset{\mathsf{ME}}{\mathsf{Dist}}(R_A, R_B) = \frac{1}{|R_A|} \sum_{a \in R_A} \min_{b \in R_B} \mathrm{dist}(a, b). \tag{5.1}$$

This idea was presented in [200] for searching duplicates in database systems. The method is known as the *Monge-Elkan similarity* (hence the ME label), which we alter here into a distance measure.

In a practical setting, we do not know which DNA strand the reads come from. If we match read a with read b, there are two possible matchings. If reads come from the complementary strands, we need to calculate $\mathrm{dist}(a, \underleftarrow{\overline{b}})$ besides $\mathrm{dist}(a, b)$. We consider only the option that leads to a lower distance.

Based on the sequencing setting, we have up to four options for how to match reads a and b. The options that need to be considered are a subset of $\mathrm{dist}(a, b)$, $\mathrm{dist}(a, \overline{b})$, $\mathrm{dist}(a, \underleftarrow{b})$ and $\mathrm{dist}(a, \underleftarrow{\overline{b}})$. Other options are redundant as, for example, $\mathrm{dist}(\overline{a}, b) = \mathrm{dist}(a, \overline{b})$.

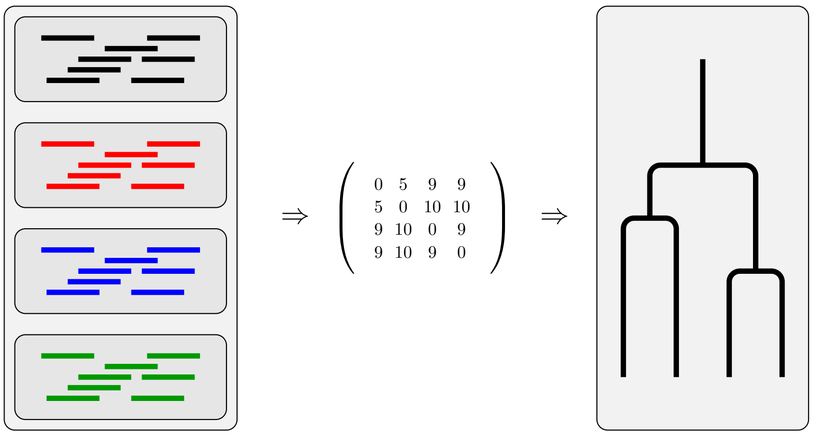


Figure 5.1: A procedure for phylogeny without assembly. In order to reconstruct the phylogenetic tree, we estimate the distance between the clustered objects directly from read bags. For this purpose, we estimate the Levenshtein distance between the DNA sequences from read bags.

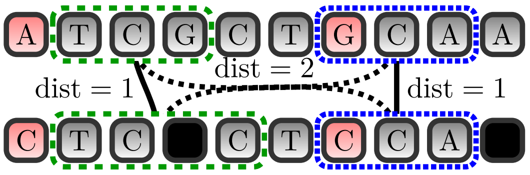


Figure 5.2: We calculate read-read distances in order to find matching pairs of reads. For each read from the first sequence, we find the least distant read in the second sequence. We see an optimal alignment of ATCGCTGCAA and CTCCTCCA. Read TCG is paired with TCC.

5.1.2 Symmetry [239]

$\mathsf{Dist}_{\mathsf{ME}}(R_A, R_B)$ is non-symmetric in general, which is undesirable given that the approximated distance $\mathrm{dist}(A, B)$ is known to be symmetric. Therefore, we define a symmetric version by averaging both directions

$$\underset{\mathsf{MES}}{\mathsf{Dist}}(R_A, R_B) = \frac{1}{2}\left(\underset{\mathsf{ME}}{\mathsf{Dist}}(R_A, R_B) + \underset{\mathsf{ME}}{\mathsf{Dist}}(R_B, R_A)\right). \tag{5.2}$$

5.1.3 Distance Scale [239]

Consider duplicating a non-empty string A into AA and assume $R_{AA} = R_A \cup R_A$. Typically, for a B similar to A, we expect that $\mathrm{dist}(AA, B) > \mathrm{dist}(A, B)$ but the (symmetric) Monge-Elkan distance will not change, i.e., $\mathsf{Dist}_{\mathsf{MES}}(R_{AA}, R_B) = \mathsf{Dist}_{\mathsf{MES}}(R_A, R_B)$, indicating a discrepancy that should be rectified.

In fact, $\mathsf{Dist}_{\mathsf{MES}}$ has the constant upper bound l. The distance is the average (c.f. (5.1) and (5.2)) of numbers no greater than l (see (2.5)). On the other hand, $\mathrm{dist}(A, B)$ has a non-constant upper bound $\max\{|A|, |B|\}$ as by (2.5).

To bring $\mathsf{Dist}_{\mathsf{MES}}(A, B)$ on the same scale as $\mathrm{dist}(A, B)$, we should, therefore, multiply it by the factor $\max\{|A|, |B|\}/l = \max\{|A|/l, |B|/l\}$. By (2.11), we have $|A| = \frac{l}{c}|R_A|$, yielding the factor $\max\{|R_A|/c, |R_B|/c\}$, in which c is a constant divisor which can be neglected in a distance function. Therefore, we modify the read distance into

$$\underset{\mathsf{MESS}}{\mathsf{Dist}}(R_A, R_B) = \max\{|R_A|, |R_B|\} \underset{\mathsf{MES}}{\mathsf{Dist}}(R_A, R_B).$$

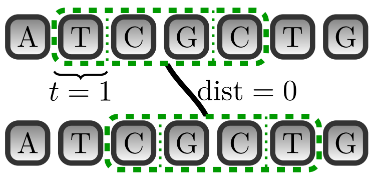


Figure 5.3: Because reads locations in sequences are random, we do not want to penalize small leading or trailing gaps.

5.1.4 Margin Gaps

[239]

Consider the situation in Figure 9.3 showing two identical sequences each with one shown read. The Levenshtein distance between the two reads is non-zero due to the one-symbol trailing (leading, respectively) gap of the top (bottom) read caused only by the different random positions of the reads rather than due to a mismatch between the sequences. Thus, there is an intuitive reason to pardon margin gaps up to a certain size t

$$t < \frac{l}{2} \tag{5.3}$$

when matching reads. Here, t should not be too large, as otherwise, the distance could be nullified for pairs of long reads with small prefix-suffix overlaps, which would not make sense.

To estimate an appropriate value for t, consider sequence A and its sampled read bag R_A. We now sample an additional read a of length l from A. Ideally, there should be a zero-penalty match for a in R_A as a was sampled from the same sequence as R_A was. This happens iff there is a read in R_A sampled from the same position in R_A as a, or from a position shifted by up to t symbols to the left or right as then the induced gaps are penalty-free. Since R_A is a uniform-probability i.i.d. sample from A, the probability that a particular read from R_A starts at one of these $1 + 2t$ positions is[1] $\frac{1+2t}{|A|}$. We want to put an upper bound $\varepsilon > 0$ on the probability that this happens for none of the $|R_A| = \frac{c}{l}|A|$ reads in R_A:

$$p = \left(1 - \frac{1+2t}{|A|}\right)^{\frac{|A|\cdot c}{l}} \leq \varepsilon.$$

Consider the first-order Taylor approximation $(1+x)^n = 1+nx+\varepsilon'$ where the difference term $\varepsilon' > 0$ decreases with decreasing $|x|$. Due to (5.3) and (2.10), $\frac{1+2t}{|A|}$ is small, and we can apply the approximation on the above formula for p, yielding

$$p = 1 - \frac{2t+1}{|A|}\frac{|A|\cdot c}{l} + \varepsilon' = 1 - (2t+1)\frac{c}{l} + \varepsilon' \leq \varepsilon.$$

For simplicity, we choose $\varepsilon = \varepsilon'$. The smallest gap size t for which the inequality is satisfied is obtained by solving $1 - (2t+1)\frac{c}{l} = 0$, yielding

$$t = \frac{1}{2}\left(\frac{l}{c} - 1\right). \tag{5.4}$$

This choice of t matches intuition in that with larger read-length l, we can allow a larger grace gap t, but with larger coverage c, t needs not be so large as there is a higher chance of having a suitably positioned read in the read bag. Another way to look at Formula

[1]Strictly speaking, this reasoning is incorrect if read a is drawn from a place close to A's margins, more precisely, if it starts in fewer than t ($t + l$, respectively) symbols from A's left (right) margin, as then not all of the $2t$ shifts are possible. This is, however, negligible due to (2.10).

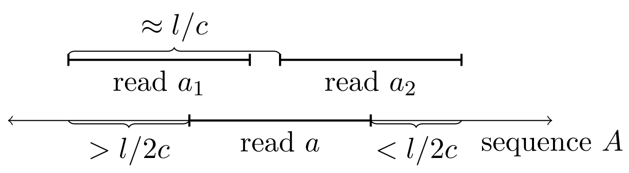


Figure 5.4: An illustration to the reasoning in Section 5.1.4.

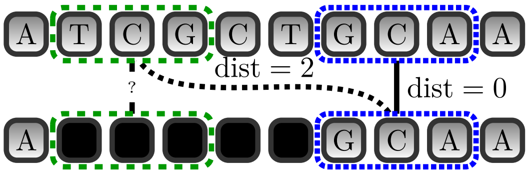


Figure 5.5: If the distance between a read and its closest counterpart is greater than threshold θ, we assume that the read matches a gap in the sequence alignment.

(5.4) is to realize that reads in a read bag are approximately $\frac{l}{c}$ positions from each other. Consider matching read a to reads from R_A. If there is read $a_1 \in R_A$ requiring a gap larger than $\frac{l}{2c}$ to match a, then there will typically be another read $a_2 \in R_A$ requiring gap at most $\frac{l}{2c}$ (see Figure 5.4). From (5.3) and (5.4) we see that this method is applicable only when $c > (\frac{1}{l} + 1)^{-1}$, which is a bit less than 1. However, the results start to be nonzero for $c > (\frac{1}{l} + 2)^{-1}$, which is a bit less than 0.5.

$\mathsf{Dist}_{\mathsf{MESS}}$ equipped with the margin gap technique is denoted $\mathsf{Dist}_{\mathsf{MESSG}}$. The modification to the Levenshtein distance defined in this section will be denoted $\mathsf{dist}_{\mathsf{rr}}$.

Implementing the grace margin gaps in function $\mathsf{Dist}_{\mathsf{MESSG}}$ requires only a small change to the standard Wagner-Fischer algorithm [296]. When the algorithm is filling the first or the last row and column of the table, margin gaps up to t symbols are not penalized. Larger margin gaps are penalized in a way that satisfies the constraint that the distance between string a and an empty string is $|a|$. In particular, the standard linear gap penalty is replaced with a piecewise linear function that gives the cost of margin gap at x-th position

$$g(x) = \begin{cases} 0, & \text{if } 0 \le x \le t-1, \\ 2\frac{x-t+1}{l+1-2t}, & \text{if } t-1 < x \le l-t, \\ 2, & \text{if } l-t < x < l. \end{cases} \tag{5.5}$$

5.1.5 Missing Reads

[239]

Sometimes there is no good match for read a in R_B. During evolution, the substring that contained a may have been inserted into A or may have vanished from B. Therefore, if

$$\text{dist}(a, b) \ge \theta$$

for some reads a and b, and threshold θ, we consider a and b to be dissimilar, and we force their distance to be l. (See Figure 5.5.)

The threshold θ should be a linear factor of the maximal distance between two sequences of length l, i.e., $\theta = \theta' \cdot l$. The value of θ' should reflect the probability that the read is in one sequence and not in the other. Because the actual probability is hidden, it needs to be determined empirically.

The distance function equipped with the missing read detection as described gives rise to the version denoted as $\mathsf{Dist}_{\mathsf{MESSGM}}$.

5.2 Theoretical Analyses

5.2.1 Asymptotic Complexity [239]

Calculating $\text{dist}(A, B)$ for sequences A and B requires $\Theta(|A||B|)$ operations if we use the standard Wagner-Fischer dynamic programming algorithm [296]. This algorithm also requires $\Theta(\min(|A|, |B|))$ memory as we are interested only in the distance and not in the alignment. To calculate $\mathsf{Dist}_{\mathsf{ME}}$, we need to know the distances between all pairs of reads, so we have to evaluate (see (2.11)) $\frac{c}{l}|A|\frac{c}{l}|B|$ distances where each one requires l^2 operations. Therefore, $c^2|A||B|$ operations are required. For the symmetric version, $\mathsf{Dist}_{\mathsf{MES}}$ we make $2c^2|A||B|$ operations, which can be reduced to $c^2|A||B|$ operations and $\Theta(l + \frac{c}{l}(|A| + |B|))$ memory. Further modifications (MESS, MESSG, MESSGM) do not change the asymptotic complexity.

Sampling, as will be described in Section 10.1 reduces runtime by c^2 factor to $\Theta(|A||B|)$, assuming that the final coverage is a small constant. Method MESSGq (see Section 10.2) does not give any theoretical guarantee on the number of pairwise edit distances that need to be explored. However, assuming this number to be a small constant yields the runtime of $\Theta(|R_A||R_B| + l^2(|R_A| + |R_B|))$.

Constants c and l are determined by the sequencing technology, and the independent complexity factors are $|A|$ and $|B|$. To calculate the distance in the conventional way as $\text{dist}(\tilde{A}, \tilde{B})$ requires reconstructing $\tilde{A}$ and $\tilde{B}$ from the respective read bags through an assembly algorithm. This is an NP-hard problem that becomes non-tractable for large $|A|$ and $|B|$, and which is avoided by our approach.

5.2.2 Metric Properties [239]

Distance $\mathsf{Dist}_{\mathsf{MES}}$ as well as the subsequent versions are all symmetric and non-negative, but none of the proposed versions satisfies the identity condition ($\text{dist}(a, b) = 0$ iff $a = b$) nor the triangle inequality, despite being based on the Levenshtein distance dist, which is a metric. For example, let $R_A = \{\mathsf{ATC}, \mathsf{ATC}, \mathsf{GGG}\}$, let $R_B = \{\mathsf{ATA}, \mathsf{GGG}\}$, and let $R_C = \{\mathsf{CTA}, \mathsf{GGG}\}$. Then $\mathsf{Dist}_{\mathsf{MES}}(R_A, R_B) = \frac{7}{12}$, and $\mathsf{Dist}_{\mathsf{MES}}(R_B, R_C) = \frac{1}{2}$ but $\mathsf{Dist}_{\mathsf{MES}}(R_A, R_C) = \frac{14}{12} > \frac{7}{12} + \frac{1}{2}$. While this might lead to a counter-intuitive behavior of the proposed distances in certain applications, the violated conditions are not requirements assumed by clustering algorithms. See Section 3.4.3 for more details about the behavior of hierarchical clustering algorithms when metric properties are not met.

Chapter 6

Monge-Elkan Distance p-value

In most bioinformatics tools, the p-value is the desired output for a statistical measure. The $\mathsf{Dist}_{\mathsf{MESG}}$ distance can be viewed as a measure of how two sequences are similar. Knowing that the distance is, say, 10 means that the sequences are more similar than in the case when the distance is 20. However, can we say that distance 10 means similar sequences A and B? The p-value formalism allows us to get statistical insights on the similarity of the sequences. We might look at all possible sequences A and B and say for how many of them the distance was less than 10. This value is called the p-value and tells us the probability of getting a more extreme value of the statistic (i.e., more similar sequences) only by chance.

6.1 Null Distribution of the Levenshtein Distance

Calculation of the p-value in a closed-form for $\mathsf{Dist}_{\mathsf{MESG}}$ is not known to us. Moreover, to our best knowledge, the closed-form for the p-value of the simplest component of the measure, the Levenshtein distance dist, has yet to be discovered today. The closest reference we could find was a question on the StackExchange forum [109] with a recommendation to use an empirically calculated null distribution of the Levenshtein distance. In addition, a connection to a related Longest Common Subsequence problem [45] is mentioned with the fact that the null distribution in this problem follows the Tracy-Widom distribution [186]. Therefore, we will assume that this distribution is known (for example, calculated empirically) and use this knowledge to build an algorithm that calculates the p-value for a distance value obtained by distance $\mathsf{Dist}_{\mathsf{MESG}}$ in Section 5.1.4.

If we use the Levenshtein distance, the null distribution is discrete with the domain of $\{0, 1, \ldots, l\}$. An example with read length $l = 100$ is in Figure 6.1. If we use the modified Levenshtein distance $\mathsf{dist}_{\mathsf{rr}}$, real distance values are possible; however, their number is still small and finite. Alternatively, discretization might be used. An example with read length $l = 100$ is in Figure 6.2. Both Figures 6.1 and 6.2 were generated under the assumption that the sequences are random, i.e., each of the nucleotides has probability $\frac{1}{4}$ on any position. This case corresponds to the Jukes-Cantor model [140]. A similar distribution can be calculated for any model that generates the sequences. One can model the genomic sequences by a Markov model of the first order and get the respective probability distributions. Alternatively, the null distribution might be constructed from an extensive database of genomic sequences or real-world reads.

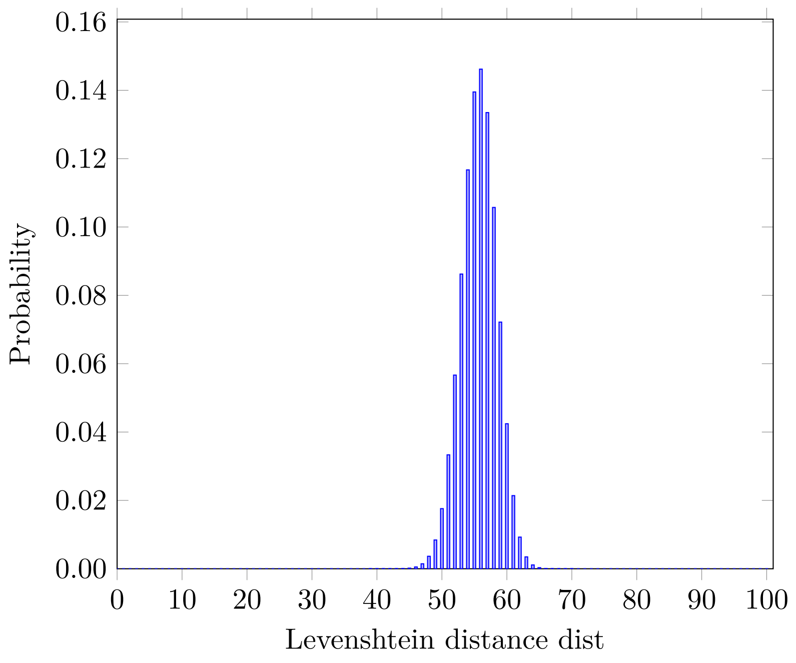


Figure 6.1: The null distribution of the Levenshtein distance dist. The distribution was calculated empirically for 10^8 trials assuming random sequences of length $l = 100$ generated by the Jukes-Cantor model [140]. Nevertheless, distances smaller than 38 and bigger than 70 were never registered as their probability is very low. The zero probabilities in the calculation could be dealt with using the Laplace smoothing (explained, for example, in textbook [236] as add-one smoothing).

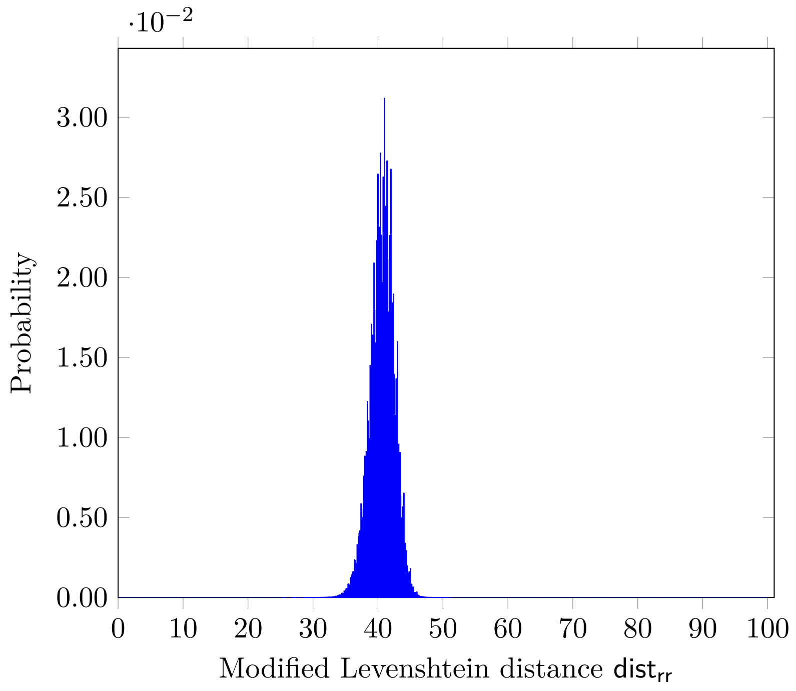


Figure 6.2: The null distribution of the modified Levenshtein distance $\mathsf{dist}_{\mathsf{rr}}$ as used in $\mathsf{Dist}_{\mathsf{MESG}}$ (see Section 5.1.4). The distribution was calculated empirically for 10^8 trials assuming random sequences of length $l = 100$ generated by the Jukes-Cantor model [140]. The histogram was calculated with the box size equal to 0.1. The distribution has a smaller mean than in the case of the Levenshtein distance, as seen in Figure 6.1.

6.2 Null Distribution of the Minimum Operation

Having the null distribution for the Levenshtein distance (dist, or alternatively its modification $\mathsf{dist}_{\mathsf{rr}}$), we can calculate the null distribution for $\min_{b \in R_B} \operatorname{dist}(a, b)$ for a single fixed random read a. To do that, we need a rather strong assumption of independence of all reads, which we know is not true. This is because the reads in R_B come from the same sequence. However, with the assumption that the read length is much smaller than the sequence length, i.e., $l \ll |B|$, we can neglect the error caused by violating this assumption.

Theorem 6.1 *Let Ω be a finite, totally ordered set. Let $p : \Omega \mapsto [0, 1]$ be a probability distribution of a discrete random variable. Suppose that S is a bag of i.i.d. samples taken from Ω with replacement. Then function $q : \Omega \mapsto [0, 1]$ defined as*

$$q(\omega) = \sum_{i=1}^{|S|} \binom{|S|}{i} \cdot p(\omega)^i \cdot \left(\sum_{\{\omega' \in \Omega | \omega' > \omega\}} p(\omega') \right)^{|S|-i} \tag{6.1}$$

represents the probability distribution of $\min S$.[1] [241]

Proof Suppose that $\min S = \omega$. Then at least one element in S equals ω. Denote i the number of elements in Ω equal to ω. There are $\binom{|S|}{i}$ possible choices to select those and each of the minimums has probability $p(\omega)$. The remaining elements in S have to be bigger than ω, termed in the right multiplicand. The overall result is obtained by summing over all possible values of i. ∎

Corollary 6.2 Suppose that the distance calculations of $\operatorname{dist}(a, b)$ are random, i.i.d. for fixed a. Then

$$P\left(\min_{b \in R_B} \operatorname{dist}(a, b) = d\right) = \sum_{i=1}^{|R_B|} \binom{|R_B|}{i} \cdot P(\operatorname{dist}(a, b) = d)^i \cdot P(\operatorname{dist}(a, b) > d)^{|R_B|-i}. \tag{6.2}$$

Example 1 *Suppose that $l = 2$ and the null distribution of* dist *is* $P(\text{dist} = 0) = \frac{1}{8}$, $P(\text{dist} = 1) = \frac{5}{8}$, *and* $P(\text{dist} = 2) = \frac{2}{8}$. *Suppose that $|R_B| = 3$. For $d = 0$, Formula (6.2) evaluates to*

$$3 \cdot \frac{1}{8}\frac{7}{8}\frac{7}{8} + 3 \cdot \frac{1}{8}\frac{1}{8}\frac{7}{8} + \frac{1}{8}\frac{1}{8}\frac{1}{8} = \frac{169}{512}.$$

The first addend represents a situation when 0 is considered only a single time in the evaluation of min, *the second when 0 is considered twice, and the third one when all distances in the* min *evaluation are equal to 0. Similarly, in the case of $d = 1$, we obtain $\frac{335}{512}$, and $\frac{8}{512}$ when $d = 2$.*

We have to be careful when applying Corollary 6.2. The distribution of the distance is calculated for fixed a. Consider the situation when universe $U = \{0, 0.1, 0.2, \ldots, 1.0\}$ and the distance metric is the absolute difference between two numbers. Then the null distribution looks different when $a = 0.5$ and $a = 0$. In the first case, the maximum distance is 0.5, and in the second case, 1.0. However, if we align the points on a circle so that the coordinates go from 0.0 to 1.0, i.e., that the distance is defined as $\min\{|a - b|, 1 - |a - b|\}$, then the null distribution of the distance looks the same for any a, and all we need is to have i.i.d. uniformly selected elements in R_B. Another example

[1] Should the last term of the multiplication be in the form of 0^0, then this value is considered 1.

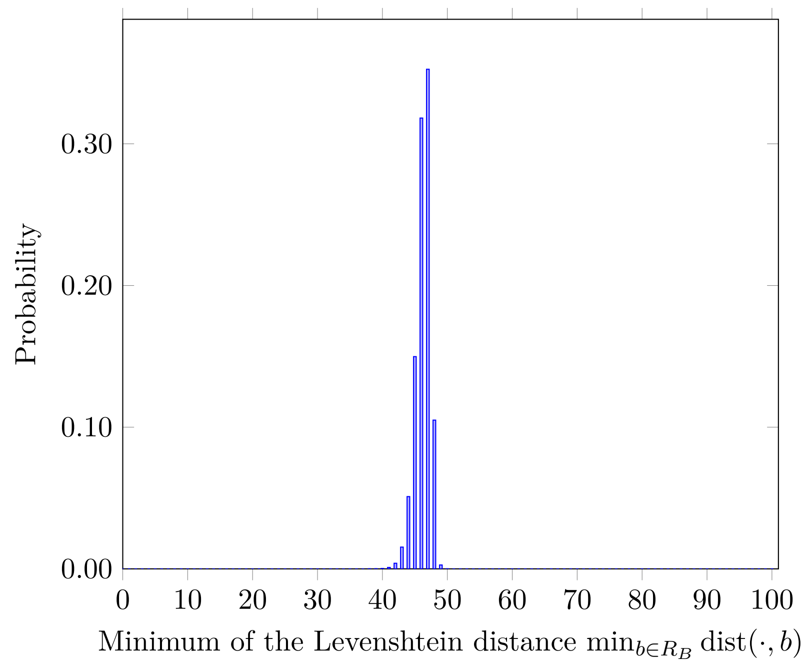


Figure 6.3: The null distribution of the minimum of the Levenshtein distance over read bags of size of 1,000. The underlying distribution of the Levenshtein distance is the same as in Figure 6.1. The distribution is not as symmetric as the original one - the smaller distances are preferred and the minimum being higher than 50 is unlikely but possible.

of a distance with the same null distribution for any selection of a is the Hamming distance [115]. [241]

The Levenshtein distance [167] is another example where the null distribution depends on a. For string AA, there is only one string of distance 1 if A is inserted, but for string CG, there are three strings of distance 1 if A is inserted. The overall effect of this repetition-based discrepancy needs to be assessed experimentally; nevertheless, with larger alphabet sizes and longer sequences, the effect of repeated symbols will be more negligible. [241]

In Figures 6.3 and 6.4, we can see Formula (6.2) applied on distributions from Figures 6.1 and 6.2. We see that the minimum operation skews the distribution towards the smaller values. The probability that all of the 1,000 distances in a sample are bigger than the mean of the original distribution is very low.

6.3 Generating Polynomials

For the p-value calculation, the multiplicative terms before the sum in (5.1) play no role. Therefore, the task of calculating the p-value of (5.1) is equivalent to calculating the p-value only for term

$$\underset{\mathsf{MESG}}{\mathsf{Dist}}'(R_A, R_B) = \underset{\mathsf{ME}}{\mathsf{Dist}}(R_A, R_B) + \underset{\mathsf{ME}}{\mathsf{Dist}}(R_B, R_A). \tag{6.3}$$

For now, assume that the distance was calculated without the leading multiplicative terms as stated in the equation above. The second modification to the equation is that we will eliminate the $\frac{1}{|R_A|}$ and $\frac{1}{R_B}$ terms by multiplying with $|R_A||R_B|$. This

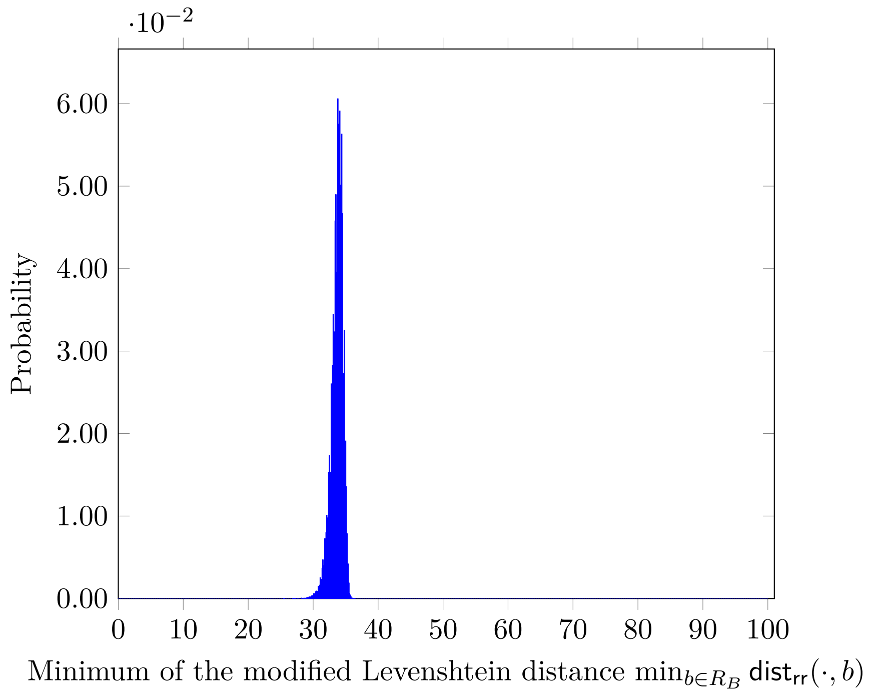


Figure 6.4: The null distribution of the minimum of the modified Levenshtein distance $\mathsf{dist}_{\mathsf{rr}}$ (see Section 5.1.4) over read bags of size of 1,000. The underlying distribution of the Levenshtein distance is the same as in Figure 6.2.

multiplication does not affect the p-value calculation. The formula becomes

$$|R_B| \sum_{a \in R_A} \min_{b \in R_B} \operatorname{dist}(a, b) + |R_A| \sum_{b \in R_B} \min_{a \in R_A} \operatorname{dist}(a, b). \tag{6.4}$$

For now, consider the simpler case when dist is the Levenshtein distance. In this case, the possible distance values are from the set $\Omega = \{0, 1, \ldots, l\}$. To evaluate the p-value, we can employ the concept of generating polynomials, which is a well-established concept in probability theory. The reader is referred to [88] for more details. [241]

Definition 6.3 (Probability-generating function) *Let Ω be a finite subset of $\mathbb{R}$. Let $p : \Omega \mapsto [0, 1]$ be a probability distribution of a discrete random variable. Then the* ***probability-generating function*** *is*

$$\operatorname{genpoly}_p(x) = \sum_{\omega \in \Omega} p(\omega) \cdot x^{\omega}. \tag{6.5}$$

Usually, the probability-generating function is defined only when Ω is a subset of non-negative integers. However, for our application, any finite subset of real numbers is admissible. [241]

Example 2 *For now, we will illustrate the usage of the probability-generating functions in a simple example. Consider tossing a biased dice that can produce three outcomes – 1, 2, and 3. The first one has a probability of $\frac{1}{6}$, the second one $\frac{1}{3}$, and the probability of 3 is equal to $\frac{1}{2}$. Then the generating polynomial is*

$$\tfrac{1}{6}x^1 + \tfrac{1}{3}x^2 + \tfrac{1}{2}x^3. \tag{6.6}$$

We immediately see the probability of an outcome as a coefficient of the respective power of x. Now, let's see what happens when we toss the dice twice and sum the numbers.

For example, the sum of 4 *can be reached by having the first dice with* 1 *and the second with* 3 *with probability of* $\frac{1}{6} \cdot \frac{1}{2} = \frac{1}{12}$*, both dices showing* 2 *with probability* $\frac{1}{9}$*, or, finally,* 3 *and* 1 *with probability* $\frac{1}{12}$*. The probability of seeing* 4 *is, therefore,* $2 \cdot \frac{1}{12} + \frac{1}{9} = \frac{10}{36}$*. The previous calculation is exactly what happens to the coefficients if we multiply two polynomials. Consider the second power of the polynomial in* (6.6)

$$\tfrac{1}{36}x^2 + \tfrac{4}{36}x^3 + \tfrac{10}{36}x^4 + \tfrac{12}{36}x^5 + \tfrac{9}{36}x^6. \tag{6.7}$$

We can notice that the power by 4 *is equal to* $\frac{10}{36}$*. To sum it up, the probability-generating function allows an easy calculation of the null distribution for a sum of independent variables.* [241]

Having this intuition at hand, we can define the generating polynomial for $a \in R_A$ in the calculation of $\min_{b \in R_B} \operatorname{dist}(a, b)$ as

$$\operatorname{genpoly}_a(x) = \sum_{d \in \{0,1,\ldots,l\}} P\left(\min_{b \in R_B} \operatorname{dist}(a, b) = d\right) \cdot x^d. \tag{6.8}$$

We can notice that if we would like to know the probability that the minimum of distances for random read a is equal to d, we only need to read the coefficient of x^d. I.e., the constant term is equal to the probability of the distance minimum being equal to zero; the leading term is equal to the probability of the distance minimum being equal to l.

Example 3 *In our running example (see Example 1), the generating polynomial is equal to*

$$\operatorname{genpoly}_a(x) = \frac{169}{512}x^0 + \frac{335}{512}x^1 + \frac{8}{512}x^2. \tag{6.9}$$

We immediately see that probability that $\min_{b \in R_B} \operatorname{dist}(a, b) = 0$ *is the coefficient of* x^0*, which is equal to* $\frac{169}{512}$*.*

6.4 Null Distribution of the Sum

Having defined the generating polynomial for the min term, we can define the generating polynomial for the whole sum in Equation (6.4). We need to account for the different weights of the left summand and right summand, i.e., the multiplicative term of $|R_A|$ (or $|R_B|$) in front of the minimum operator. To do so, we might use $\operatorname{genpoly}_a(x^{|R_B|})$ in the final calculation instead of $\operatorname{genpoly}_a(x)$, yielding

$$\operatorname{genpoly}_{\mathsf{MESG'}}(x) = \left(\operatorname{genpoly}_a(x^{|R_B|})\right)^{|R_A|} \cdot \left(\operatorname{genpoly}_b(x^{|R_A|})\right)^{|R_B|}. \tag{6.10}$$

Theorem 6.4 allows us to calculate the p-value.

Example 4 *Assume that besides definitions from Example 1, we know that* $|R_A| = 2$*. Then we can define, analogously to* (6.9)

$$\operatorname{genpoly}_b(x) = \frac{15}{64}x^0 + \frac{45}{64}x^1 + \frac{4}{64}x^2. \tag{6.11}$$

Using the Formula (6.10)*, we can compute that*

$$\begin{aligned}\mathrm{genpoly}_{\mathsf{MESG}}(x) &= \left(\frac{169}{512}x^0 + \frac{335}{512}x^3 + \frac{8}{512}x^6\right)^2 \cdot \left(\frac{15}{64}x^0 + \frac{45}{64}x^2 + \frac{4}{64}x^4\right)^3 \\ &= \frac{1}{512^2 \cdot 64^3}\Big(96393380 + 867540400x^2 + 382151200x^3 + \\ &\quad +2679736000x^4 + 3439361000x^5 + 3453195000x^6 + \\ &\quad +10623800000x^7 + 4205565000x^8 + 12170500000x^9 + \\ &\quad +10844910000x^{10} + 2995825000x^{11} + 12336800000x^{12} + \\ &\quad +747478800x^{13} + 2877468000x^{14} + 582508700x^{15} + \\ &\quad +254251400x^{16} + 134107200x^{17} + 14224260x^{18} + 11577600x^{19} + \\ &\quad +1601280x^{20} + 343040x^{21} + 138240x^{22} + 4096x^{24}\Big)\end{aligned} \tag{6.12}$$

In the following theorem, we will show that

$$P\left(\underset{\mathsf{MESG}}{\mathsf{Dist}}'(R_A, R_B) = 3\right) = \frac{382151200}{512^2 \cdot 64^3}, \tag{6.13}$$

i.e., that the coefficients of polynomial $\mathrm{genpoly}_{\mathsf{MESG}'}(x)$ *match to the null distribution of* $\mathsf{Dist}'_{\mathsf{MESG}}(R_A, R_B)$ *under the assumption of independence.*

6.5 The p-value

Theorem 6.4 *Suppose that* a_d *is the coefficient by* x^d *of the polynomial* $\mathrm{genpoly}_{\mathsf{MESG}'}(x)$ *in* (6.10)*. Let* $\mathsf{Dist}'_{\mathsf{MESG}}$ *be used in the form of* (6.4)*. Then*

$$P\left(\underset{\mathsf{MESG}}{\mathsf{Dist}}(R_A, R_B) = d\right) = a_d, \tag{6.14}$$

under the assumption that the addends of the sum in (6.4) *are independent.*

Proof Polynomial $\mathrm{genpoly}_{\mathsf{MESG}}$ is constructed by Formula (6.10). By the expansion of the product, we get $(l+1)^{|R_A|} \cdot (l+1)^{|R_B|}$ terms, each of them in the form

$$\begin{aligned} &P\left(|R_B| \min_{b \in R_B} \mathrm{dist}(\cdot, b) = d_1\right) \cdot x^{d_1} \cdot P\left(|R_B| \min_{b \in R_B} \mathrm{dist}(\cdot, b) = d_2\right) \cdot x^{d_2} \cdots \\ &\quad \cdot P\left(|R_B| \min_{b \in R_B} \mathrm{dist}(\cdot, b) = d_{|R_A|}\right) \cdot x^{d_{|R_A|}} \cdot \\ &\cdot P\left(|R_A| \min_{a \in R_A} \mathrm{dist}(a, \cdot) = d_{|R_A|+1}\right) \cdot x^{d_{|R_A|+1}} \cdot \\ &\quad \cdot P\left(|R_A| \min_{a \in R_A} \mathrm{dist}(a, \cdot) = d_{|R_A|+2}\right) \cdot x^{d_{|R_A|+2}} \cdots \\ &\quad \cdot P\left(|R_A| \min_{a \in R_A} \mathrm{dist}(a, \cdot) = d_{|R_A|+|R_B|}\right) \cdot x^{d_{|R_A|+|R_B|}}. \end{aligned} \tag{6.15}$$

Each of the powers d_j represents a possible distance selected in sum (6.4). Their combination represents a possibility of the resulting distance being equal to

$$d = \sum_{j=1}^{|R_A|+|R_B|} d_j. \tag{6.16}$$

Also, by the independence assumption, the probability of this particular combination is equal to

$$\prod_{j=1}^{|R_A|} P\left(|R_B| \min_{b \in R_B} \operatorname{dist}(\cdot, b) = d_j\right) \cdot \prod_{j=|R_A|+1}^{|R_A|+|R_B|} P\left(|R_A| \min_{a \in R_A} \operatorname{dist}(a, \cdot) = d_j\right). \tag{6.17}$$

Again, we use the independence assumption in a form that is not strictly true. We have a matrix of random variables and calculate row minimums and column minimums. Even if, in each row, the values were independent (up to a common distribution that the distances were sampled from), the minimums in each row and column would be dependent. Nevertheless, we assume they are independent. However, we can argue that the matrix is large enough (i.e., the sizes of R_A and R_B are in order of hundreds or thousands) so that the effect can be neglected.

The probability $P\left(\mathsf{Dist}'_{\mathsf{MESG}}(R_A, R_B) = d\right)$ is equal to the sum of probabilities of all possible combinations that fulfill Equation (6.16). The probabilities are given by (6.17), which is, in turn, equal to a_d. ■

Corollary 6.5 Suppose that

$$\operatorname{genpoly}_{\mathsf{MESG}'}(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n. \tag{6.18}$$

Then the p-value of $\mathsf{Dist}'_{\mathsf{MESG}}(R_A, R_B) = d$ is equal to

$$\frac{\sum_{i=0}^{d} a_i}{\operatorname{genpoly}_{\mathsf{MESG}'}(1)} = \sum_{i=0}^{d} a_i. \tag{6.19}$$

Example 5 *Using the result from Equation* (6.12)

$$\operatorname{genpoly}_{MESG'}(x) = \frac{1}{512^2 \cdot 64^3}\Big(96393380 + 867540400x^2 + 382151200x^3 + \\ +2679736000x^4 + 3439361000x^5 + \cdots\Big), \tag{6.20}$$

we can calculate that the p-value of $\mathsf{Dist}'_{MESG}(R_A, R_B) = 3$ *is equal to (up to an error caused by violating the independence assumptions)*

$$\frac{1}{512^2 \cdot 64^3}(96393380 + 0 + 867540400 + 382151200) \doteq 0.020. \tag{6.21}$$

6.6 Runtime Requirements

At first sight, it may seem that the calculation of the p-value is too slow as a naive implementation of the result (6.19) requires $(l+1)^{|R_A|+|R_B|}$ multiplications in the polynomial evaluation. However, this number can be reduced by two tricks. The first one is that we need only coefficients of the polynomial only up to calculated distance d (as of Algorithm 6.1). Figures 6.3 and 6.4 show that the distribution is slightly skewed to the left, with the expected value around 50 and 33, meaning that we can expect the runtime to improve two to three times. Those distances were generated assuming random sequences; for real world-data, the runtime improvements are likely to be higher as real-world sequences should be more similar than random sequences.

The second trick is based on fast power calculation for large numbers. Suppose we are about to calculate the n-th power of number x. Then, we do not need to do n consecutive multiplications. Instead, we calculate powers of $x^1, x^2, x^4, \ldots, x^{\lfloor \log_2 n \rfloor}$

and multiply the respective powers to obtain x^n. As a result, we need only $2 \cdot \log n$ multiplications. [241]

Applying the same trick to our problem of polynomial power decreases the time complexity so that there are only $2\log_2|R_A| + 2\log_2|R_B|$ polynomial multiplications needed. Each polynomial multiplication takes at most d^2 using a naive implementation. Better results can be obtained if we employ the Fast Fourier Transform for polynomial multiplication, which can be done in $\mathcal{O}(d\log d)$ [36], yielding the overall run time of $\mathcal{O}(d\log d \cdot (\log(|R_A| \cdot |R_B|)))$.

The last trick can be used to decrease d without actually influencing the result. This modification might not always be available. In Formula (6.4), we might still divide the sum by $\gcd(|R_A|, |R_B|)$. The asymptotically same effect can be obtained by using hashing. If the greatest common divisor is equal to one or is small, there is still room for an approximation. We might slightly change the read bag sizes so that the result is not influenced much. For example, the ratio of bag sizes 1000 to 1201 is close to 5 to 6, resulting in almost 200 times smaller value of d and runtime by the order of 3 magnitudes faster.

```
function MESG-DISTANCE($R_A$, $R_B$)
    $d = \frac{|R_B|}{\gcd(|R_A|,|R_B|)} \sum_{a\in R_A} \min_{b\in R_B} \operatorname{dist}(a,b) + \frac{|R_A|}{\gcd(|R_A|,|R_B|)} \sum_{b\in R_B} \min_{a\in R_A} \operatorname{dist}(a,b)$

    $P(\operatorname{dist}(\cdot,\cdot)) \leftarrow$ null distribution of dist
    Calculate the null distribution of $\min_{b\in R_B} \operatorname{dist}(\cdot,b)$ and $\min_{a\in R_A} \operatorname{dist}(a,\cdot)$ using Formula (6.2)
    Use this null distribution to construct $\operatorname{genpoly}_a$ and $\operatorname{genpoly}_b$ using Formula (6.10)
    Calculate $\left(\operatorname{genpoly}_a(x^{\frac{|R_B|}{\gcd(|R_A|,|R_B|)}})\right)^{|R_A|} \cdot \left(\operatorname{genpoly}_b(x^{\frac{|R_A|}{\gcd(|R_A|,|R_B|)}})\right)^{|R_B|}$
    $p$-value is equal to the sum of coefficients of this polynomial up to $x^d$
    return $d \cdot \frac{\gcd(|R_A|,|R_B|)}{|R_A||R_B|} \cdot \frac{1}{2}\max\{|R_A|,|R_B|\}$ and the $p$-value
end function
```

Algorithm 6.1: The p-value algorithm.

6.7 A General Case - the p-value of the Monge-Elkan Distance [241]

It is useful to have an algorithm to approximate the p-value for the method developed in this thesis. However, from the point of possible future applicability, it is much more important to develop the p-value algorithm for the general case. We did so in paper [241], whose contents will be presented in this section with some minor modifications.

The Monge-Elkan similarity was proposed in [200]. The paper used the concept to solve the *field matching problem*, i.e., the problem of deciding whether two different fields (say, strings) represent the same entity. For example, *John Doe* and *Doe, John* are likely to represent the same person despite different textual representations. The field matching problem often arises in databases when multiple data sources are combined into a single one.

Due to the simplicity of the idea, it has found its way into many applications, especially in the approximate string matching field and related tasks such as name matching [49], information extraction from health records [144], title matching [100], business process model matching [2], or toponym matching in geography [253]. There are also generalizations of the original paper as [138].

The Monge-Elkan similarity found its way into bioinformatics as well. Besides the usage for read bag comparison in our works [237, 239], it is worth mentioning the identification of duplicate biological entities in bioinformatical databases [273], or the ontology alignment problem [277, 40].

6.7.1 Definition of the Problem

Let U be a universe of elements. Let dist be a distance function (for purposes of this section, not necessarily the Levenshtein distance) on the universe with a finite range $Y \subset \mathbb{R}$, i.e., $\mathrm{dist} : U \times U \mapsto Y$ where $|Y| < \infty$. Recall the Monge-Elkan distance (5.1), for which we will calculate the p-value.

The Monge-Elkan distance has the *null distribution* if all elements in $R_A \cup R_B$ are sampled i.i.d. with replacement from U while keeping bag sizes $|R_A|$ and $|R_B|$. The alternative hypothesis states that the bags are similar. Given a value of $\mathsf{Dist}_{\mathsf{ME}}(R_A, R_B)$, we call the alternative hypothesis if the probability of obtaining that value or smaller under the null hypothesis is smaller than a threshold.

6.7.2 The p-value Calculation

The innermost part of the calculation in (5.1) is the dist distance. The null distribution of dist is assumed to be given. We also assume that the distance has a finite range as defined in Section 6.7.1.

Once we have the null distribution of the minimum operation (by exploiting Theorem 6.1 and Corollary 6.2), we need to evaluate the null distribution of the sum. To this end, we will exploit the *probability-generating function* as in the previous section, and start from the definition of $\mathrm{genpoly}_a(x)$ in Equation (6.8). The polynomial for the whole distance in (5.1) is then the respective power of $\mathrm{genpoly}_a(x)$

$$\mathrm{genpoly}_{\mathsf{ME}}(x) = \left(\mathrm{genpoly}_a(x)\right)^{|R_A|}. \tag{6.22}$$

This polynomial can then be used to calculate the null distribution of the Monge-Elkan distance. However, the calculation will have one condition on independence, similarly to dice tosses in (6.7), that we will discuss later. [241]

Theorem 6.6 *Assume that the probability $P\left(\min_{b \in R_B} \mathrm{dist}(a,b) = d\right)$ is independent of the choice R_B. Then the coefficients of the polynomial $\mathrm{genpoly}_{\mathsf{ME}}(x)$ represent the null distribution of the Monge-Elkan distance up to a multiplicative term of $\frac{1}{|R_A|}$.*[2] *[241]*

Proof Polynomial $\mathrm{genpoly}_{\mathsf{ME}}(x)$ is constructed according to Formula (6.22). By expanding the product, we get $|Y|^{|R_A|}$ terms, each of them in the form

$$P\left(\min_{b \in R_B} \mathrm{dist}(a,b) = d_1\right) \cdot x^{d_1} \cdot \cdots \cdot P\left(\min_{b \in R_B} \mathrm{dist}(a,b) = d_{|R_A|}\right) \cdot x^{d_{|R_A|}}. \tag{6.23}$$

Each of the powers d_j represents a possible distance selected by the min operator in the sum in (5.1). Their combination represents a possibility of the resulting sum being equal to

$$d = \sum_{j=1}^{|R_A|} d_j. \tag{6.24}$$

[2]In other words, if a_d is the coefficient of x^d in the polynomial $\mathrm{genpoly}_{\mathsf{ME}}$, then $P\left(\mathsf{Dist}_{\mathsf{ME}}(R_A, R_B) = \frac{1}{|R_A|} d\right) = a_d$.

Also, by the independence assumption, the probability of this particular combination is equal to

$$\prod_{j=1}^{|R_A|} P\left(\min_{b\in R_B} \operatorname{dist}(a,b) = d_j\right). \tag{6.25}$$

The probability $P\left(\mathsf{Dist}_{\mathsf{ME}}(R_A, R_B) = \frac{1}{|R_A|}d\right)$ is equal to the sum of probabilities of all possible combinations of d_j that fulfill the equation in Formula (6.24). The probabilities of the combinations are in (6.25), the sum of which is, in turn, equal to a_d. ■

Corollary 6.7 Suppose that [241]

$$\operatorname{genpoly}_{\mathsf{ME}}(x) = a_0 x^{d_0} + a_1 x^{d_1} + a_2 x^{d_2} + \cdots + a_n x^{d_n}. \tag{6.26}$$

Then the p-value of $\mathsf{Dist}_{\mathsf{ME}}(R_A, R_B) = d$ is equal to

$$\begin{aligned} P\left(\underset{\mathsf{ME}}{\mathsf{Dist}}(R_A,R_B) \le d\right) &= \sum_{\{d_i\,:\,d_i \le |R_A| d\}} P\left(\underset{\mathsf{ME}}{\mathsf{Dist}}(R_A,R_B) = \frac{1}{|R_A|}d_i\right) \\ &= \sum_{\{d_i\,:\,d_i \le |R_A| d\}} a_i. \end{aligned} \tag{6.27}$$

6.7.3 The Independence Assumption [241]

We have to look in more detail at the independence assumption in Theorem 6.6. The fact that the minimum selection should be independent of the set over which we select the minimum is very restrictive and can be satisfied exactly only for very trivial distances. The calculation in Theorem 6.6 corresponds to the situation when for the first element $a \in R_A$, we generate bag R_B and calculate the summand according to the Monge-Elkan distance (5.1). Then for the second element $a \in R_A$, a *new* set R_B is generated independently, and the summand is calculated. The calculation then follows with the new set R_B for each $a \in R_A$.

However, the set R_B is kept throughout the calculation in the Monge-Elkan distance. Hence, we are making some errors in the p-value calculation. Imagine a space defined by distance function dist. The space looks different in the case where elements R_B are selected randomly uniformly and in the case where all elements in bag R_B are the same. The first case is, however, common, while the second one is unlikely. Therefore, there are still many situations when Theorem 6.6 will be applicable as a reasonable approximation.

6.8 Experimental Evaluation [241]

As the presented approach contains an approximation based on the independence assumption that is not true, the null distribution calculated according to Theorem 6.6 needs to be compared to the ground truth data. To do so, we selected three simple examples of distance functions and evaluated them for various choices of bag sizes as well as different universa. The distances include:

- the Levenshtein distance [167] on binary strings of length l;
- the Hamming distance [115] on binary strings of length l;

- the absolute difference between two numbers from $\{0, 1, 2, \ldots, n-1\}$ aligned on a circle. I.e., $\min\{|a-b|, 1-|a-b|\}$ where a and b are numbers in the respective set.

The parameters l and n were selected so that the null distribution of the Monge-Elkan distance could be calculated by mere enumeration of all possible bags R_A and R_B. For simplicity of presentation, $|R_A|$ was set the same as $|R_B|$. The null distribution was then calculated by enumeration of all possible bags and using Theorem 6.6. Those two distributions were then compared visually as well as using the Kullback-Leibner divergence [155] (the KL divergence, sometimes called the relative entropy). In the KL divergence, the natural logarithm was used.

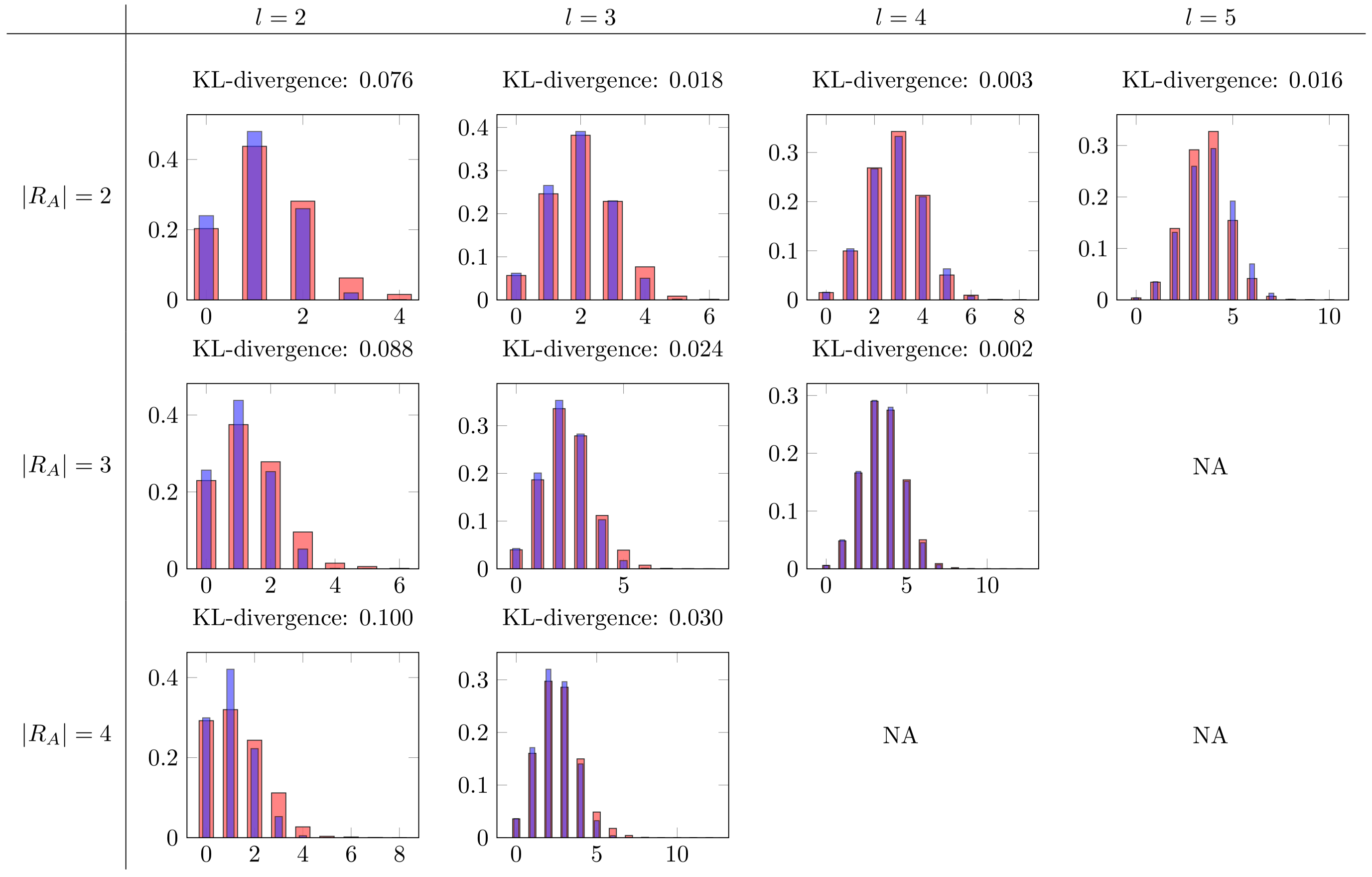
In the case of the string distances, the boundary set for enumeration was $2l|R_A| = 27$, which meant 2^{27} elements in the null distribution at most. In the case of the distance between the numbers, the limit was $n^{2|R_A|} = 10^{10}$, which meant 10^{10} elements in the null distribution at most.

The experimental evaluation is in Figure 6.5. From the figures, we can notice that the KL divergence is growing with larger bag sizes (the independence assumption is more relied on in the multiplication). It might be expected that the KL divergence would decrease with universa of more elements (i.e., higher l or n); however, this trend is supported by the data only in the case of the string distances.

From the plots, we can also notice that the approximation underrepresents the low distances for more cardinal universa U. This is not a desired behavior; however, there remains an open window for future work in modifying the approach so that the p-value cannot be underestimated.

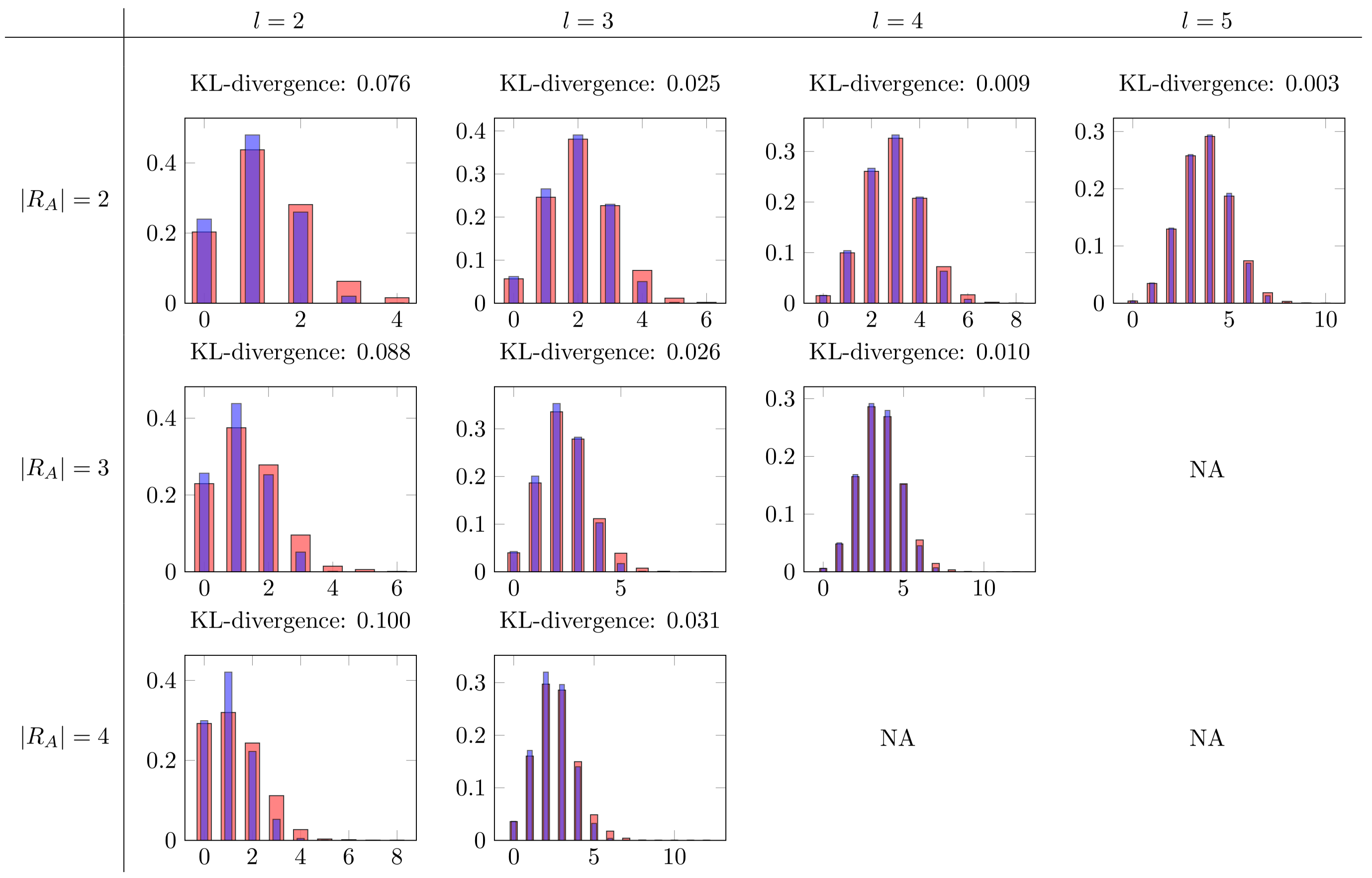
6.8.1 Summary

We have presented an algorithm to estimate the null distribution of the Monge-Elkan distance that can be used to compare the sequence similarity from unassembled read bags. The methodology contains two simplifying assumptions that represent possible sources of error. However, we have empirically confirmed that their detrimental effect is generally not significant. In particular, the KL divergence between the calculated distribution and the one obtained by Monte-Carlo sampling tends to be negligible. The contributed method thus represents a feasible tool that may even become a necessity when Monte-Carlo sampling is intractable due to the slow evaluation of the Monge-Elkan distance.



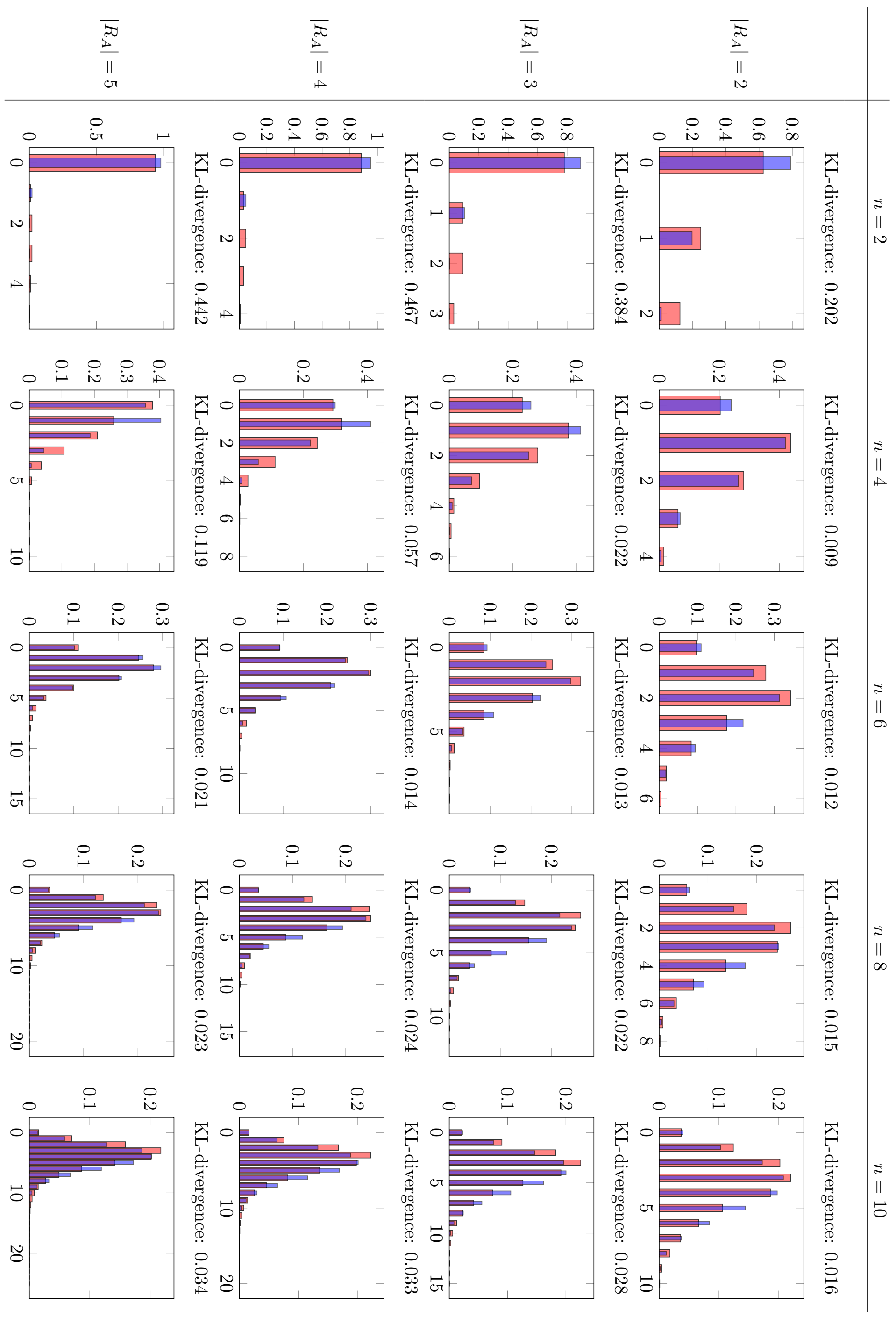
(a) The Levenshtein distance [167] over binary strings of length l.

Figure 6.5: A comparison of the approximated (blue, narrow) and the ground-truth (red, wide) null distribution. The approximated distribution was calculated using Theorem 6.6, while the ground-truth distribution was calculated by enumerating all possible choices of R_A and R_B of the same size. NA means that with the given settings, it was not feasible to enumerate the distribution.



(b) The Hamming distance [115] over binary strings of length l.

Figure 6.5: A comparison of the approximated (blue, narrow) and the ground-truth (red, wide) null distribution (cont.).



(c) $\min\{|a-b|, 1-|a-b|\}$ for $a, b \in \{0, 1, 2, \ldots, n-1\}$.

Figure 6.5: A comparison of the approximated (blue, narrow) and the ground-truth (red, wide) null distribution (cont.).

Chapter 7 Monge-Elkan Distance as a Lower Bound

In this section, we will present an alternative view on the Monge-Elkan distance [200] between the read bags as a lower bound on the Levenshtein distance [167] between the original sequences. The lower bound will hold only up to a certain probability, which we will estimate too pessimistically in a closed form. Then, we will provide a guide to calculate this probability more precisely.

First, we will present a more straightforward problem when only reads for one sequence are generated and the second sequence is known. We will present a lower bound on the Levenshtein distance for this simple scenario. Then we will substitute the known sequence with a bag of reads to extend the lower bound so that it holds with a certain probability.

7.1 A Simple Case - Only One Read Bag

Assume that we are about to estimate $\mathrm{dist}(A, B)$ when sequence B is known, and instead of sequence A, we have read bag R_A. For now, consider that a is a fixed read. We will start with a lower bound using a modified distance between read a and sequence B.

The idea of the Monge-Elkan-distance-based approach presented in Chapter 5 is to find the most similar pairs of reads. Assuming read a and sequence B, we might modify the Levenshtein distance accordingly so that it searches for the part of B that is the most similar to a.

The search can be done by a modification of the Wagner-Fischer algorithm [296]. We replace the penalty for leading or trailing margin gaps in B with a zero penalty. This procedure is similar to what we did in Section 5.1.4, only the margin gap penalty function is different. As a result, the algorithm finds a substring of B that has the lowest distance from query a. Denote this substring B_a. Notation and gap costs are illustrated in Figure 7.1. This modification to the Wagner-Fischer algorithm was presented by Sellers in [257], and the idea is similar to the local search by the Smith-Waterman algorithm [268]. [245]

Denote the distance defined in the former paragraph $\mathsf{dist}_{\mathsf{rS}}$. Formally,

$$\mathsf{dist}_{\mathsf{rS}}(a, B) = \mathrm{dist}(a, B_a) = \min_{b' \in \mathsf{Subseq}(B)} \mathrm{dist}(a, b'). \tag{7.1}$$

We will use a very similar distance to search for reads in contigs in Chapter 8. For a while, we might think about this distance (similarly to the Levenshtein distance) as a path in a graph represented by the cells of the dynamic programming table. We already used this symbolism in Figure 7.1. Each edge of the graph corresponds to an operation allowed by the distance. Each edge is assigned a weight defined by the distance; for

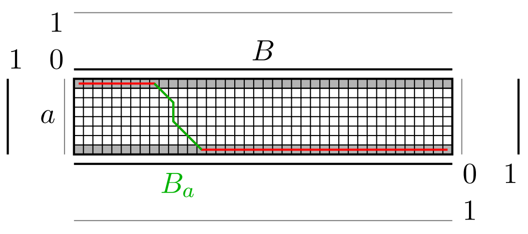


Figure 7.1: A modification of the Wagner-Fischer algorithm [296] for search. In sequence B, the leading and trailing gaps are not penalized. The closest part to read a (denoted B_a) is marked with the green line. The path denotes the $\mathsf{dist}_{\mathsf{rS}}$ alignment between the closest substring of B and read a. [245]

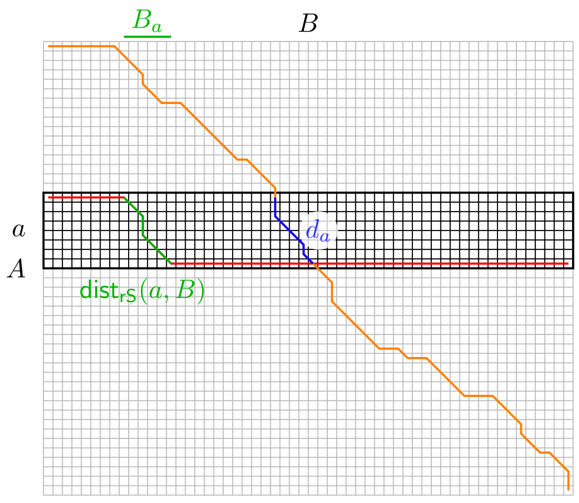


Figure 7.2: An illustration to Lemma 7.1. Out of sequence A, we know only a single read a. The green path represents the distance between read a and sequence B. The lemma then proves that the green path is not shorter than the blue path.

example, the insert operation has a weight of 1. The distance is then the shortest path length in such a graph.

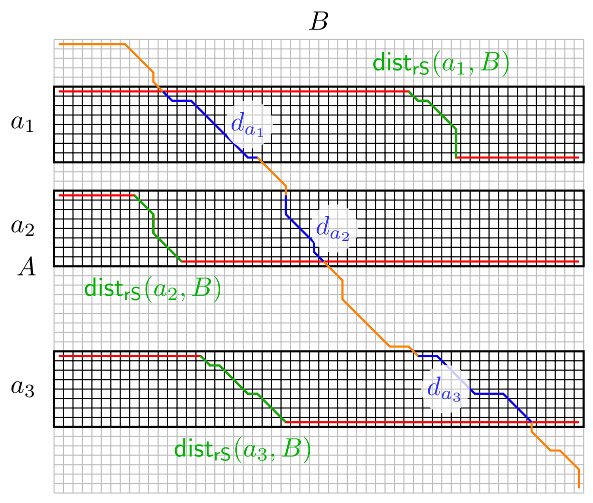
We now construct such a graph with both the Levenshtein distance $\mathrm{dist}(A, B)$ and $\mathsf{dist}_{\mathsf{rS}}(a, B)$. This situation is illustrated in Figure 7.2. We will denote d_a part of the path defined by the Levenshtein distance $\mathrm{dist}(A, B)$ that corresponds to read a. The following lemma will be the first part of the lower bound.

Lemma 7.1 *Let a be a substring of A. Consider the Levenshtein distance of A and B and denote b' substring of B that is aligned to string a. Denote* $\mathrm{dist}(a, b') = d_a$. *It holds that*

$$\mathsf{dist}_{rS}(a, B) \leq d_a. \tag{7.2}$$

Proof Refer to Figure 7.2. Distance $\mathsf{dist}_{\mathsf{rS}}$ selects the minimum from all paths in the alignment graph that start with an unspecified number of gaps, alignment of a to some substring of B, and then end by an unspecified number of gaps. The gaps at the beginning and the end are not penalized. The possible paths for minimum include not only the path corresponding to $\mathsf{dist}_{\mathsf{rS}}(a, B)$ but also a path that starts with penalty-free gaps, continues by the alignment between a and b' of length[1] d_a, and ends with penalty-

[1]A small reminder is in the place. The path length is, in this case, the sum of the costs of the insert,



B
$\mathsf{dist}_{\mathsf{rS}}(a_1, B)$
a_1
d_{a_1}
a_2
d_{a_2}
A
$\mathsf{dist}_{\mathsf{rS}}(a_2, B)$
a_3
d_{a_3}
$\mathsf{dist}_{\mathsf{rS}}(a_3, B)$

Figure 7.3: An illustration to Lemma 7.2. Out of the sequence A, we know several non-intersecting reads. The green paths represent the distances between those reads and sequence B. The lemma then proves that the length of the green paths is not smaller than the length of the orange path (including the blue parts).

free gaps. The length of the path is d_a. As the $\mathsf{dist}_{\mathsf{rS}}$ distance selects by definition the minimum from a set that contains d_a, Equation (7.2) holds. ■

7.1.1 Sum of Multiple Non-intersecting Reads

The existence of a bound for only a single read is not really practical, but we might combine this bound for multiple reads. Recall the definition of intersecting reads from Section 2.5. Suppose that two intervals I_a, and $I_{a'}$ represent indices of nucleotides in A that were sequenced into reads a and a'. If and only if those two intervals do not intersect, the reads do not intersect. We will prove that the lower bounds for the non-intersecting reads can be summed while preserving the inequality. The whole situation is illustrated in Figure 7.3.

Lemma 7.2 *Let R_A be a read bag where no pair of reads intersect. Then*

$$\sum_{a \in R_A} \mathsf{dist}_{\mathsf{rS}}(a, B) \leq \operatorname{dist}(A, B). \tag{7.3}$$

Proof Refer to Figure 7.3. The path representing the Levenshtein distance $\operatorname{dist}(A, B)$ can be split into multiple subpaths, each corresponding to distance d_a for read a (see Lemma 7.1 for the definition of d_a). As reads do not intersect and no edge on the path has a negative weight; we can write

$$\operatorname{dist}(A, B) \geq \sum_{a \in R_A} d_a \geq \sum_{a \in R_A} \mathsf{dist}_{\mathsf{rS}}(a, B) \tag{7.4}$$

which finishes the proof. ■

delete, and substitute operations on the path. I.e., the matched symbols do not count toward the path length.

7.1.2 General Read Bag Case

Lemma 7.2 allows us to sum the distance estimates for non-intersecting reads. This fact is quite limiting, as we usually do not know whether a prefix-suffix overlap between two reads represents an intersection or not. Lemma 7.2 would also limit the method only to a coverage of less than one. However, it turns out that it is possible to sum the read sequence distances for many reads knowing only a single hidden parameter - the *maximum per-base coverage.*

Definition 7.3 (Maximum per-base coverage) *Let $A = A_1A_2 \cdots A_{|A|}$ be a sequence and R_A a read bag sequenced from this sequence. Suppose that $c(i)$ denotes how many reads span over position i in A, i.e., cover nucleotide A_i. Then the* ***maximum per-base coverage*** *c_m is*

$$c_m = \max_{i \in \{1,2,\ldots,|A|\}} c(i). \tag{7.5}$$

When we sequence a genome, the maximum per-base coverage parameter is unknown; however, we might believe that it is not far from average coverage c up to some multiplicative factor.[2] This statement comes from the assumption that the reads are uniformly i.i.d. generated from the set of all possible reads. For now, suppose, therefore, that an oracle provides us with this unknown parameter. Then the following lemma tells us that any read bag can be split into c_m non-intersecting read bags.

[2]The distribution of the maximum per-base coverage can be computed. Coverage $c(i)$ follows the binomial distribution but for an error close to the beginning or end of the sequence and the dependence between the neighboring nucleotides. We will neglect this error. For a fixed position i, the probability that a read of length l spans over this position is $\frac{l}{|A|-l+1}$. The nominator is the length of the read; the denominator is the total number of all reads. Therefore, the probability that $c(i) = k$ is very close to

$$P(c(i) = k) \approx \binom{|R_A|}{k} \left(\frac{l}{|A|-l+1}\right)^k \left(1 - \frac{l}{|A|-l+1}\right)^{|R_A|-k}. \tag{7.6}$$

Using Equation (7.6), we can calculate the probability that $\max_{i \in \{1,2,\ldots,|A|\}} c(i)$ is equal to a specific value. To do so, we might exploit Theorem 6.1 and change the minimum calculation to a maximum calculation. We obtain (with some approximation — the per-base coverage for adjacent nucleotides is dependent, but in the following calculation, we assume independence)

$$P\left(\max_{i \in \{1,2,\ldots,|A|\}} c(i) = c_m\right) = \sum_{j=1}^{|R_A|} \binom{|R_A|}{j} P(c(i) = c_m)^j P(c(i) < c_m)^{|R_A|-j}. \tag{7.7}$$

It is hard to grasp the probability (7.6) by hand. Nevertheless, we might get a result that is easier to interpret. The binomial distribution with parameters n ($|R_A|$ in our case) and p (probability $\frac{l}{|A|-l+1}$) can be approximated by a normal distribution with mean np and variance $np(1-p)$. This fact has been known since 1738 as the de Moivre-Laplace theorem. Assuming that $|A| \gg l$, we might calculate the mean of this normal approximation as

$$|R_A| \cdot \frac{l}{|A|-l+1} \approx \frac{|R_A|l}{|A|} = c. \tag{7.8}$$

Similarly, the variance is

$$|R_A| \cdot \frac{l}{|A|-l+1} \cdot \left(1 - \frac{l}{|A|-l+1}\right) \approx \frac{|R_A|l}{|A|} \cdot (1-0) = c. \tag{7.9}$$

For example, if coverage is equal to 3, then in order to reach coverage $c(i)$ 3 times bigger than the average coverage c, we need deviation from mean of $2 \cdot 3 = (2 \cdot \sqrt{3})\sqrt{3}$. As the standard deviation is $\sqrt{c} = \sqrt{3}$, the z-score needs to be equal to $2 \cdot \sqrt{3}$, which is more than 3.4. This means that we need, on average, 3,000-nucleotide long sequences. Generally, to get the maximum per-base coverage k times bigger than the average, we need approximately $1/p$ long sequence on average, where p is the probability of reaching a z-score of $(k-1)\sqrt{c}$.

Lemma 7.4 *For any read bag R_A, there are c_m read bags $R_{A1}, R_{A2}, \ldots, R_{Ac_m}$ such that for any read bag R_{Ai} ($i \in \{1, 2, \ldots, c_m\}$), no pair of reads in R_{Ai} intersect, and*

$$\bigcup_{i=1}^{c_m} R_{Ai} = R_A. \tag{7.10}$$

Moreover, c_m is the smallest number of read bags that fulfills this condition.

Proof We will use the concept of graph coloring, known from the graph theory field. Any read from R_A will form a vertex. Two reads will be connected by an edge if and only if they cover the same position in sequence A, i.e., they intersect. Then, the problem of splitting reads is equivalent to coloring the vertices (reads in R_A) with c_m colors, as no two reads covering the same position can be colored with the same color. Denote such a graph G.

The reads that span a nucleotide with the maximum per-base coverage form a c_m-clique in G. Therefore, the chromatic number $\chi(G)$ is at least c_m

$$\chi(G) \geq c_m. \tag{7.11}$$

By the algorithm of *sequential coloring* [304] (sometimes called *greedy coloring*, or extended as *Brook's theorem* [31]), we know that

$$\chi(G) \leq (\text{the maximum degree of a vertex in } G) + 1 = c_m. \tag{7.12}$$

Equations (7.11) and (7.12) prove that G can be colored with c_m colors but not less. Hence, we can treat the colors as sets R_{Ai}, which proves the lemma. ■

The coloring mentioned in the proof can be found by the sequential coloring algorithm (sometimes called the Welsch-Powell algorithm [304]) - we order reads by their position in A and assign each of them a color from a set of size c_m. Read a is colored by the first color that is not used by any reads that intersect with a and start earlier in the sequence. By applying this algorithm, we can notice that we need at least c_m colors to color the reads. Moreover, c_m colors are sufficient as there will always be at least one color available unless the per-base coverage is greater than c_m.

In the following theorem, we will prove that we actually do not need to know the exact split of the reads into the bags of non-intersecting reads; we will need only the maximum coverage parameter.

Theorem 7.5 *For any sequences A, B and bag of reads R_A holds*

$$\mathrm{dist}(A, B) \geq \frac{1}{c_m} \sum_{a \in R_A} \mathsf{dist}_{rS}(a, B). \tag{7.13}$$

Proof By Lemma 7.4, R_A can be split to c_m read bags $R_{A1}, R_{A2}, \ldots, R_{Ac_m}$ such that within any of those, the reads are non-intersecting. We can apply Lemma 7.2 to each of those read bags individually, yielding a set of equations

$$\sum_{a \in R_{Ai}} \mathsf{dist}_{\mathsf{rS}}(a, B) \leq \mathrm{dist}(A, B), \tag{7.14}$$

where $i \in \{1, 2, \ldots, c_m\}$. By summing the c_m equations in (7.14), we obtain

$$\sum_{i \in \{1,2,\ldots,c_m\}} \sum_{a \in R_{Ai}} \mathsf{dist}_{\mathsf{rS}}(a, B) \leq c_m \cdot \mathrm{dist}(A, B). \tag{7.15}$$

By applying condition (7.10), we can use only a single sum

$$\sum_{a \in R_A} \mathsf{dist}_{\mathsf{rS}}(a, B) \leq c_m \cdot \mathrm{dist}(A, B). \tag{7.16}$$

Dividing this equation by c_m finishes the proof. ■

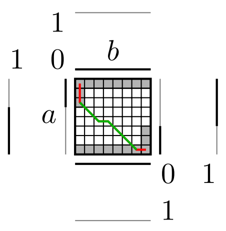


Figure 7.4: In the case of read data, we search for a only in b, which is a randomly selected substring of B. Because the length of the alignment of B_a and a may be longer than l, we do not penalize the first t leading or trailing gaps in a. In the figure, $t = 2$.

Formula (7.13) has a straightforward connection to the Monge-Elkan distance [200] we use. In our case, the read-sequence part $\mathsf{dist}_{\mathsf{rS}}$ is approximated by the minimum distance between all reads. Instead of calculating the optimum path as visualized in Figure 7.1, many reads represent sequence B, and hopefully, one of them is close enough to the closest B_a. The free margin gap penalty, which we introduced in Section 5.1.4, solves the shift and length difference. The idea of approximating $\mathsf{dist}_{\mathsf{rS}}$ by sampling reads will be extended in the next section; we will try to develop a pessimistic estimate of the probability that this bound works.

7.2 A Pessimistic Bound for Two Read Bags

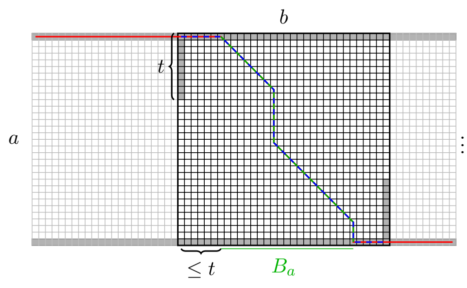
In the past section, we developed the bound for a single bag of reads while the first sequence was known. Now, we extend the ideas from the previous section into a situation solved in the thesis when there are two read bags. The key difference is in Equation (7.2) that will hold only with a certain probability. First, we will estimate this probability based on the location of R_B reads in B. This section will present the most pessimistic case when all bounds hold simultaneously.

Recall the margin gaps penalty proposed in Section 5.1.4. In this alternate penalty definition, we do not penalize the first t margin gaps when aligning the reads. Therefore, if a read b is sequenced within t nucleotides from the beginning of B_a, there is a good chance that the calculated distance will be less or equal to d_a. We will provide an upper bound on the probability of this event. We start with a bound for a single pair of reads.

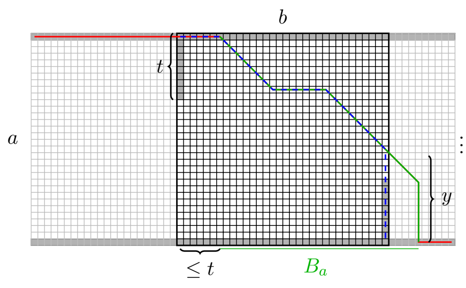
We will also modify $\mathsf{dist}_{\mathsf{rr}}$ in this section to be more similar to the $\mathsf{dist}_{\mathsf{rS}}$ distance. However, there will be only a small difference to the definition we used in Section 5.1.4. There will be no penalty for leading or trailing margin gaps of arbitrary length in the direction representing sequence B, i.e., in the first row or column of the dynamic programming table. On the contrary, the penalty is without any cost only for t symbols in the other direction - in the first and the last column. See Figure 7.4 for further illustration. We need this asymmetric definition so that some of the proofs work (proof of the Lemma 7.6); however, we will stick to the symmetric case in the implementation. The asymmetry would cause the algorithm to run approximately two times slower, which is undesired. This modified distance will be denoted $\mathsf{dist}_{\mathsf{rr}}{}'$ in this section.

Lemma 7.6 *Let $a \in R_A$, and $b \in R_B$ be reads. Suppose that $|B_a| \leq l$ and read b starts within t symbols from B_a starting position. Then $\mathsf{dist}_{rr}{}'$ provides a lower bound on dist_{rS}, i.e.,*

$$\mathsf{dist}_{rS}(a, B) \geq \mathsf{dist}_{rr}{}'(a, b). \tag{7.17}$$



(a) The first part of the proof.



(b) The second part of the proof.

Figure 7.5: An illustration to the proof of Lemma 7.6. The figure illustrates the $\mathsf{dist}_{\mathsf{rS}}$ alignment path. Its cost-free part is colored red, and the green part then illustrates the part which has possibly non-zero cost. The blue dashed path then shows one of the possible alignments that are under consideration of $\mathsf{dist}_{\mathsf{rr}}'(a, b)$ and which has a cost of at most $\mathsf{dist}_{\mathsf{rS}}(a, B)$.

Proof Refer to Figure 7.5. Suppose that read b starts at most t nucleotides before B_a, i.e., before the optimal alignment defined by $\mathsf{dist}_{\mathsf{rS}}(a, B)$. Then, the alignment $\mathsf{dist}_{\mathsf{rr}}'$ leaves read b in one of the two following places:

1. The $\mathsf{dist}_{\mathsf{rS}}$ alignment leaves read b in one of the horizontal or vertical margin gaps with no penalty. This case is illustrated in Figure 7.5a. Then there exists an $\mathsf{dist}_{\mathsf{rr}}$ alignment between a and b that has a cost smaller or equal to $\mathsf{dist}_{\mathsf{rS}}(a, B)$ defined by this path.

2. The $\mathsf{dist}_{\mathsf{rS}}$ alignment leaves read b above the free margin gaps (as in Figure 7.5b). Suppose that this occurs y symbols from the bottom right corner of the rectangle defined by a and b. Then the $\mathsf{dist}_{\mathsf{rS}}$ alignment path to the right from this intersection needs to contain at least $y-t$ vertical gaps (i.e., indels in a) to fulfill condition $|B_a| \leq l$. The $y-t$ vertical gaps bound is because of the fact that the part of the $\mathsf{dist}_{\mathsf{rS}}$ alignment has to contain at most t symbols from B. At the same time, it has to cover exactly y symbols from A. As a result, there exists a path that follows the $\mathsf{dist}_{\mathsf{rS}}$ alignment between a and B, and once it leaves the interval defined by b, the alignment uses those $y-t$ gaps in a to get to the cost-free margin gaps provided by $\mathsf{dist}_{\mathsf{rr}}'$. This path has length at most $\mathsf{dist}_{\mathsf{rS}}(a, B)$ and is considered among valid alignments in $\mathsf{dist}_{\mathsf{rr}}'(a, b)$. Therefore, $\mathsf{dist}_{\mathsf{rr}}'(a, b)$ can be only smaller than $\mathsf{dist}_{\mathsf{rS}}(a, B)$ in this case.

In both cases, Equation (7.17) holds. The situation when b starts at most t nucleotides after B_a is symmetric. ∎

Lemma 7.7 *Let R_B be a read bag of sequence B sequenced with coverage c and read length l. Let a be a read of sequence A with length l. Then,*

$$\mathsf{dist}_{rS}(a, B) \geq \min_{b \in R_B} \mathsf{dist}_{rr}'(a, b) \tag{7.18}$$

with a probability at least

$$1 - \left(1 - \frac{l}{2c(|B| - l + 1)}\right)^{\frac{|B|c}{l}}. \tag{7.19}$$

Proof Due to the assumption of the Jukes-Cantor model, we might claim that the probability of indels in a and B_a is the same. Therefore, with probability at least $\frac{1}{2}$,

$|B_a| \leq a$. Consider single read b. This read can be located in one of $2t+1$ positions so that assumptions of Lemma 7.6 hold. The probability that, for a randomly selected read b, Equation (7.17) holds is, therefore,

$$\frac{1}{2}\frac{2t+1}{|B|-l+1}. \tag{7.20}$$

The nominator represents the count of the favorable positions for read b; the denominator is the number of all reads in R_B. By exploiting the complementary events, we can calculate the probability that at least one of the reads in read bag R_B matches one of the $2t+1$ positions as

$$1-\left(1-\frac{1}{2}\frac{2t+1}{|B|-l+1}\right)^{|R_B|}. \tag{7.21}$$

Plug $t=\frac{1}{2}\left(\frac{l}{c}-1\right)$ (as shown in (5.4)) into (7.21) to obtain

$$1-\left(1-\frac{l}{2c(|B|-l+1)}\right)^{|R_B|}. \tag{7.22}$$

Using the dependence between the number of reads and the sequence length in (2.11), we get Equation (7.19). ■

Example 6 *Assuming a small experiment with* $|B| = 10{,}000$*,* $l = 100$*, and* $c = 3$*, we can calculate* $|R_B| = 300$*. Inequality* (7.18) *then holds with approximately* 40 % *probability.*

We can further simplify Formula (7.19) using the Taylor series approximation. The first-order approximation gives

$$1-\left(1-\frac{1}{2c}\frac{l}{|B|-l+1}\right)^{\frac{|B|c}{l}} \approx 1-1+\frac{1}{2c}\frac{l}{|B|-l+1}\frac{|B|c}{l} \approx \frac{1}{2}. \tag{7.23}$$

The second approximation in (7.23) can be justified by the fact that $|B| \gg l$. The second-order approximation is very close to the result from Example 6. This time we will use the fact that $|R_B| = \frac{|B|c}{l} \gg 1$ to obtain the approximation

$$1-\left(1-\frac{1}{2c}\frac{l}{|B|-l+1}\right)^{\frac{|B|c}{l}} \approx$$
$$\approx 1-1+\frac{1}{2c}\frac{l}{|B|-l+1}\frac{|B|c}{l}-\frac{1}{2}\left(\frac{1}{2c}\frac{l}{|B|-l+1}\right)^2\frac{|B|c}{l}\left(\frac{|B|c}{l}-1\right) \approx \frac{3}{8}. \tag{7.24}$$

Following the order of Section 7.1, we shall now extend Theorem 7.5. Unfortunately, here comes into play the pessimistic part - the lower bound is based on a sum for many reads in R_A. Hence, the bound holds if all of the inequalities in the sum hold. This gives a very low probability. In terms of Example 6, this probability would be as low as 0.4^{300}. However, there is still room for improvement, as we will show in the next section.

Theorem 7.8 *For any sequences* A, B *and bag of reads* R_A, R_B *holds*

$$\operatorname{dist}(A,B) \geq \frac{1}{c_m}\sum_{a\in R_A}\min_{b\in R_B} \mathsf{dist}_{rr}'(a,b) \tag{7.25}$$

with a probability of at least

$$\left[1-\left(1-\frac{1}{2c}\frac{l}{|B|-l+1}\right)^{|R_B|}\right]^{|R_A|}. \tag{7.26}$$

Proof Inequality (7.25) comes from plugging (7.18) into (7.13). The probability in (7.26) is the joint probability that all of the $|R_A|$ inequalities hold simultaneously. ■

7.3 A Tighter Bound - Empirical Bound

As we said, the bound in Theorem 7.8 has a very low probability estimate (7.26), far from being practical. Nevertheless, this bound can be made tighter using the following observation - the bound is based on the inequality that compares $\mathsf{dist_{rr}}'$ to $\mathsf{dist_{rS}}$. However, in fact, we need to compare path length d_a defined by the alignment $\text{dist}(A, B)$, which does not need to be minimal for read a as $\mathsf{dist_{rS}}(a, B)$ is.

The inequality in (7.2) is not necessarily equality and, therefore, if the estimate given by $\min_{b \in R_B} \mathsf{dist_{rr}}'(a, b)$ is greater than $\mathsf{dist_{rS}}(a, R_B)$, there is still room for Inequality (7.18) to hold. Also, if for some reads, Inequality (7.18) does not hold, there might be other read in R_A that can compensate this invalid inequality in the final Equation (7.25). We will try to capture those events theoretically based on an empirically calculated distribution of the error between $\mathsf{dist_{rr}}'(a, b)$ and the corresponding part of alignment cost in $\text{dist}(A, B)$, i.e., d_a.

Figure 7.6 shows the true distribution of the error defined by the difference between $\min_{b \in R_B} \mathsf{dist_{rr}}'(a, b)$ and d_a for several types of sequences. Assuming that the pool of sequences is known in advance, we can empirically estimate the probability distribution of deviation between $\mathsf{dist_{rr}}'(a, b)$ and the partial alignment cost d_a. Denote this distribution $p(\Delta = i)$. Integer parameter i has the meaning of the difference between $\min_{b \in R_B} \mathsf{dist_{rr}}'(a, b)$ and d_a. [245]

Using an approach similar to Chapter 6, we can use the distribution to build a generating polynomial

$$\text{genpoly}_p(x) = \sum_{i=-\infty}^{\infty} p(\Delta = i) x^i.$$

Theorem 7.9 *For any two sequences A and B, let R_A and R_B be their read bags. Let c_m be the maximum per-base coverage. Let p be the probability distribution of difference $\min_{b \in R_B} \mathsf{dist_{rr}}'(a, b) - d_a$. Then,*

$$\frac{1}{c_m} \sum_{a \in R_A} \min_{b \in R_B} \mathsf{dist_{rr}}'(a, b) \le \text{dist}(A, B) \tag{7.27}$$

with probability $\sum_{i=-\infty}^{0} a_i$, where

$$\sum_{i=-\infty}^{\infty} a_i x^i = \left(\text{genpoly}_p(x)\right)^{|R_A|}. \tag{7.28}$$

Proof The proof can be split into proving two inequalities, as in the equation below

$$\frac{1}{c_m} \sum_{a \in R_A} \min_{b \in R_B} \mathsf{dist_{rr}}'(a, b) \le \frac{1}{c_m} \sum_{a \in R_A} d_a \le \text{dist}(A, B). \tag{7.29}$$

The first inequality comes from the fact that we know the distribution p of the error between d_a and its approximation by the sum $\min_{b \in R_B} \mathsf{dist_{rr}}'(a, b)$. Similarly to the proof of Theorem 6.4, we can argue that the probability that the inequality holds is the same as the probability of the sum of errors being less or equal to zero, which can be found in coefficients of (7.28).

We have already proven the second inequality as part of the proof of Theorem 7.5 and holds with a probability equal to 1. ■

To generate Figure 7.6, we used ten pairs of sequences to account for different data settings. Six pairs included real-world sequences; the remaining four were random sequences (generated under the uniform i.i.d. assumption using a tool on `http:`

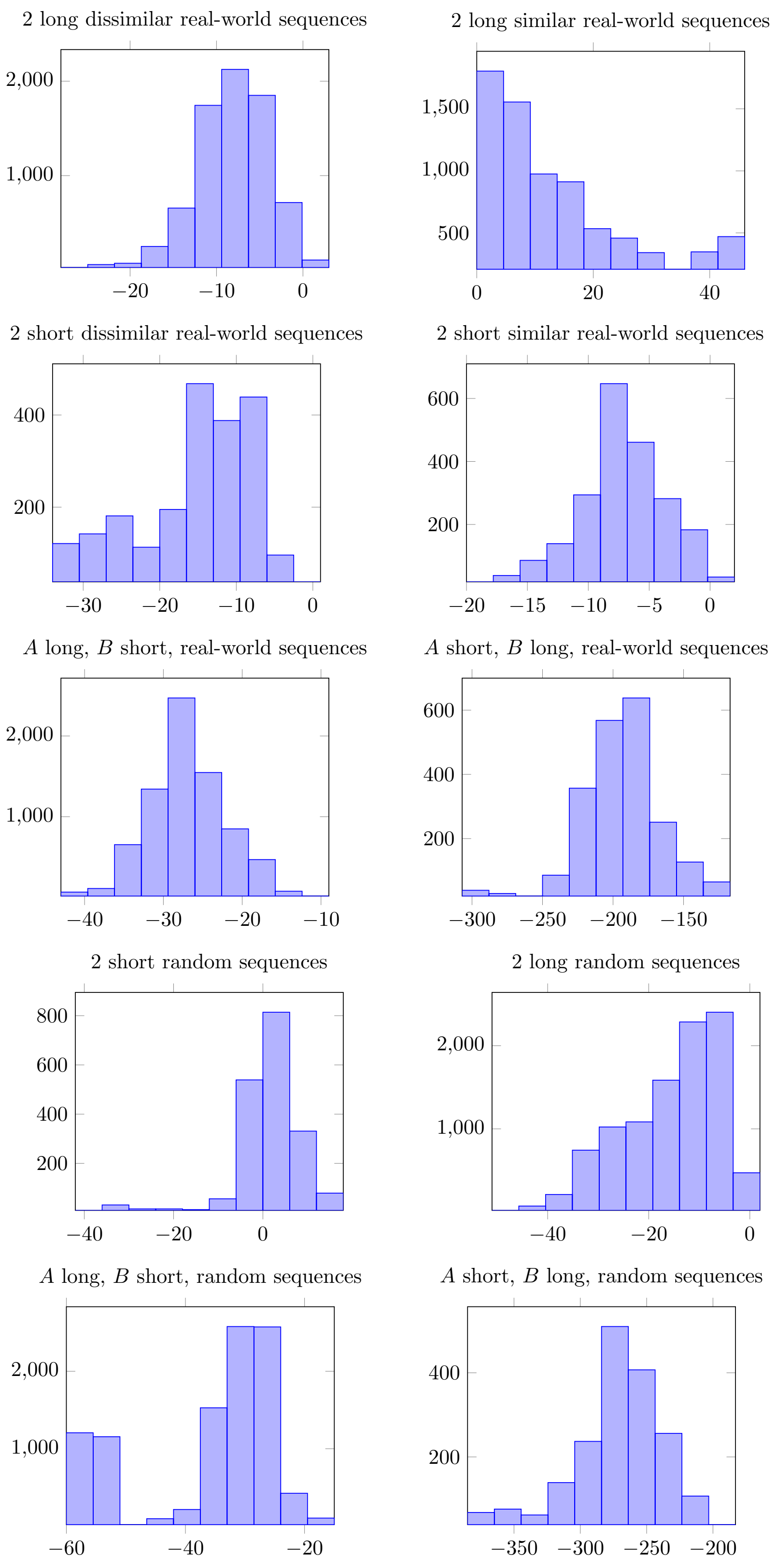


Figure 7.6: A histogram of error approximating the partial alignment cost d_a in $\text{dist}(A, B)$ by $\min_{b \in R_B} \mathsf{dist_{rr}}'(a, b)$ for $c = 3$ and $t = 0$.

Dataset description	A	B	$\|A\|$	$\|B\|$
2 long dissimilar real-world sequences	DQ812094	AB447435	7702	7511
2 long similar real-world sequences	DQ812094	DQ219396	7702	7707
2 short dissimilar real-world sequences	FJ966079	EF015778	2280	2687
2 short similar real-world sequences	FJ966079	FJ966080	2280	2274
A long, B short, real-world sequences	DQ812094	FJ966080	7702	2280
A short, B long, real-world sequences	FJ966079	DQ219396	2280	7707
2 short random sequences	random	random	2000	2000
2 long random sequences	random	random	10000	10000
A long, B short, random sequences	random	random	10000	2000
A short, B long, random sequences	random	random	2000	10000

Table 7.1: Description of the datasets. The code under the real-world datasets refers to the ENA [166] accession, by which it is possible to find the sequence online. [245]

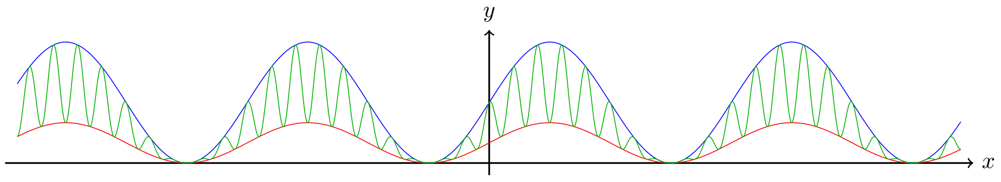


Figure 7.7: An illustration why we might prefer correlated bounds to tighter bounds. The blue function is the approximated one. We see that the green function is a tighter bound than the red one; however, the red one is a better estimate.

`//www.faculty.ucr.edu/~mmaduro/random.htm`). Two real-world pairs included similar virus DNA sequences; in the first case, they were short, and in the second long. For both dissimilar sequences and random sequences, we considered all four possible combinations of short and long sequences A and B. The summary of the used datasets is in Table 7.1. [245]

7.4 A Correlation View

The lower bound in the section above will hold almost for sure. However, a tighter lower bound is not always the desired output. Imagine two bounds; one is equal to one-tenth of the predicted distance $\text{dist}(A, B)$, and the second oscillates randomly between $\text{dist}(A, B)$ and $\frac{1}{10}\,\text{dist}(A, B)$. Clearly, the second bound is tighter than the first one; however, the first one is preferable in terms of correlation. An illustration of this situation may be found in Figure 7.7.

Therefore, instead of the error between $\min_{b \in R_B} \mathsf{dist_{rr}}'(a, b)$ and d_a, we might be interested in the correlation of the latter. To further investigate this question, we measured the dependence of Pearson's correlation coefficient (see Figure 7.8) and Spearman's correlation coefficient (see Figure 7.9) between $\min_{b \in R_B} \mathsf{dist_{rr}}'(a, b)$ and d_a. Those measurements were done on the same datasets as in Section 7.3. From Figures 7.8 and 7.9, we see that on such a small scale, the correlation is in half of the cases close to 0.1 or 0.2. However, the experiments in Chapter 11 show that once we sum the distances, the correlation is often above 0.9.

We can notice that for random datasets, the dependence is more smooth. When at least one of the sequences is long, the correlation does not improve much when $c > 3$.

The best results are, however, reached for two short random sequences. If we compare the first two rows in Figure 7.8, we can notice that the correlation is higher for long sequences in the case of dissimilar sequences. However, for similar sequences, shorter sequences result in a higher correlation than long sequences. The third row also shows that comparing a long sequence to a short one is better than the other way. Fewer reads in R_B mean that there is less chance of hitting a more similar read by chance. The other direction leads to a negative correlation. This holds for both real-world and artificial datasets.

The author of this text proposes a possible explanation of the latter phenomena by the ensemble effect [68] that is well known from algorithms as AdaBoost [97] or random forest [124]. Each of the reads in the bag is a weak predictor of the distance. However, their sum is a good predictor. In the simplest scenario, there is a noise sampled from an unknown, however, fixed distribution. This noise causes large errors in the case of a single measurement (calculation of $\min_{b \in R_B} \mathsf{dist}_{\mathsf{rr}}'(a, b)$). However, as we select more and more measurements (i.e., we calculate the sum $\sum_{a \in R_A}$), the noise either cancels out (in the case of a symmetric noise distribution) or converges to a sum that is only a multiple of the number of reads $|R_A|$. The statistical reason for this behavior is backed by the central limit theorem [92].

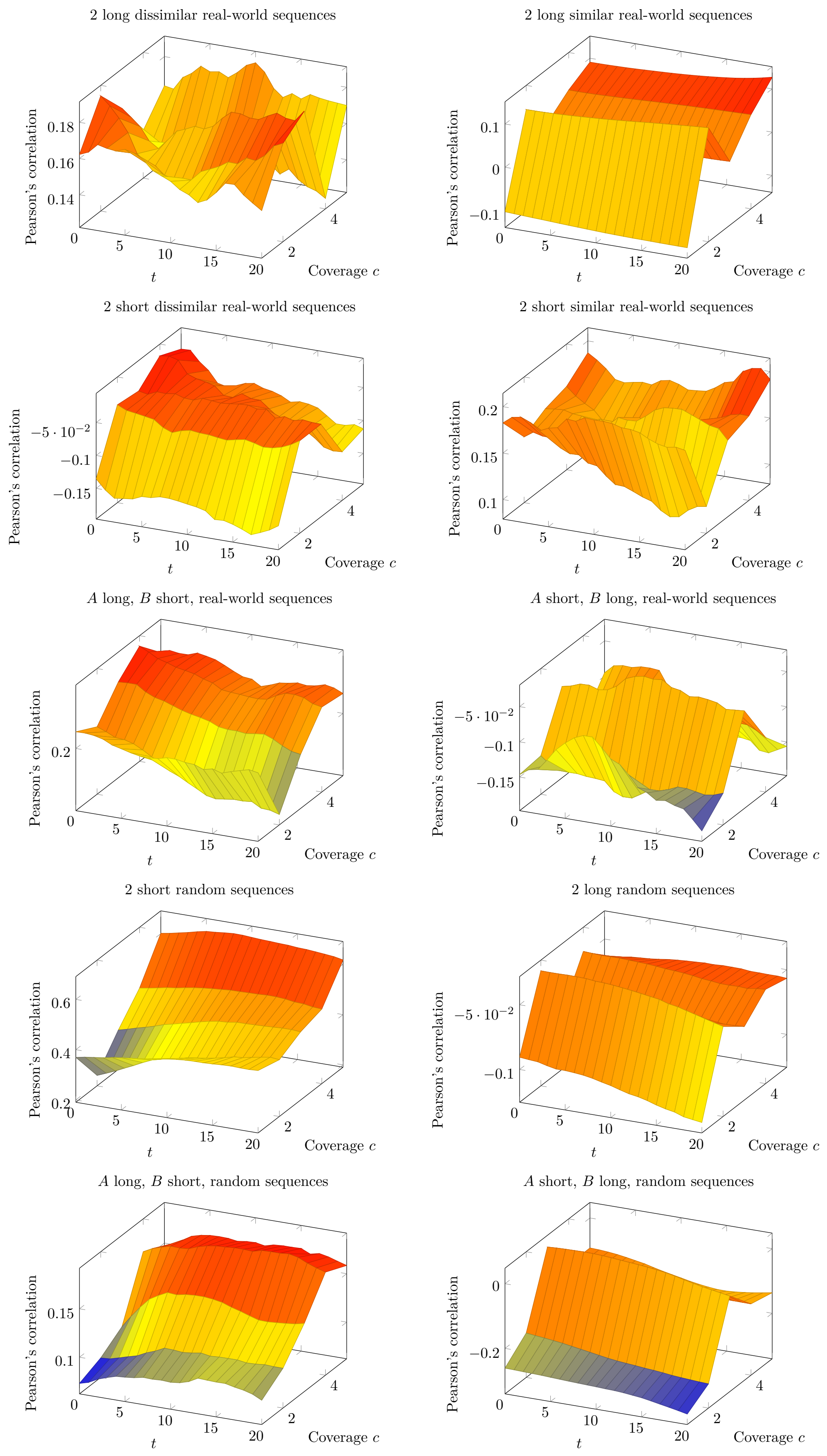


Figure 7.8: Pearson's correlation coefficient between d_a and $\min_{b \in R_B} \mathsf{dist_{rr}}'(a, b)$ for different choices of coverage c and the number of cost-free margin gaps t.

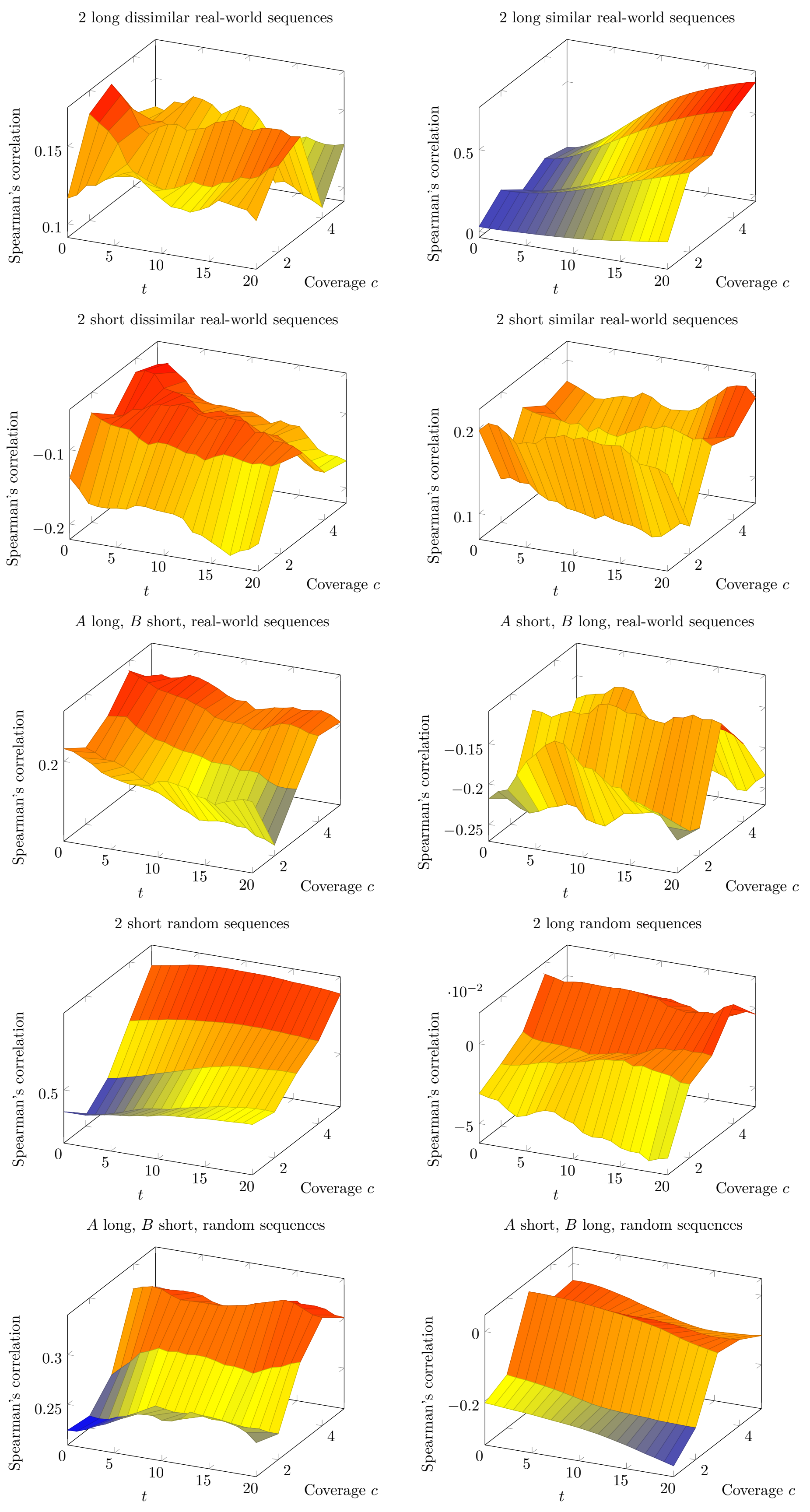


Figure 7.9: Spearman's correlation coefficient between d_a and $\min_{b \in R_B} \mathsf{dist_{rr}}'(a, b)$ for different choices of coverage c and the number of cost-free margin gaps t.

Chapter 8

Estimating Similarity from Contig Sets

In this chapter, we will present an approach that shows how to estimate the sequence similarity from contig sets. We will assume that an assembly algorithm provides us with contigs, i.e., the easy parts of the assembly step. Once we know the contigs, we may exploit more information in the provided distance measure. Similarly to Chapter 5, we will provide a distance function for pairs of contig sets. The contents of this chapter correspond to paper [238].

Our goal is to propose a dissimilarity measure $\mathsf{Dist}_{\mathsf{CC}}(C_A, C_B)$ that approximates $\text{dist}(A, B)$. First, given any pair $(\alpha \in C_A, \beta \in C_B)$, we estimate their most likely overlap in the unknown optimal alignment of A and B, assuming they do overlap. Note that here the term 'overlap' is with respect to the mutual positioning of α and β in the *alignment* of A and B, so the parts of α and β deemed to overlap may, in general, include insertions and mismatches. Usually, the term overlap is used for prefix-suffix overlap between two contigs from the same sequence. In this Chapter, we will also use overlap for similar, high-similarity regions between two contigs from different organisms. Second, the $|C_A||C_B|$ estimates resulting from the latter for all pairs of contigs are filtered using further (heuristic) constraints imposed on M-to-N contig matching; here, we mainly want to filter out those pairs of contigs which likely *do not* overlap in the optimal alignment of A and B. Lastly, the resulting set of hypothesized contig overlaps is used to estimate the distance between A and B. These steps are described in the following three sections, respectively. [238]

8.1 Estimating Overlaps for Contig Pairs [238]

Consider two contigs $\alpha \in C_A$ and $\beta \in C_B$. In the optimal alignment of A and B, there are several options for how α and β can be positioned with respect to each other, assuming they have an overlap - recall Definition 2.20 of the possible overlap set $\mathsf{Overlaps}$. Let $\mathsf{Pref}(\alpha)$ ($\mathsf{Suff}(\alpha)$) be the set of all prefixes (suffixes) longer[1] than 20 of string α, and let $\alpha_{\vdots}$ denote an unspecified substring of α. The first option is that a suffix of one contig overlaps with a prefix of the other contig, which leads to the first two sets in the union in Equation (2.13). The second option is that one contig matches a substring of the other contig, which matches the former two sets in Equation (2.13).

The following function then yields our estimate of the most likely mutual overlap of

[1]The threshold of 20 nucleotides is set to prevent very short random overlaps. This number approximately matches the free gap parameter t (introduced in Section 5.1.4) for most common read length $l = 100$ and coverage between 2 and 3.

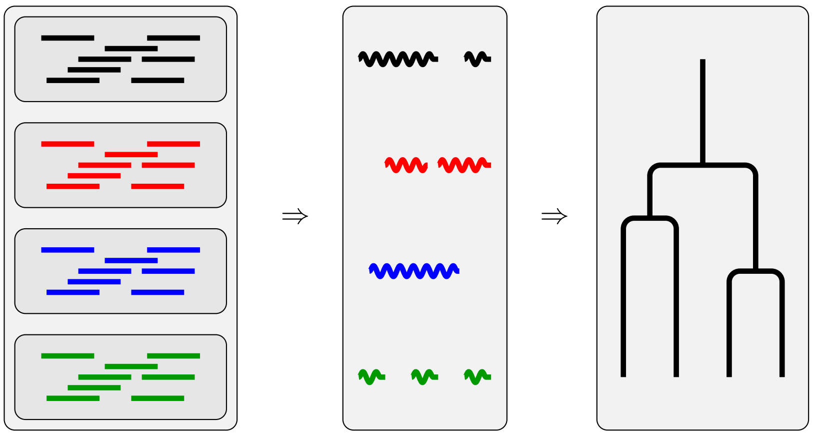


Figure 8.1: A procedure for phylogeny with only contig assembly. We estimate the distance between the clustered objects from contig sets to reconstruct the phylogenetic tree. For this purpose, we estimate the Levenshtein distance between the DNA sequences from pairs of contig sets.

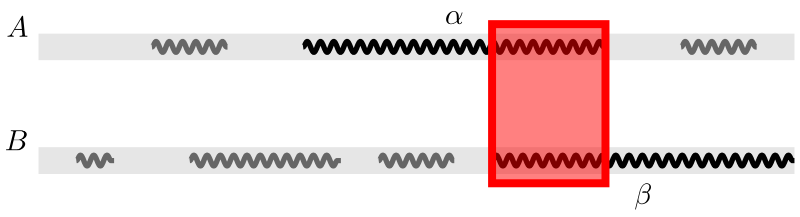


Figure 8.2: The first step of the procedure. For two contigs α and β, we find their likely overlap assuming that the contigs do overlap in the optimal alignment of original sequences A and B.

α and β, assuming that the latter overlap at all.

$$\mathsf{overlap}(\alpha, \beta) = \operatorname*{arg\,min}_{(\alpha_{:}, \beta_{:}) \in \mathsf{Overlaps}(\alpha, \beta)} \overline{\mathrm{dist}}(\alpha_{:}, \beta_{:}), \tag{8.1}$$

where $\overline{\mathrm{dist}}$ is the *post-normalized Levenshtein distance*. See Definition 2.5 for details.

To calculate $\mathsf{overlap}(\alpha, \beta)$, we modify the Smith-Waterman local alignment algorithm [268]. For this purpose, we maintain an array `dist`, storing the Levenshtein distance of suffixes of prefixes of α and β. Two other arrays `lenA` and `lenB` store lengths of matching substrings. When filling each cell, we decide the value based on (14.2). The complete algorithm that we use is described in Algorithm 8.1. It is a heuristic algorithm; we have not been able to design an exact algorithm with the time complexity of $\mathcal{O}(|\alpha||\beta|)$ solving (8.1).[2] An execution of the algorithm is exemplified in Table 8.1.

A technical remark is in order. Prior to executing the above algorithm, we need to pre-process the contig sets for reasons irrelevant to the algorithmic principles described. In particular, because contigs represent DNA, we do not know which strand they come from. Therefore, if we calculate the overlap of α and β, we do not know whether to match α and β or α with the reversed complement of β, i.e., $\overline{\underleftarrow{\beta}}$. To deal with this, we simply expand one (but not both, for obvious reasons) of the two contig sets by the reversed complements of all its elements.

[2]While the problem is polynomial, the time complexity of the brute-force solution incurs a polynomial of order 4, rendering it unusable even on small datasets.

```
function overlap($\alpha$,$\beta$)
    $dist, lenA, lenB \leftarrow$ 2D arrays of zeros of size $(|\alpha|+1) \times (|\beta|+1)$
    for $i \in \{1,2,\dots,|\alpha|\}$ do                                    ▷ for each row
        for $j \in \{1,2,\dots,|\beta|\}$ do                                 ▷ for each column
            choose the option that leads to the lowest $\frac{d}{\max\{lA,lB\}}$:
            gap in $\alpha$: $d \leftarrow dist[i,j-1]+1; lA \leftarrow lenA[i,j-1]; lB \leftarrow lenB[i,j-1]+1$
            gap in $\beta$: $d \leftarrow dist[i-1,j]+1; lA \leftarrow lenA[i-1,j]+1; lB \leftarrow lenB[i-1,j]$
            (mis)match: $d \leftarrow dist[i-1,j-1] + (\alpha[i-1] \neq \beta[j-1])$;
                $lA \leftarrow lenA[i-1,j-1]+1; lB \leftarrow lenB[i-1,j-1]+1$
            $dist[i,j] \leftarrow d, lenA[i,j] \leftarrow lA, lenB[i,j] \leftarrow lB$
        end for
        CHECKOPTIMUM($i, |\beta|$)
    end for
    for $j \in \{1,2,\dots,|\beta|-1\}$ do CHECKOPTIMUM($|\alpha|, j$) end for
end function

function CHECKOPTIMUM($i, j$)
    if $\frac{dist[i,j]}{\max\{lenA[i,j],lenB[i,j]\}}$ is the smallest & $\min\{lenA[i,j], lenB[i,j]\} \geq 20$ then
        the new optimum is located at $(i,j)$, store it
    end if
end function
```

Algorithm 8.1: Pseudocode for the heuristic used for finding overlap(α, β).

`dist` matrix

			A	A	G	C
		0	1	2	3	4
	0	0	0	0	0	0
C	1	0	1	1	1	0
A	2	0	0	1	2	1
T	3	0	1	1	2	2
G	4	0	2	2	**1**	2

`lenA` matrix

			A	A	G	C
		0	1	2	3	4
	0	0	0	0	0	0
C	1	0	1	1	1	1
A	2	0	1	1	1	2
T	3	0	2	2	2	3
G	4	0	3	3	**3**	3

`lenB` matrix

			A	A	G	C
		0	1	2	3	4
	0	0	0	0	0	0
C	1	0	0	0	0	1
A	2	0	1	2	3	1
T	3	0	1	2	3	1
G	4	0	1	2	**3**	4

Table 8.1: An illustration of Algorithm 8.1 showing the three involved data matrices for inputs $\alpha =$ CATG, $\beta =$ AAGC and minimum overlap 2. The entries yielding the minimum value of the criterion are marked in boldface, producing (ATG, AAG) as the detected overlap.

8.1.1 Exact Algorithm

For comparison, we can provide an exact algorithm running in cubic time capable of optimization of (8.1) in an exact manner. This algorithm was not part of [238] but was published in [244]. Due to the high polynomial, it cannot be used in real-world applications; however, the exact algorithm can show that Algorithm 8.1 is a good heuristic for real-world data.

The idea behind the algorithms is that we try to search for all suffixes of the first sequence in the second sequence. We enforce a prefix of the second sequence to overlap with the whole suffix of the first sequence. Then we do the same procedure with the reversed role of the first and the second sequence. This way, we avoid the $\mathcal{O}(n^4)$ and obtain a cubic algorithm. A pseudocode for the algorithm is given in Algorithm 8.2. [244]

```
    function overlap($\alpha, \beta$)
        for $i \in \{0, 1, \dots, |\alpha| - 20\}$ do overlap($\alpha, \beta, i$) end for        ▷ for each suffix of $\alpha$
        for $i \in \{0, 1, \dots, |\beta| - 20\}$ do overlap($\beta, \alpha, i$) end for        ▷ for each suffix of $\beta$
    end function
5:  function overlap($\alpha, \beta, aoffset$)
        $dist \leftarrow$ 2D array of zeros
        for $j \in \{1, 2, \dots, |\beta|\}$ do $dist[aoffset][j] = j$ end for        ▷ initialize the first row
        for $i \in \{aoffset, aoffset + 1, \dots, |\alpha| - 1\}$ do        ▷ fill the table as in Wagner-Fischer
            $dist[i + 1][0] = dist[i][0]$
10:         for $j \in \{0, 1, \dots, |\beta|\}$ do
                $dist[i+1][j+1] = \min \begin{cases} dist[i+1][j] + 1, \\ dist[i][j+1] + 1, \\ dist[i][j] + [\![\alpha_i \neq \beta_j]\!] \end{cases}$
            end for        ▷ (3.1)
            if $i - aoffset + 1 \geq 20$ and $\frac{dist[i+1][|\beta|]}{\max\{i - aoffset + 1, |\beta|\}}$ is the smallest then
                the new optimum is at $i + 1$, $|\beta|$        ▷ check the last column for an optimum
15:         end if
        end for
        for $j \in \{20, 21, \dots, |\beta| - 1\}$ do
            if $\frac{dist[|\alpha|][j]}{\max\{|\alpha| - aOffset, j\}}$ is the smallest then
                the new optimum is at $|\alpha|$, $j$        ▷ check the last row for an optimum
20:         end if
        end for
    end function
```

Algorithm 8.2: An exact algorithm to the heuristic in Algorithm 8.1. [244]

8.2 Estimating Overlaps for Contig Sets [238]

The procedure from the previous section can be used to yield the most likely overlap for each possible pair of contigs $\alpha \in C_A$, $\beta \in C_B$. Of course, not all such $|C_A||C_B|$ pairs actually overlap in the unknown optimal alignment of A and B. To filter the overlap candidates towards a smaller, more plausible set, we adhere to the following rules.

1. For given contig $\alpha \in C_A$, we should only pick elements from set $\{\mathsf{overlap}(\alpha, \beta) \mid \beta \in C_B\}$ which do not intersect between themselves,

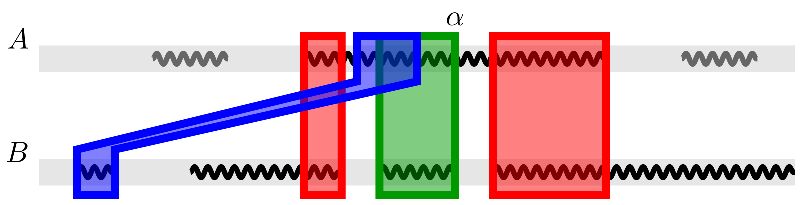


Figure 8.3: The second step of the procedure. After all possible overlaps with contig α are identified, only some of them are selected. Out of the blue and the green overlap, only one should be selected.

2. the resulting overlap pairs should minimize the sum of the $\overline{\text{dist}}$ values.

Note that the two selection rules are only a heuristic. Not even rule 1 is dictated strictly by biological principles. Indeed, it may, in fact, happen that two contigs $\beta, \beta' \in C_B$ map to the same contig (or its substring) $\alpha \in C_A$ in a way making β and β' overlap.

The application of the two selection rules reduces to the *weighted interval scheduling* problem defined in [153]. In this problem, we have n tasks, each of a value v_t (for $t = 1, 2, \ldots, n$), and a starting and finishing time. Our goal is to select a subset of non-intersecting tasks that maximizes the sum of the selected task values.

Weighted interval scheduling can be solved in $\mathcal{O}(n \log n)$ time by a simple dynamic programming algorithm. We pass the tasks ordered by the finishing time, and for each task, we have two options — to include it or not. If S_t is an optimal solution for all tasks up to a task t (in the sorted order), then S_{t+1} is the maximum of S_t and $v_{t+1} + S_{t'}$, where t' is the task with the largest finishing time, which is smaller than the starting time of $t + 1$.

In our case, the starting and finishing times represent the location of $\alpha_{\bullet}$ in α. The value we assign to a pair $(\alpha_{\bullet}, \beta_{\bullet})$ is given by

$$v_{(\alpha_{\bullet},\beta_{\bullet})} = \frac{1}{\overline{\text{dist}}(\alpha_{\bullet}, \beta_{\bullet})} = \frac{\max\{|\alpha_{\bullet}|, |\beta_{\bullet}|\}}{\text{dist}(\alpha_{\bullet}, \beta_{\bullet})}. \tag{8.2}$$

Note that this value can be computed since Algorithm 8.1 has maintained for each potential overlap $(\alpha_{\bullet}, \beta_{\bullet})$ the values of $\text{dist}(\alpha_{\bullet}, \beta_{\bullet})$, $|\alpha_{\bullet}|$, $|\beta_{\bullet}|$.

To sum up, the procedure described accepts $\alpha \in C_A$ and C_B, and produces a set we denote $\mathsf{overlap}(\alpha, C_B)$ which is selected from the initial overlap candidates, i.e.

$$\mathsf{overlap}(\alpha, C_B) \subseteq \{\, \mathsf{overlap}(\alpha, \beta) \mid \beta \in C_B \,\}\,. \tag{8.3}$$

Further, overloading the $\mathsf{overlap}$ functor for contig sets, we denote

$$\mathsf{overlap}(C_A, C_B) = \bigcup_{\alpha \in C_A} \mathsf{overlap}(\alpha, C_B).$$

Note that the above function is not symmetric, and this fact will be dealt with in the next section.

8.3 Combining the Results [238]

Having the filtered set of suspected overlaps, we first define the pre-distance $d(C_A, C_B)$ of contig sets C_A, C_B as the sum of the distances associated with the individual overlaps. This, however, does not reflect how much of the alignment of A and B is covered by the overlaps. Therefore, this sum is scaled

$$d(C_A, C_B) = \frac{\sum_{(\alpha_{\bullet},\beta_{\bullet}) \in \mathsf{overlap}(C_A,C_B)} \text{dist}(\alpha_{\bullet}, \beta_{\bullet})}{\sum_{(\alpha_{\bullet},\beta_{\bullet}) \in \mathsf{overlap}(C_A,C_B)} \max\{|\alpha_{\bullet}|, |\beta_{\bullet}|\}} \cdot \frac{l \max\{|R_A|, |R_B|\}}{c}. \tag{8.4}$$

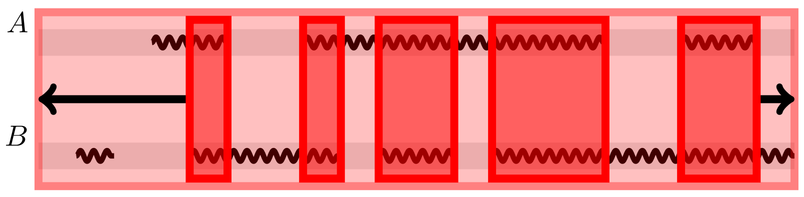


Figure 8.4: The last step of the procedure. Once the overlaps are filtered, we sum the distances and scale based on the fraction of nucleotides covered by the overlaps.

The scaling is done by dividing by the maximum distance that all matching substrings can have (the sum in the denominator) and multiplying by the maximum distance that A and B can have. For the latter, it estimates sequence lengths $|A|, |B|$ from (2.11).

Since $\mathsf{overlap}(C_A, C_B)$ is not symmetric, neither is $d(C_A, C_B)$, the final measure $\mathsf{Dist}_{\mathsf{CC}}(C_A, C_B)$ averages $d(C_A, C_B)$ and $d(C_B, C_A)$

$$\mathsf{Dist}_{\mathsf{CC}}(C_A, C_B) = \frac{d(C_A, C_B) + d(C_B, C_A)}{2}. \tag{8.5}$$

Chapter 9 Combination of the Measures

In this chapter, we will exploit the results from the previous sections. Namely, we will combine the read-to-read distance presented in Chapter 5 together with the method from Chapter 8. The matching strategy differs for the four possible pairing sorts (refer to Figure 9.1): read-to-read, read-to-contig, contig to one or more reads, and contig to one or more contigs. This chapter details them individually and also describes how the matching results integrate into the final distance estimate. Figure 9.2 reveals the data flow among the methodological components.

9.1 Contig-Contig Mapping [240]

The mapping between contigs and their distance was already covered in Chapter 8 as a single measure. Therefore, we will not describe the complete reasoning in this section. For later use, we will start with the asymmetric pre-distance defined in (8.4). In the contig pre-distance, denoted $\mathsf{Dist}'_{\mathsf{CC}}$, we will leave out the scaling factors, i.e., the resulting measure will be from the $[0, 1]$ interval. We will address the distance range in a later step (9.10) involving scaling. To sum it up, the contig-contig part measure for purposes of this chapter will be

$$\mathsf{Dist}'_{\mathsf{CC}}(C_A, C_B) = \frac{\sum_{(c,d)\in\mathsf{overlap}(C_A,C_B)} \mathrm{dist}(c,d)}{\sum_{(c,d)\in\mathsf{overlap}(C_A,C_B)} \max\{|c|,|d|\}}. \tag{9.1}$$

9.2 Avoiding Redundancy [240]

Before we start with mapping reads to contigs, we need to filter out duplicate information. There are two reasons for that — runtime and the aim to avoid bias caused by

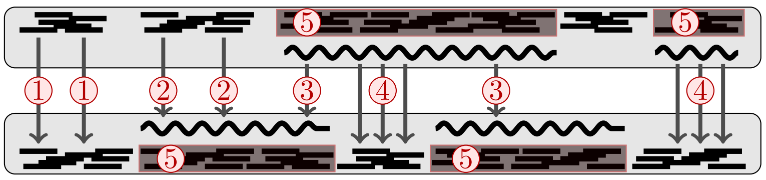


Figure 9.1: An overview of possible matching between reads and contigs. A read can map either to a read (①, [237]), or a contig (②, Section 9.4). Similarly, a contig can map to a part of another contig (③, [238]) or multiple reads (④, Section 9.5). Reads that were assembled to a contig do not need to be considered (⑤, Section 9.2).

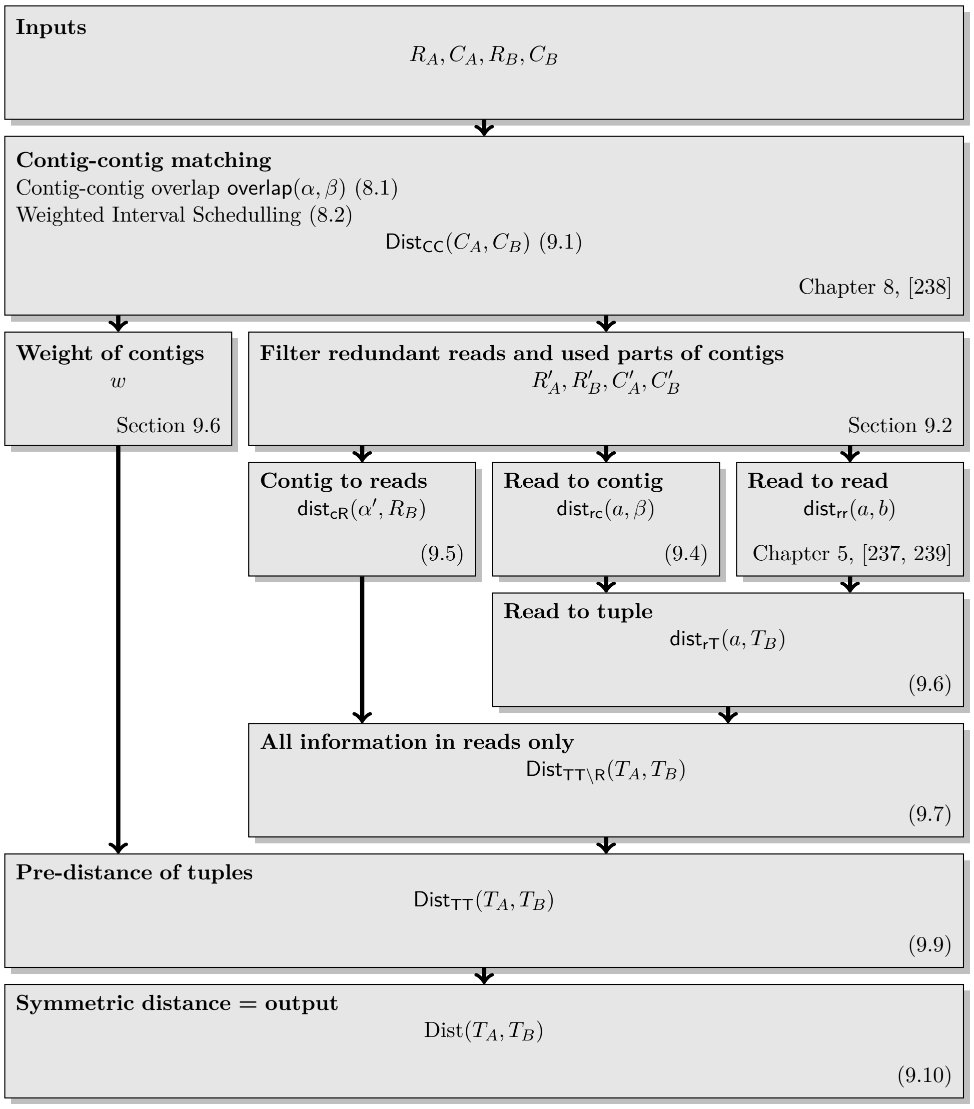

Inputs
R_A, C_A, R_B, C_B
Contig-contig matching
Contig-contig overlap $\mathsf{overlap}(\alpha, \beta)$ (8.1)
Weighted Interval Schedulling (8.2)
$\mathsf{Dist}_{\mathsf{CC}}(C_A, C_B)$ (9.1)
Chapter 8, [238]
Weight of contigs
w
Section 9.6
Filter redundant reads and used parts of contigs
R'_A, R'_B, C'_A, C'_B
Section 9.2
Contig to reads
$\mathsf{dist}_{\mathsf{cR}}(\alpha', R_B)$
(9.5)
Read to contig
$\mathsf{dist}_{\mathsf{rc}}(a, \beta)$
(9.4)
Read to read
$\mathsf{dist}_{\mathsf{rr}}(a, b)$
Chapter 5, [237, 239]
Read to tuple
$\mathsf{dist}_{\mathsf{rT}}(a, T_B)$
(9.6)
All information in reads only
$\mathsf{Dist}_{\mathsf{TT\setminus R}}(T_A, T_B)$
(9.7)
Pre-distance of tuples
$\mathsf{Dist}_{\mathsf{TT}}(T_A, T_B)$
(9.9)
Symmetric distance = output
$\mathrm{Dist}(T_A, T_B)$
(9.10)

Figure 9.2: An overview of the algorithm with references to the corresponding sections or equations.

reusing some parts of sequences twice. Firstly, contigs are generated from reads. There is no need to find matches for a read assembled into a contig since we already found a match for the contig.

To avoid this duplicate work, we re-define

$$R'_A \leftarrow R_A \setminus \{i \in R_A \mid i \text{ is a substring of some } \alpha \in C_A\}. \tag{9.2}$$

To eliminate the reads that are substrings of a contig, we use the Aho-Corasick algorithm [5], which builds an automaton on reads in R_A and finds their occurrences in a single linear pass through a contig.

Secondly, if a part of contig $\alpha \in C_A$ is matched with another contig in C_B, we do not need to match it with reads in R_B — we know that the counterparts are already in C_B. Instead of C_A, we work further with C'_A that contains for all contigs α only substrings of α that are not in $\mathsf{overlap}(\alpha, C_B)$. In other words, let $\mathsf{overlap}(\alpha, C_B) = \{(\alpha_{\bullet 1}, \beta_1), (\alpha_{\bullet 2}, \beta_2), \ldots, (\alpha_{\bullet n}, \beta_n)\}$, let $\alpha = \alpha'_0\alpha_{\bullet 1}\alpha'_1\alpha_{\bullet 2}\alpha'_2 \cdots \alpha_{\bullet n}\alpha'_n$, and let

$$\overline{\mathsf{overlap}}(\alpha, C_B) = \{\alpha'_0, \alpha'_1, \alpha'_2 \ldots \alpha'_n\}$$

be a set of all substrings of α that are not matched to any contig in C_B. Then,

$$C'_A = \bigcup_{\alpha \in C_A} \overline{\mathsf{overlap}}(\alpha, C_B). \tag{9.3}$$

9.3 Read-Read Mapping

[240]

We adopt the method from [237] (see Chapter 5) based on the Monge-Elkan distance [200] to establish the distance function for two read bags. It follows the same spirit as above; in particular, each read a in R_A is matched with the closest read in R_B. Recall the Monge-Elkan distance from Equation (5.1). As the innermost distance function, we use read-read distance $\mathsf{dist}_{\mathsf{rr}}$ from Section 5.1.4 for selecting the closest read in the other read bag. Distance $\mathsf{dist}_{\mathsf{rr}}$ is essentially the Levenshtein distance except for the following adjustment. Because read locations are random, the first $t = \frac{1}{2}\left(\frac{l}{c} - 1\right)$ leading or trailing gaps are not penalized in the alignment of two reads. See [237] or Section 5.1.4 for the derivation of the value assigned to t.

9.4 Read-Contig Mapping

[240]

When aligning a read to a contig, the read can either overlap with one of the contig ends, or it can match a substring of the contig. Therefore, the match for the read can be defined as a substring of β that has the lowest distance from a but for borders where the first t margin gaps are not penalized. Otherwise, the read-contig distance is defined as

$$\mathsf{dist}_{\mathsf{rc}}(a, \beta) = \min_{\substack{i \in [2, |\beta| - l - 2], \\ j \in [i+1, |\beta| - l - 1]}} \mathrm{dist}(a, \beta_i^j), \tag{9.4}$$

where β_i^j denotes a substring of β that starts at i and ends at j. By definition, the $\mathsf{dist}_{\mathsf{rc}}$ distance is very similar to $\mathsf{dist}_{\mathsf{rS}}$ used in Chapter 7.

Following the reasoning from Section 9.3, it is not desirable to penalize leading or trailing gaps up to a certain length at contig ends. Therefore, we modify the Wagner-Fischer dynamic programming algorithm [296] so that it does not penalize the first t leading or trailing gaps caused by a random location of the read or different lengths of the contig and the read. The cost function used for the margin gap penalty is illustrated in Figure 9.3. By using this modified dissimilarity measure as distance $\mathsf{dist}_{\mathsf{rr}}$ in the definition of $\mathsf{dist}_{\mathsf{rc}}(a, \beta)$, we can calculate the distance between a read and a contig.

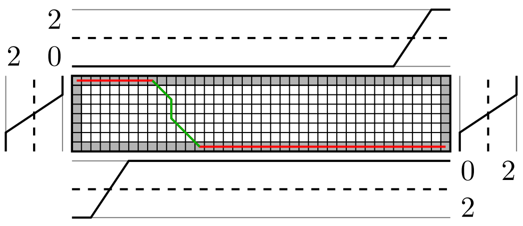


Figure 9.3: An illustration to Formula (9.4). Instead of constant 1 (dashed line), the gap extension penalty on margins changes to the solid line. In this case, the cost-free margin gaps are $t = 2$.

9.5 Contig-Reads Mapping [240]

Here, the mapping is very similar to the one used in the previous section. There is, however, a slight difference - contig α is long, and as a result, there should be multiple reads $b \in R_B$ that map to α. By (2.11), there should be $|\alpha| \cdot c/l$ contigs on average. We, therefore, calculate the distance from α to all $b \in R_B$ and select the $|\alpha| \cdot c/l$ minimum distances.

From Section 9.2, we know that we will use only unmatched substrings α'. Therefore, we define dissimilarity for α' as

$$\underset{\mathsf{cR}}{\mathsf{Dist}}(\alpha', R_B) = \min_{\left\{S \subseteq R'_B \middle| |S| = \left\lfloor \frac{|\alpha'| \cdot c}{l} \right\rfloor \right\}} \sum_{b \in S} \mathsf{dist}_{\mathsf{rc}}(b, \alpha'). \tag{9.5}$$

The formula above selects subset S of read bag R'_B that minimizes the sum of the distances between the reads and α'.

9.6 Combination of the Measures [240]

The final measure based on reads follows Equation (5.1). For each read $a' \in R'_A$, there should be exactly one match — either a read or a contig. We define the distance from a read to the closest read or contig as

$$\mathsf{dist}_{\mathsf{rT}}(a', T_B) = \min \left\{ \min_{b' \in R'_B} \mathsf{dist}_{\mathsf{rr}}(a', b'), \min_{\beta \in C_B} \mathsf{dist}_{\mathsf{rc}}(a', \beta) \right\}. \tag{9.6}$$

The sum of (9.6) and (9.5) over all reads and contigs gives a pre-measure that contains all information captured in reads as

$$\mathsf{Dist}_{\mathsf{TT}\setminus\mathsf{R}}(T_A, T_B) = \frac{\sum_{a' \in R'_A} \mathsf{dist}_{\mathsf{rT}}(a', T_B) + \sum_{\alpha' \in C'_A} \mathsf{dist}_{\mathsf{cR}}(\alpha', R_B)}{l \left(|R'_A| + \sum_{\alpha' \in C'_A} \left\lfloor \frac{|\alpha'| \cdot c}{l} \right\rfloor \right)}. \tag{9.7}$$

The denominator in Formula (9.7) scales to $[0, 1]$ interval and is calculated by substituting l for distance calculations.

In the next step, we will have to combine $\mathsf{Dist}_{\mathsf{TT}\setminus\mathsf{R}}(T_A, T_B)$ with the measure capturing the distance between the contigs. To do so, we use a weighted average based on the part of the original sequences covered by the contigs. $\mathsf{Dist}_{\mathsf{CC}}(C_A, C_B)$ uses only a part of sequence A, namely

$$|\mathsf{overlap}|\,(C_A, C_B) = \sum_{(\alpha_{\bullet}, \cdot) \in \mathsf{overlap}(C_A, C_B)} |\alpha_{\bullet}|.$$

On the contrary, the estimate of A length is $|R_A| \cdot l/c$. The weight assigned to the contig-contig measure is, therefore,

$$w = \min\left\{1, |\mathsf{overlap}|\,(C_A, C_B) \cdot \frac{c}{|R_A| l}\right\}. \tag{9.8}$$

The min operation is to prevent errors of assembly because there is no guarantee that the contigs will be shorter than the true sequence. We define pre-distance as

$$\mathsf{Dist}_{\mathsf{TT}}(T_A, T_B) = w\,\mathsf{Dist}_{\mathsf{CC}}(C_A, C_B) + (1-w)\,\mathsf{Dist}_{\mathsf{TT}\setminus\mathsf{R}}(T_A, T_B). \tag{9.9}$$

The final distance is the scaled (see Formula (2.11) — $\mathsf{Dist}_{\mathsf{TT}}$ is from the $[0, 1]$ interval while $\mathrm{dist}(A, B)$ is from the $[0, \max\{|A|, |B|\}]$ interval) symmetric version

$$\mathsf{Dist}(T_A, T_B) = \frac{\mathsf{Dist}_{\mathsf{TT}}(T_A, T_B) + \mathsf{Dist}_{\mathsf{TT}}(T_B, T_A)}{2} \cdot \frac{l \cdot \max\{|R_A|, |R_B|\}}{c}. \tag{9.10}$$

Chapter 10 Implementation and Optimizations

In this chapter, we will uncover some implementation details of the presented methods. Their goal will be mainly to improve runtime. From this perspective, the most effective ones will include sampling (as the presented methods require a smaller coverage than the conventional assembly) and embeddings that allow us to avoid evaluating all pairwise distance calculations.

10.1 Sampling [239]

Unlike the assembly, the approach in Chapter 5 does not require a high coverage to produce good-quality results. Therefore, for data sequenced with high coverage, we can randomly sample only a small amount of reads to improve runtime. This observation motivates us to conduct a pre-processing step in which the high-coverage input data are replaced by a relatively small random sample thereof in order to gain efficiency. How radically one can afford to downsize the input data set while maintaining the estimation quality can only be determined empirically. We will address this question as part of the experimental evaluation in Chapter 11. A lower-case subscript α will denote the distance measures equipped with the sampling technique.

10.2 Embedding [239]

Given the heuristic nature of the read-read matching method in Chapter 5, the minimum searched in (5.1) need not be exact; an approximate value is acceptable if that allows a significant efficiency gain. We achieve such an approximation through a *read-embedding* technique based on the following basic idea. The high-dimensional space of reads equipped with the Levenshtein distance is mapped to a lower-dimensional space with another distance function, which is easier to compute. The two representations correspond mutually in that elements close (distant) in the original space are also close (distant) in the target space. In the latter, we first find all the elements minimizing the distance to the image of read a in (5.1). For this small set of candidates, we finally compute their true Levenshtein distance to a in the original space to determine the closest one.

The particular embedding we use is based on the q-gram profile and the q-gram distance presented in Section 2.2.2. Theorem 2.11 justifies the usage of the q-gram distance as a fast approximation of the Levenshtein distance.

We instantiate q to $q = 3$ as this value has been known to provide a good balance between the runtime and quality of estimates in the context of the BLAST [9] algorithm. Work [207] recommends using $q = \log_{|\Sigma|} l$. The dimension of the embedding space is

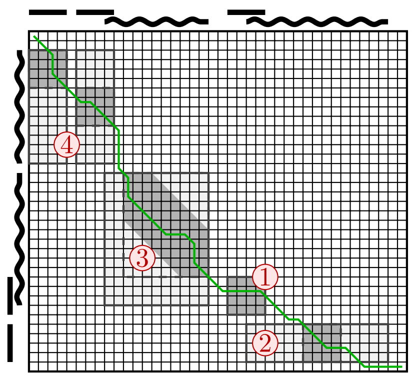


Figure 10.1: The main idea of the efficiency improvements in Section 10.2. Instead of pairwise alignment of all reads and contigs vs. all reads and contigs, we run the quadratic alignment only for a small subset of those. See Figure 9.1 for the meaning of the caption numbers.

thus $|\Sigma|^q = 64$. While we need $\mathcal{O}(|A||B|)$ time to calculate the Levenshtein distance, the q gram distance is much faster to compute: we need $\mathcal{O}(q + |A| + |B| + 4^q)$ or only $\mathcal{O}(4^q)$ if we precompute the q-gram profiles in advance.

The distance measures implementing the embedding technique will be denoted by a lower subscript of q. For example, the distance measure stemming from $\mathsf{Dist}_{\mathsf{MESSG}}$ while adopting the embedding technique will be referred to as $\mathsf{Dist}_{\mathsf{MESSGq}}$.

In the following subsections, we will extend the embedding idea to the more complex distance measures described in Chapters 8 and 9. Equation (9.10) gives a way to compute the distance. However, this method is too slow for practical use as it requires alignment of all data in T_A and T_B. Similarly to the reads, most of those alignments are, however, not necessary as they do not count towards the minimum in (8.3), (9.6), and (9.5). Therefore, we will use only a carefully selected subset of alignments, as illustrated in Figure 10.1. [240]

10.2.1 Efficient Read-Contig Matching

Matching a read to a contig means finding the contig's substring, which is the most similar to the read. Instead of the quadratic dynamic programming approach, we can exploit the q-gram distance and find a set of candidates in linear time. Then for short substrings in the set, we can call the exact quadratic alignment.

A naive search implementation under the q-gram distance requires a quadratic amount of work. This can be reduced to linear time using the *sliding window* method. Once we have calculated the q-gram distance for a prefix of length l of the contig, we can only update the distance by observing the first-to-add q-gram of the contig and last-to-remove q-gram of the contig. Once we have the correct candidates, we trigger the exact quadratic alignment.

10.2.2 Efficient Contig-Contig Matching

We relax the matching problem by positing the following assumption. Referring to (8.1), we assume that for $(\alpha_{\bullet}, \beta_{\bullet}) = \mathsf{overlap}(\alpha, \beta)$, it holds that $|\alpha_{\bullet}| = |\beta_{\bullet}|$. This reduces the quadratic number of possible overlap candidates to a linear number.

We adapt the sliding window approach from the previous section that updates the current distance only by two q-grams as contigs slide one against each other. This allows us to find an overlap candidate that minimizes the ratio of the q-gram distance over the length of the overlap. Once having this candidate, we can calculate the exact Levenshtein distance of the overlapping contigs parts faster using the Ukkonen's cutoff heuristic from [288] and [22].

We now address the choice of q. In the two preceding sections, we compared sequences of fixed length l. Here, the overlap length grows up to the minimum of the contig lengths. According to study [207], the q-gram distance works well for sequences of length 4^q. With longer sequences, the q-gram profiles start to get closer to a fixed distribution (uniform for random sequences). To avoid this bias towards the long overlaps, we need to switch between q values as overlaps get longer. For a value of q, we consider overlaps of length from interval $[4^{q-1}, 4^{q+1}]$. However, direct comparison is still impossible, as our goal is to minimize the Levenshtein distance. Instead of comparing the ratio of dist_q over the overlap length, we divide this ratio by q, which is a direct result of Proposition (2.11).

10.3 Tries

[243]

In paper [243], we proposed an approach to evaluate Formula (5.1) exactly but faster than the naïve approach. The method was based on traversing two tries concurrently, similarly as in the case of similarity joins published in paper [258]. This approach is effective for extremely short reads; however, ineffective for typical read lengths.

We propose to use two tries to iterate over the read bags to evaluate Formula (5.2). A triple of arrays represents each trie. Each entry in those arrays corresponds to a node of a trie. The index of a node is determined based on the assumption that any node has a lower index than all its children. Either the *preorder* traversal can be used, or as in our case, the *BFS* traversal is appropriate.

In the `parent` array, we store a pointer to the parent node. The `character` array contains information about which character labels traversal from the parent to the node. The third array, named `count`, is used only for leaf nodes and stores information about the count of the read in the bag. Recall that all reads are supposed to have the same length l. Therefore, the `count` array only contains useful information for the trie's last layer.

With this representation, we can formulate the basic algorithm. We use the same approach as the standard Wagner-Fischer algorithm [296] for calculating the Levenshtein distance [167]. The only difference is that the algorithm looks at the parent node in the trie, and instead of comparing characters in strings, it compares labels in the trie.

When the whole table is filled, the algorithm needs to look at the entries that correspond to the Cartesian product of leaf nodes of the tries. For each row and column, we need to find a minimum corresponding to $\min_{b \in R_B}$ in (5.1). Then we multiply by the count of the corresponding read, and finally, we average over both directions.

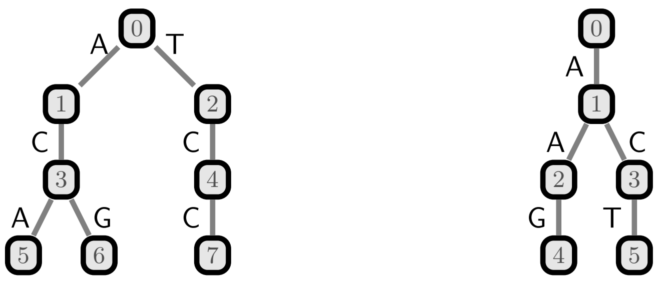
The complete pseudocode of the method is presented in Algorithm 10.1. We will further show how the approach works with an example. Consider two read bags {ACA, ACG, TCC, TCC} and {AAG, ACT}. Their graphical representation as tries, as well as internal representation in the form of arrays, is depicted in Figure 10.2.

The tries are used to fill the dynamic programming table. In the case of tries in Figure 10.2, we obtain Table 10.1. Using the values from Table 10.1, we quantify the distance as

$$\frac{1}{2}\left(\frac{1\cdot 1 + 1\cdot 1 + 2\cdot 2}{4} + \frac{1\cdot 1 + 1\cdot 1}{2}\right) = \frac{5}{4}.$$

```
function SymMonge-ElkanDistance($R_A$, $R_B$)
    $T_A \leftarrow$ GetTrie($R_A$), $T_B \leftarrow$ GetTrie($R_B$)
    $arr \leftarrow$ empty 2D array of size $|T_A| \times |T_B|$
    $arr[0][0] \leftarrow 0$
    for $i \in \{1, 2, \ldots, |T_A|\}$ do                      ▷ initialize the first column
        $arr[i][0] \leftarrow arr[T_A.par(i)][0] + 1$
    end for
    for $j \in \{1, 2, \ldots, |T_B|\}$ do                      ▷ initialize the first row
        $arr[0][j] \leftarrow arr[0][T_B.par(j)] + 1$
    end for
    for $i \in \{1, 2, \ldots, |T_A|\}$ do                      ▷ for each row
        for $j \in \{1, 2, \ldots, |T_B|\}$ do                  ▷ for each column
            $mis \leftarrow T_A.char(i) \neq T_B.char(j)$
            $arr[i][j] \leftarrow \min \begin{cases} arr[T_A.par(i)][j] + 1, \\ arr[i][T_B.par(j)] + 1, \\ arr[T_A.par(i)][T_B.par(j)] + mis \end{cases}$
        end for
    end for
    $s_A \leftarrow 0, s_B \leftarrow 0$                        ▷ evaluate (5.1)
    for $i \in T_A.terminalStates$ do
        $s_A \mathrel{+}= \min_{j \in T_B.terminalSt} \{arr[i][j]\} \cdot T_A.count[i]$
    end for
    for $j \in T_B.terminalStates$ do
        $s_B \mathrel{+}= \min_{i \in T_A.terminalSt} \{arr[i][j]\} \cdot T_B.count[j]$
    end for
    return $\frac{1}{2}\left(\frac{s_A}{|R_A|} + \frac{s_B}{|R_B|}\right)$   ▷ evaluate (5.2)
end function
```

Algorithm 10.1: Pseudocode for the trie approach to evaluate the Monge-Elkan distance. [243]



Node	0	1	2	3	4	5	6	7
parent	∅	0	0	1	2	3	3	4
char	∅	A	T	C	C	A	G	C
count	0	0	0	0	0	1	1	2

Node	0	1	2	3	4	5
parent	∅	0	1	1	2	3
char	∅	A	A	C	G	T
count	0	0	0	0	1	1

Figure 10.2: A representation of read bags {ACA, ACG, TCC, TCC} and {AAG, ACT} as trie structures.

	0	1	2	3	4	5
0	0	1	2	2	3	3
1	1	0	1	1	2	2
2	1	1	2	2	3	2
3	2	1	1	0	2	1
4	2	2	2	1	3	2
5	3	2	1	1	**2**	**1**
6	3	2	2	1	**1**	**1**
7	3	3	3	2	**3**	**2**

Table 10.1: A table that shows the dynamic programming array for the example from Figure 10.2.

Chapter 11 Experimental Evaluation

This chapter contains a summary of experiments presented in papers [237],[239], [238], and [240]. As this is the main experimental section, we will show the results of the methods presented in Chapters 5, 8, and 9. The results show some additional results that were left out in the aforementioned publications for space reasons. However, the results do not show all possible methodological combinations of all measured characteristics and all possible algorithms. Besides that, some minor experiments will be seen in the next chapter, especially those taken from [243], [244], and [245].

The purpose of the experiments is to compare different methods for estimating the Levenshtein distance $\mathrm{dist}(A, B)$ for various real DNA sequences A, B from their read bags R_A, R_B, contig sets C_A, C_B, or the combination of the latter T_A, T_B. The methods include

- the *reference* distance $\mathrm{dist}(A, B)$ (ground truth);
- the most developed version of the approach presented in Chapter 5 (MESSG, MESSGM) applied directly on R_A, R_B;
- modifications of the latter implementing the sampling or embedding optimizations from Chapter 10, denoted by a lower index α and/or q;
- the conventional method based on assembly estimates $\tilde{A}, \tilde{B}$ of the original sequences A, B using five common de-novo gene assemblers (ABySS [266], Edena [122], SSAKE [301], SPAdes [210] and Velvet [316]) and then estimating $\mathrm{dist}(A, B)$ as $\mathrm{dist}(\tilde{A}, \tilde{B})$. Whenever a result of an assembly procedure consisted of multiple contigs, we selected the two longest contigs to represent the distance;
- method applicable on contigs C_A, C_B (in combination with the five assembly algorithms), denoted $\mathsf{Dist}_{\mathsf{CC}}$, as presented in Chapter 8;
- method that combines reads and contigs, denoted by Dist (in combination with the five assembly algorithms), as presented in Chapter 9;
- a *trivial baseline* method estimating $\mathrm{dist}(A, B)$ as $\max\{|R_A|, |R_B|\}$
- the *alignment-free measures* $d_2, d_2^*, D_2^*, D_2, d_2^q, d_2^{q*}, D_2^q$, and D_2^{q*} developed in papers [271, 55, 25, 231]. Those methods were implemented by Matteo Comin's group and can be found at GitHub [156];
- the Mash tool [213].

Implementation of the methods presented in this thesis was done in Java with a maximum of shared code. The implementation was single-threaded. All five assembly algorithms were configured with the default parameters, and the current official C++ version was used. The implementation can be found online in the following repositories:

- `https://github.com/petrrysavy/readsIDA2016`
  (implementation for paper [237], see Chapter 5);

- `https://github.com/petrrysavy/readsDAMI2017`
  (implementation for paper [239], including the experimental setup, see Chapter 5);

- `https://github.com/petrrysavy/ida2017`
  (implementation for paper [238], see Chapter 8); and

- `https://github.com/petrrysavy/reference-free-phylogeny`
  (implementation of the complete tool, intended mostly for paper [240], see Chapter 9).

In the preliminary **tuning** experiment, the value $\theta' = 0.35$ (see Section 5.1.5) achieved the best Pearson's correlation coefficient on the training dataset, and we carried this value over to the testing experiments in the MESSGM method.

11.1 Evaluation Criteria

The evaluation criteria consist of

- the *Pearson's correlation coefficient* measuring the similarity of the distance matrices produced by the respective methods to the reference distance matrix;

- the *Fowlkes-Mallows index* [96] that measures the similarity between a tree produced based on the distance estimates and the reference tree defined by $\mathrm{dist}(A, B)$. The Fowlkes-Mallows index shows how much two hierarchical clusterings differ in structure. Both hierarchical trees are first cut into k clusters for $k = 2, 3, \ldots, n-1$. Then clusterings are compared based on the number of common objects among each pair of clusters. In this way, we obtain a set of values B_k that shows the distances of the trees at various levels;

- the *assembly time* (if applicable);

- the distance *matrix calculation time*;

- how many times was distance calculation successful (i.e., assembly produced at least one contig, and distance comparison *finished* in the time limit);

- for both distance calculation time and Pearson's correlation coefficient, we provide the average rank of the methods. This is because the time and correlation are hard to interpret, together with information on whether each algorithm provided valid results and finished within the time limit. Therefore, we sorted the results for each choice of coverage and read length, placing the methods that did not finish last, together with the case when the correlation was not defined (i.e., all sequences were equidistant). Then the rank is defined as the number of better methods in the sorted list. The rank was then averaged over coverage and read length values.

For hierarchical clustering, we used the UPGMA algorithm [269] and the neighbor-joining algorithm [249].

Name	Source	Read origin	Strand known	5'-3' known	n	c	l	Limit
Influenza	ENA [166]	i.i.d., uniform	✗	✗	13	0.1 to 100	3 to 500	2 hours
Various	ENA [166]	i.i.d., uniform	✗	✗	18	0.1 to 100	3 to 500	2 hours
Hepatitis	ENA [166]	[127]	✗	✓	81	$10, 30, 50$	$30, 70, 100$	1 day
Chromosomes	[1]	real-world	✗	✓	23	4.32	76	1 day

Table 11.1: An overview of the datasets used in the experiments.

11.2 Testing Data

The testing data contains five datasets. They are summarized in Table 11.1. The *influenza* dataset[1] contains 12 influenza virus genome sequences plus an outgroup sequence. The *various* dataset[2] contains 17 genomes of different viruses. Furthermore, we used an independent third *training* dataset[3] to tune the value of θ' (see Section 5.1.5) in the MESSGM distance. All the sequences were downloaded from the ENA repository [166] `http://www.ebi.ac.uk/ena`. The two datasets contained artificial reads sampled under the assumption that reads are i.i.d. and that c and l are constant. We artificially sampled the two datasets several times with an extensive range of coverage and read length values[4]. For averaging, we left out the three most outlying values of coverage c and read length l.

The *hepatitis* dataset contains 81 Hepatitis A segments. This time we used the ART [127] program to simulate sequencing to obtain read data for $(c, l) \in \{10, 30, 50\} \times \{30, 70, 100\}$. We sampled with higher coverage, and when using our methods, we downsampled to the coverage of 2.

The *chromosomes* dataset contains 23 regions of the human genome. The data were obtained from the *1000 Genomes Project* [1]. For each chromosome, we selected one 20 kbp long region and obtained reads that were sequenced from this region. We used the Ensembl [130] reference human genome to calculate the reference distance matrix. In this dataset, the average coverage per chromosome was 4.32.

A time limit was applied to the assembly step and the distance matrix calculation. Whenever an algorithm did not finish in time or the assembly step failed to produce any contig, we marked the attempt as unsuccessful and did not count it towards the average.

To generate read and contig sets from the genomes, we employ two strategies. One is 'idealized', based on error-free sampling from i.i.d. uniform distribution. The other simulates closely real-life erroneous sequencing conducted by Illumina technology and is facilitated by the ART [127] program. For the *influenza* and *various* datasets, we use the idealized read sampling option, the *hepatitis* dataset uses the ART program, and

[1] AF389115, AF389119, AY260942, AY260945, AY260949, AY260955, CY011131, CY011135, CY011143, HE584750, J02147, K00423 and outgroup AM050555. The genomes are available at `http://www.ebi.ac.uk/ena/data/view/<accession>`.

[2] AB073912, AB236320, AM050555, D13784, EU376394, FJ560719, GU076451, JN680353, JN998607, M14707, U06714, U46935, U66304, U81989, X05817, Y13051 and outgroup AY884005

[3] CY011119, CY011127, CY011140, FJ966081, AF144300, AF144300, J02057, AJ437618, FR717138, FJ869909, L00163, KJ938716, KP202150, D00664, HM590588, KM874295, $c = 4$, $l = 40$

[4] $(c, l) \in \{0.1, 0.3, 0.5, 0.7, 1, 1.5, 2, 2.5, 3, 4, 5, 7, 10, 15, 20, 30, 40, 50, 70, 100\} \times \{3, 5, 10, 15, 20, 25, 30, 40, 50, 70, 100, 150, 200, 500\}$

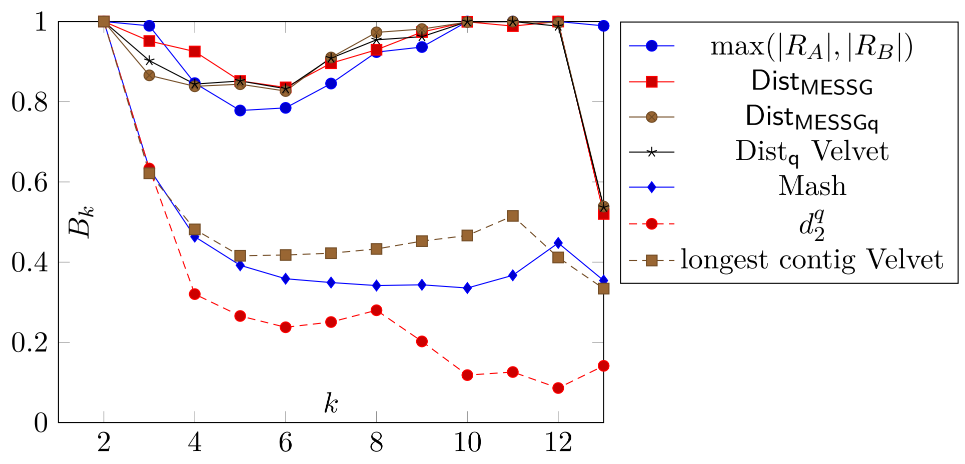


Figure 11.1: A plot of the average Fowlkes-Mallows index B_k versus k on the *various* dataset. The index compares trees generated by the neighbor-joining algorithm. The tree is compared with the tree generated from the original sequences. If all values are equal to 1, the structures of the trees are the same. We show only a selection of the methods and assembly algorithms.

the *chromosomes* dataset contains real-world reads.

The last dataset is named *e-coli*, and it contains 15 bacterial DNA sequences of the same species. In this case, no official assembly for each read bag exists; therefore, we compare the algorithms in quantitative measures that do not need assembly. The dataset works as proof of the concept that the method is applicable to real-world data.

11.3 Experimental Results

The main experimental results are shown in Tables 11.2, 11.3, 11.4, and 11.5, which show the average results on both datasets. For averaging on *influenza* and *various* datasets, we exclude the three most extreme values of coverage and read length. The extreme values were calculated so that, for example, Figure 11.7 shows the trend for a wider range of settings. The columns of the table show Pearson's correlation coefficient, run time, and the Fowlkes-Mallows index for selected levels. Figures 11.2, 11.1, 11.3, and 11.4 show the Fowlkes-Mallows index. Figures 11.5 and 11.6 show the plot of Pearson's correlation coefficient on coverage for influenza and various dataset, respectively. Similarly, Figures 11.7 and 11.8 show a plot of Pearson's correlation coefficient on read length.

11.3.1 Qualitative Evaluation

Pearson's correlation coefficient (column 'corr.') demonstrates the quality of the MESSG and MESSGM methods (Sections 5.1.4 and 5.1.5), which are the most developed versions of the read-based approach. The proposed method $\mathsf{Dist}_{\mathsf{CC}}(C_A, C_B)$ gives results of quality comparable to the latter two. For low-coverage data, the read-based estimation is more successful because it does not need assembled contigs. On the opposite, for high coverage data, $\mathsf{Dist}_{\mathsf{CC}}(C_A, C_B)$ produces results with higher correlation because it uses more information available in the read overlaps. The proposed methods are better than the baseline method $\max(|R_A|, |R_B|)$ and also than the simple approach that considers only the longest contig.

Findings based on Pearson's correlation coefficient are generally supported also by the Fowlkes-Mallows index (the last four columns). Figures 11.2, 11.1, 11.3, and 11.4

Data	method	finished	$\frac{\text{assem.}}{\text{ms}}$	$\frac{\text{distances}}{\text{ms}}$	$\frac{\textit{rank}}{\textit{dists.}}$	corr.	$\frac{\textit{rank}}{\textit{corr.}}$	$\frac{\text{NJ}}{B_4}$	$\frac{\text{NJ}}{B_8}$	$\frac{\text{UPGMA}}{B_4}$	$\frac{\text{UPGMA}}{B_8}$
Influenza	reference	112/112	0	2,602	29.2	1	1	1	1	1	1
	$\max(\lvert R_A\rvert, \lvert R_B\rvert)$	112/112	0	335	**13.3**	.801	46.5	.658	.319	.67	.319
	$\text{Dist}_{\text{MESSG}}(R_A, R_B)$	107/112	0	899,270	60.1	**.983**	**9.7**	.998	.997	1	1
	$\text{Dist}_{\text{MESSGq}}(R_A, R_B)$	112/112	0	50,808	42.5	.966	27.9	.998	.967	.999	.977
	$\text{Dist}_{\text{CC}}(C_A, C_B)$ ABySS	87/112	21,628	17,469	44.7	.951	37.7	.983	.799	.983	.868
	$\text{Dist}_{\text{CC}}(C_A, C_B)$ Edena	72/112	285	18,483	47.2	.96	41.7	.996	.845	.997	.878
	$\text{Dist}_{\text{CC}}(C_A, C_B)$ SPAdes	43/112	13,529	22,661	56.8	.973	49.4	.989	.932	.994	.948
	$\text{Dist}_{\text{CC}}(C_A, C_B)$ SSAKE	68/112	2,079	17,735	48.5	.944	44.5	.974	.838	.972	.867
	$\text{Dist}_{\text{CC}}(C_A, C_B)$ Velvet	110/112	385	23,567	43.8	.958	31.6	.991	.906	.994	.91
	$\text{Dist}(T_A, T_B)$ ABySS	112/112	20,900	508,511	54.5	.979	17	.998	.997	.997	.998
	$\text{Dist}(T_A, T_B)$ Edena	112/112	233	430,385	53.4	.98	**15.1**	.998	.997	1	.997
	$\text{Dist}(T_A, T_B)$ SPAdes	112/112	12,380	625,883	56.7	**.983**	**8.9**	.998	.997	.997	1
	$\text{Dist}(T_A, T_B)$ SSAKE	112/112	1,655	552,860	53.4	**.98**	15.7	.998	.997	.997	.997
	$\text{Dist}(T_A, T_B)$ Velvet	111/112	378	749,033	57.9	.971	29.1	.998	.987	1	.994
	$\text{Dist}_{\text{q}}(T_A, T_B)$ ABySS	112/112	5,583	23,565	35.3	.963	32	1	.925	1	.934
	$\text{Dist}_{\text{q}}(T_A, T_B)$ Edena	112/112	264	16,090	35.6	.966	31.1	1	.942	1	.949
	$\text{Dist}_{\text{q}}(T_A, T_B)$ SPAdes	112/112	14,345	28,690	37.6	.971	23.1	1	.944	1	.954
	$\text{Dist}_{\text{q}}(T_A, T_B)$ SSAKE	112/112	2,302	27,515	36.3	.967	28.7	1	.951	1	.956
	$\text{Dist}_{\text{q}}(T_A, T_B)$ Velvet	112/112	446	22,478	37.7	.956	35.3	.998	.973	.996	.977
	Mash	112/112	0	**101**	**9**	.679	46.8	.438	.61	.476	.575
	d_2	112/112	0	335	17	.708	48.1	.427	.988	.303	.665
	d_2^*	112/112	0	389	18.3	.837	44.7	.402	.899	.378	.712
	d_2^{q*}	112/112	0	328	16.6	.631	50.3	.32	.272	.365	.105
	D_2	112/112	0	374	17.2	.443	59.2	.324	.001	.675	.316
	D_2^*	112/112	0	318	16.6	−.102	62.6	.436	.002	.504	.004
	D_2^{q*}	112/112	0	312	16.5	0	63.6	.32	.272	.365	.123
	d_2^q	112/112	0	**281**	15.2	.631	50.3	.32	.272	.365	.105
	D_2^q	112/112	0	327	16.5	0	63.6	.32	.272	.365	.123
	longest contig ABySS	87/112	21,628	1,182	26.4	.67	45.5	.621	.432	.629	.427
	longest contig Edena	72/112	285	1,055	33.3	.675	49.2	.624	.465	.636	.472
	longest contig SPAdes	43/112	13,529	1,465	48.2	.751	51.5	.713	.555	.743	.558
	longest contig SSAKE	68/112	2,079	785	32.5	.664	51	.606	.357	.594	.352
	longest contig Velvet	110/112	385	**38**	**7.5**	.569	53.8	.457	.23	.452	.234

Table 11.2: Average runtime, Pearson's correlation coefficient between the distance matrices, and the Fowlkes-Mallows index for $k = 4$ and $k = 8$ on the *influenza* dataset. The 'reference' method calculates the distances of the original sequences. For an explanation of the rank column, see Section 11.1. Note that the table is truncated for space reasons. Therefore, the rank columns show higher numbers than expected. The excluded rows mostly show the behavior of the presented methods on error-free idealized artificial contigs (see Chapter 12).

Data	method	finished	assem. ms	distances ms	*rank* *dists.*	corr.	*rank* *corr.*	NJ B_4	NJ B_8	UPGMA B_4	UPGMA B_8
Various	reference	112/112	0	57,099	16.9	1	1	1	1	1	1
	$\max(\lvert R_A\rvert, \lvert R_B\rvert)$	112/112	0	847	**4.1**	.907	14.1	.846	.924	.671	.655
	$\mathsf{Dist}_{\mathsf{MESSG}}(R_A, R_B)$	64/112	0	1,299,980	24.8	**.933**	13	.925	.93	.622	.882
	$\mathsf{Dist}_{\mathsf{MESSGq}}(R_A, R_B)$	109/112	0	605,647	20	.927	**8.7**	.838	.973	.659	.768
	$\mathsf{Dist}_{\mathsf{CC}}(C_A, C_B)$ ABySS	72/112	7,959	779,370	22.4	.921	14.7	.843	.926	.629	.753
	$\mathsf{Dist}_{\mathsf{CC}}(C_A, C_B)$ Edena	57/112	679	1,070,977	23.8	.92	16.9	.858	.932	.629	.773
	$\mathsf{Dist}_{\mathsf{CC}}(C_A, C_B)$ SPAdes	58/112	7,342	899,891	22.9	.923	15.5	.857	.944	.635	.798
	$\mathsf{Dist}_{\mathsf{CC}}(C_A, C_B)$ SSAKE	108/112	1,235	749,197	20.7	.928	**5.4**	.839	.922	.632	.902
	$\mathsf{Dist}_{\mathsf{CC}}(C_A, C_B)$ Velvet	34/112	17,783	1,239,632	25.5	.917	19.8	.877	.945	.638	.838
	$\mathrm{Dist}(T_A, T_B)$ ABySS	70/112	5,086	1,684,468	24.7	.928	13.2	.862	.923	.626	.82
	$\mathrm{Dist}(T_A, T_B)$ Edena	69/112	168	1,681,308	24.6	**.932**	12.3	.916	.934	.629	.876
	$\mathrm{Dist}(T_A, T_B)$ SPAdes	70/112	3,449	1,666,859	24.1	.932	12.2	.913	.931	.63	.874
	$\mathrm{Dist}(T_A, T_B)$ SSAKE	64/112	568	1,635,059	26.1	.919	12.9	.831	.909	.611	.892
	$\mathrm{Dist}(T_A, T_B)$ Velvet	67/112	13,897	1,584,465	24.9	.932	12.4	.92	.929	.623	.882
	$\mathrm{Dist}_{\mathsf{q}}(T_A, T_B)$ ABySS	110/112	10,937	252,197	16.5	.919	11.7	.85	.932	.65	.755
	$\mathrm{Dist}_{\mathsf{q}}(T_A, T_B)$ Edena	112/112	790	360,304	15.9	.921	10.9	.843	.941	.661	.752
	$\mathrm{Dist}_{\mathsf{q}}(T_A, T_B)$ SPAdes	110/112	6,197	316,445	16.2	.922	10.7	.852	.941	.65	.766
	$\mathrm{Dist}_{\mathsf{q}}(T_A, T_B)$ SSAKE	111/112	2,231	428,540	17.9	**.934**	**6.4**	.844	.954	.726	.847
	$\mathrm{Dist}_{\mathsf{q}}(T_A, T_B)$ Velvet	110/112	19,583	355,127	16.3	.922	10.1	.845	.945	.646	.765
	Mash	84/112	0	**562**	8.3	.664	17.8	.464	.342	.396	.315
	d_2	109/112	0	741	8.7	.269	23.8	.469	.355	.358	.41
	d_2^*	110/112	0	756	8.5	.442	20.1	.453	.316	.378	.19
	d_2^{q*}	109/112	0	**721**	8	.573	17.4	.32	.28	.446	.099
	D_2	107/112	0	739	8.4	.294	22.5	.463	.067	.671	.641
	D_2^*	110/112	0	734	7.8	.291	22.6	.474	.114	.604	.428
	D_2^{q*}	110/112	0	774	7.7	0	25	.32	.28	.446	.122
	d_2^q	109/112	0	760	7.6	.573	17.4	.32	.28	.446	.099
	D_2^q	111/112	0	743	**7.4**	0	25	.32	.28	.446	.122
	longest contig ABySS	72/112	7,959	6,439	13.6	.562	20.1	.495	.345	.512	.443
	longest contig Edena	57/112	679	18,206	17.8	.571	20.8	.537	.389	.542	.452
	longest contig SPAdes	58/112	7,342	14,861	15.6	.626	20.1	.533	.377	.548	.465
	longest contig SSAKE	108/112	1,235	**385**	**3.5**	.386	20.9	.482	.433	.448	.166
	longest contig Velvet	34/112	17,783	34,858	22.5	.681	21.4	.625	.498	.632	.614

Table 11.3: Average runtime, Pearson's correlation coefficient between the distance matrices, and the Fowlkes-Mallows index for $k = 4$ and $k = 8$ on the *various* dataset. The table was generated under the same conditions as Table 11.2.

Data	method	finished	assem. ms	distances ms	*rank* *dists.*	corr.	*rank* *corr.*	NJ B_4	NJ B_8	UPGMA B_4	UPGMA B_8
Hepatitis	reference	9/9	0	1,748,984	16.9	1	1	1	1	1	1
	$\max(\lvert R_A \rvert, \lvert R_B \rvert)$	9/9	0	29,340	5.8	.181	19.3	.724	.828	.553	.368
	$\mathsf{Dist}_{\mathsf{MESSG}}(R_A, R_B)$	9/9	0	42,332,682	21.1	.965	8.3	1	.904	.99	.954
	$\mathsf{Dist}_{\mathsf{MESSGq}\alpha}(R_A, R_B)$	9/9	0	1,118,585	15.4	.897	14.2	1	.935	.913	.942
	$\mathsf{Dist}_{\mathsf{CC}}(C_A, C_B)$ ABySS	9/9	35,145	48,256,963	22.3	.949	9.9	1	.912	.948	.937
	$\mathsf{Dist}_{\mathsf{CC}}(C_A, C_B)$ Edena	9/9	7,038	44,548,818	21.2	.892	**5.1**	1	.839	.954	.912
	$\mathsf{Dist}_{\mathsf{CC}}(C_A, C_B)$ SPAdes	2/9	76,514	31,517,537	23.1	.869	20.6	1	.893	.87	.931
	$\mathsf{Dist}_{\mathsf{CC}}(C_A, C_B)$ SSAKE	5/9	69,156	55,880,178	23.2	.901	15.1	1	.945	.976	.947
	$\mathsf{Dist}_{\mathsf{CC}}(C_A, C_B)$ Velvet	4/9	11,090	59,898,794	23.9	**.98**	14.6	1	.988	1	.974
	$\mathrm{Dist}(T_A, T_B)$ ABySS	0/9	NaN	NaN	24.4	NaN	24.4	NaN	NaN	NaN	NaN
	$\mathrm{Dist}(T_A, T_B)$ Edena	0/9	NaN	NaN	24.4	NaN	24.4	NaN	NaN	NaN	NaN
	$\mathrm{Dist}(T_A, T_B)$ SPAdes	0/9	NaN	NaN	24.4	NaN	24.4	NaN	NaN	NaN	NaN
	$\mathrm{Dist}(T_A, T_B)$ SSAKE	0/9	NaN	NaN	24.4	NaN	24.4	NaN	NaN	NaN	NaN
	$\mathrm{Dist}(T_A, T_B)$ Velvet	0/9	NaN	NaN	24.4	NaN	24.4	NaN	NaN	NaN	NaN
	$\mathsf{Dist}_{\mathsf{q}\alpha}(T_A, T_B)$ ABySS	9/9	48,194	520,227	12.7	.957	10.6	1	.932	.823	.89
	$\mathsf{Dist}_{\mathsf{q}\alpha}(T_A, T_B)$ Edena	9/9	12,889	520,091	12.8	.929	10.3	.976	.942	.835	.823
	$\mathsf{Dist}_{\mathsf{q}\alpha}(T_A, T_B)$ SPAdes	9/9	130,268	373,244	11.7	.911	13.6	.966	.843	.75	.862
	$\mathsf{Dist}_{\mathsf{q}\alpha}(T_A, T_B)$ SSAKE	9/9	88,516	615,615	14.2	.901	12.9	.961	.937	.851	.862
	$\mathsf{Dist}_{\mathsf{q}\alpha}(T_A, T_B)$ Velvet	9/9	27,814	1,729,999	16.9	.955	7.9	1	.994	.934	.941
	Mash	9/9	0	**2,350**	**1.4**	.967	8.1	1	.918	.964	.966
	d_2	7/9	0	**27,145**	9.1	**.973**	10.1	1	.864	.982	.963
	d_2^*	7/9	0	28,189	10.9	.972	10.4	1	.902	.893	.972
	d_2^{q*}	9/9	0	29,296	**6.4**	.972	**6.8**	1	.896	.894	.973
	D_2	8/9	0	30,458	8.2	−.181	21.2	.93	.429	.662	.545
	D_2^*	7/9	0	27,718	9.2	−.783	22.9	.929	.407	.662	.545
	D_2^{q*}	9/9	0	27,151	**4.6**	−.782	23.4	.922	.404	.662	.545
	d_2^q	9/9	0	27,885	6.7	**.973**	**5.1**	1	.87	.984	.96
	D_2^q	7/9	0	31,481	12.1	−.187	22.3	.931	.415	.662	.545
	longest contig ABySS	9/9	35,145	2,493,455	18.2	.53	18.1	.946	.685	.654	.686
	longest contig Edena	9/9	7,038	1,581,613	15.6	.515	17.8	.918	.76	.783	.8
	longest contig SPAdes	2/9	76,514	3,242,365	21.1	.395	23	.867	.776	.822	.775
	longest contig SSAKE	5/9	69,156	764,321	16.7	.334	21	.862	.624	.712	.661
	longest contig Velvet	4/9	11,090	**515**	13.1	.296	21.3	.919	.473	.637	.58

Table 11.4: Average runtime, Pearson's correlation coefficient between the distance matrices, and the Fowlkes-Mallows index for $k = 4$ and $k = 8$ on the *hepatitis* dataset. The table was generated under the same conditions as Table 11.2.

Data	method	finished	assem.	distances	*rank*	corr.	*rank*	NJ	NJ	UPGMA	UPGMA
			ms	ms	*dists.*		*corr.*	B_4	B_8	B_4	B_8
Chroms	reference	1/1	0	668,767	20	1	1	1	1	1	1
	$\max(\lvert R_A\rvert, \lvert R_B\rvert)$	1/1	0	2,184	13	.331	18	.613	.298	.64	.404
	$\mathsf{Dist}_{\mathsf{MESSG}}(R_A, R_B)$	1/1	0	23,758,416	24	.848	14	.585	.26	.408	.227
	$\mathsf{Dist}_{\mathsf{MESSGq}\alpha}(R_A, R_B)$	1/1	0	202,517	19	.825	15	.9	.247	.404	.227
	$\mathsf{Dist}_{\mathsf{CC}}(C_A, C_B)$ ABySS	0/1	NaN	NaN	27	NaN	25	NaN	NaN	NaN	NaN
	$\mathsf{Dist}_{\mathsf{CC}}(C_A, C_B)$ Edena	0/1	NaN	NaN	27	NaN	25	NaN	NaN	NaN	NaN
	$\mathsf{Dist}_{\mathsf{CC}}(C_A, C_B)$ SPAdes	0/1	NaN	NaN	27	NaN	25	NaN	NaN	NaN	NaN
	$\mathsf{Dist}_{\mathsf{CC}}(C_A, C_B)$ SSAKE	0/1	NaN	NaN	27	NaN	25	NaN	NaN	NaN	NaN
	$\mathsf{Dist}_{\mathsf{CC}}(C_A, C_B)$ Velvet	0/1	NaN	NaN	27	NaN	25	NaN	NaN	NaN	NaN
	Dist(T_A, T_B) ABySS	1/1	17,838	24,085,638	25	.911	6	.638	.342	.774	.307
	Dist(T_A, T_B) Edena	1/1	1,063	23,404,639	22	.879	12	.676	.211	.553	.334
	Dist(T_A, T_B) SPAdes	1/1	22,898	23,757,934	23	.873	13	.676	.211	.553	.334
	Dist(T_A, T_B) SSAKE	1/1	51,604	20,576,658	21	.903	7	.805	.342	.359	.334
	Dist(T_A, T_B) Velvet	1/1	7,866	24,668,207	26	.902	8	.585	.26	.619	.334
	$\mathsf{Dist}_{\mathsf{q}\alpha}(T_A, T_B)$ ABySS	1/1	17,838	144,725	18	.914	5	.805	.211	.488	.334
	$\mathsf{Dist}_{\mathsf{q}\alpha}(T_A, T_B)$ Edena	1/1	1,063	126,282	14	.887	9	.805	.211	.455	.334
	$\mathsf{Dist}_{\mathsf{q}\alpha}(T_A, T_B)$ SPAdes	1/1	22,898	127,061	16	.881	11	.805	.329	.553	.208
	$\mathsf{Dist}_{\mathsf{q}\alpha}(T_A, T_B)$ SSAKE	1/1	51,604	126,565	15	**.914**	**4**	.805	.211	.819	.43
	$\mathsf{Dist}_{\mathsf{q}\alpha}(T_A, T_B)$ Velvet	1/1	7,866	127,200	17	.881	10	.805	.222	.553	.208
	Mash	1/1	0	**173**	**3**	.33	19	.599	.382	.588	.307
	d_2	1/1	0	696	5	.258	22	.706	.368	.503	.497
	d_2^*	1/1	0	697	6	.301	20	.805	.303	.503	.328
	d_2^{q*}	1/1	0	697	6	**.959**	**2**	.805	.316	.519	.283
	D_2	1/1	0	692	4	$3.933 \cdot 10^{-2}$	23	.9	.283	.64	.404
	D_2^*	1/1	0	1,018	11	$2.919 \cdot 10^{-2}$	24	.852	.273	.471	.358
	D_2^{q*}	1/1	0	714	8	0	25	.805	.316	.519	.283
	d_2^q	1/1	0	734	9	**.959**	**2**	.805	.316	.519	.283
	D_2^q	1/1	0	1,010	10	0	25	.805	.316	.519	.283
	longest contig ABySS	1/1	17,838	1,213	12	.34	17	.706	.364	.55	.38
	longest contig Edena	0/1	NaN	NaN	27	NaN	25	NaN	NaN	NaN	NaN
	longest contig SPAdes	0/1	NaN	NaN	27	NaN	25	NaN	NaN	NaN	NaN
	longest contig SSAKE	1/1	51,604	**152**	**2**	.297	21	.588	.302	.66	.538
	longest contig Velvet	1/1	7,866	**31**	**1**	.574	16	.805	.404	.519	.158

Table 11.5: Average runtime, Pearson's correlation coefficient between the distance matrices, and the Fowlkes-Mallows index for $k = 4$ and $k = 8$ on the *chroms* dataset. The table was generated under the same conditions as Table 11.2.

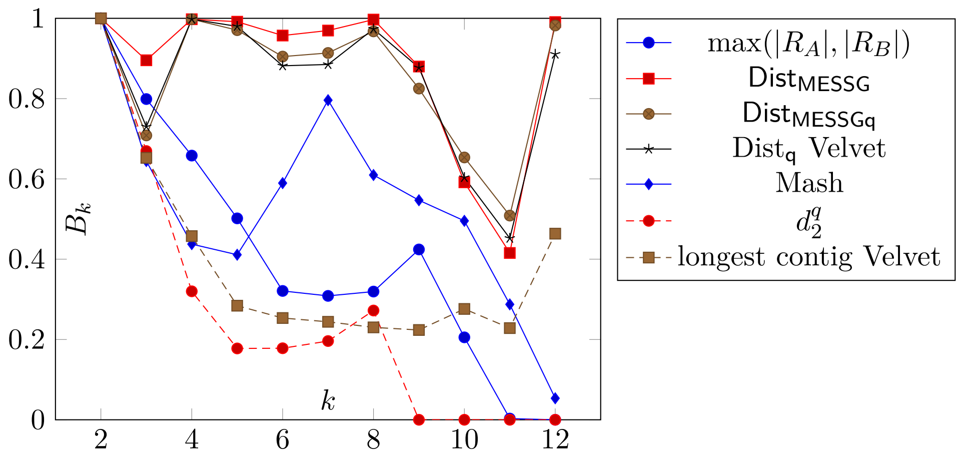


Figure 11.2: A plot of the average Fowlkes-Mallows index B_k versus k on the *influenza* dataset. The index compares trees generated by the neighbor-joining algorithm.

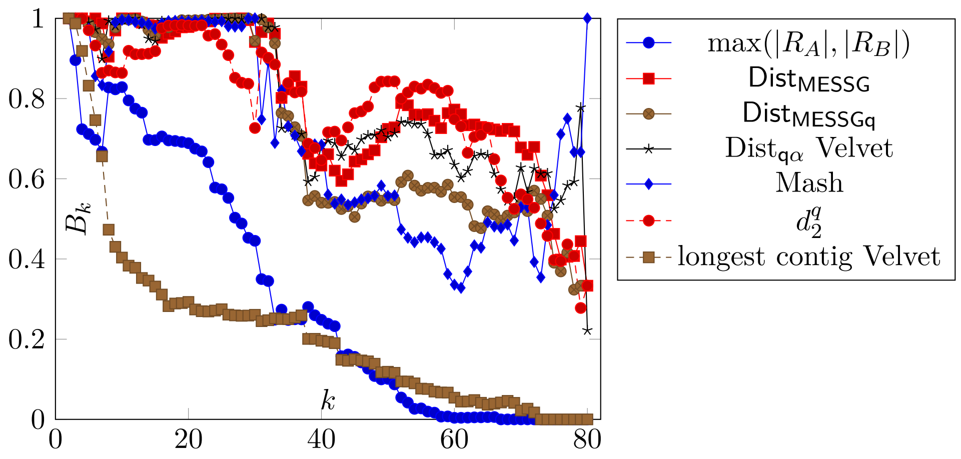


Figure 11.3: A plot of the average Fowlkes-Mallows index B_k versus k on the *hepatitis* dataset. The index compares trees generated by the neighbor-joining algorithm.

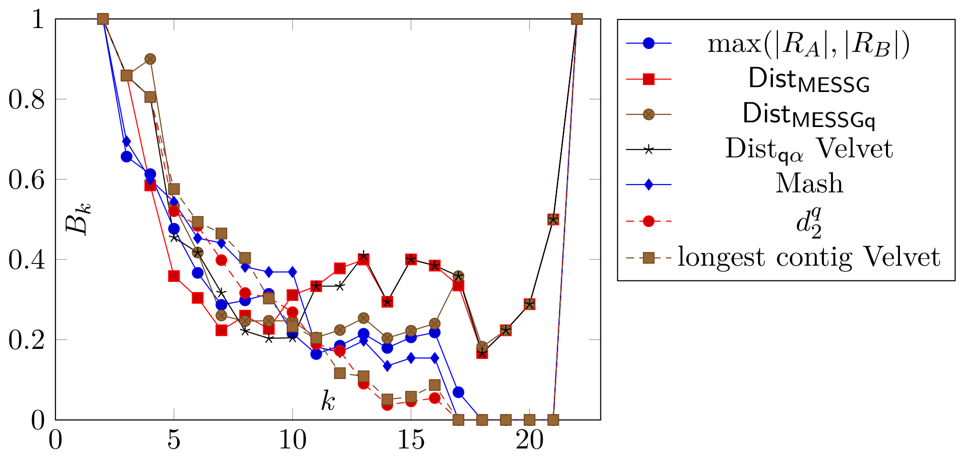


Figure 11.4: A plot of the average Fowlkes-Mallows index B_k versus k on the *chromosomes* dataset. The index compares trees generated by the neighbor-joining algorithm.

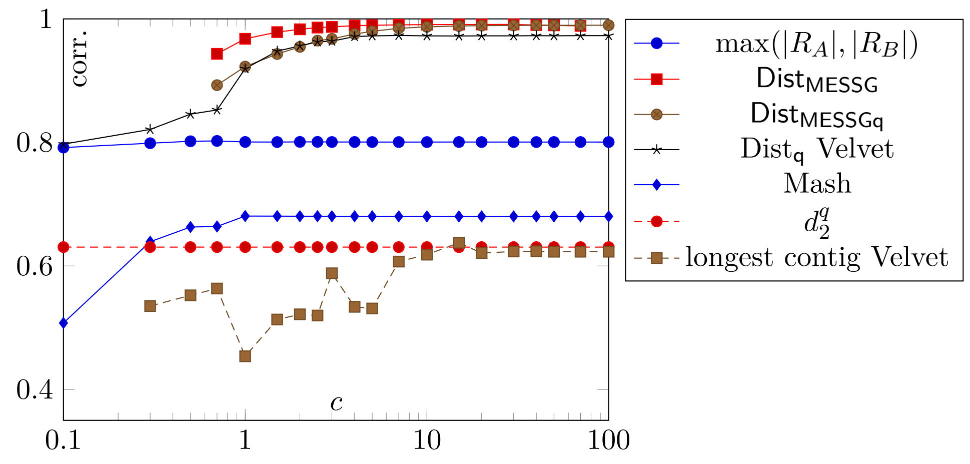


Figure 11.5: A plot of the average Pearson's correlation coefficient for several choices of coverage values on the *influenza* dataset.

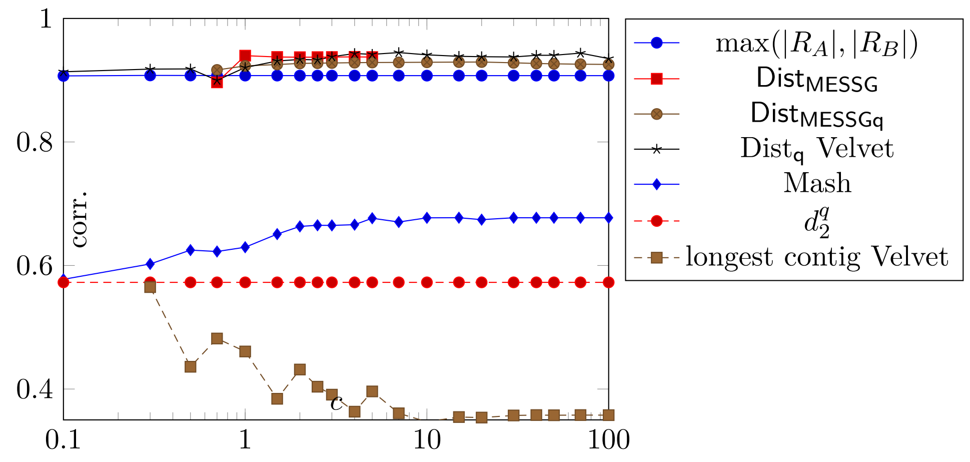


Figure 11.6: A plot of the average Pearson's correlation coefficient for several choices of coverage values on the *various* dataset.

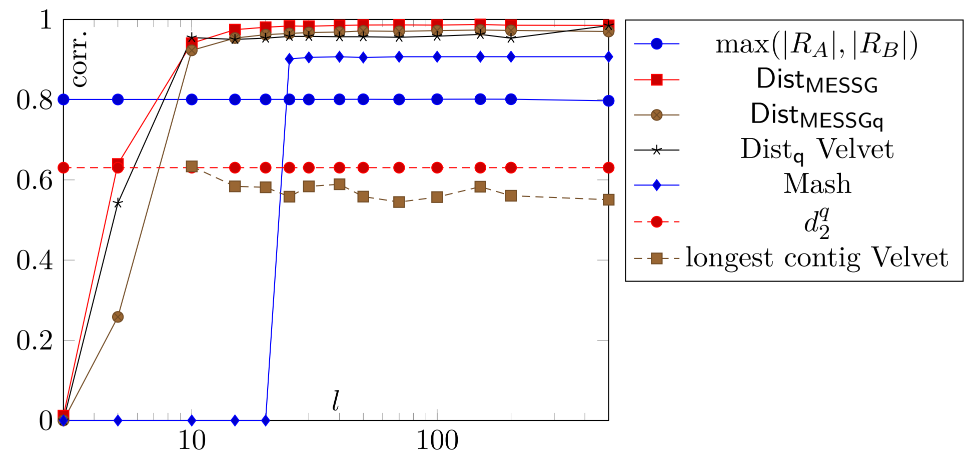


Figure 11.7: A plot of the average Pearson's correlation coefficient for several choices of read length on the *influenza* dataset.

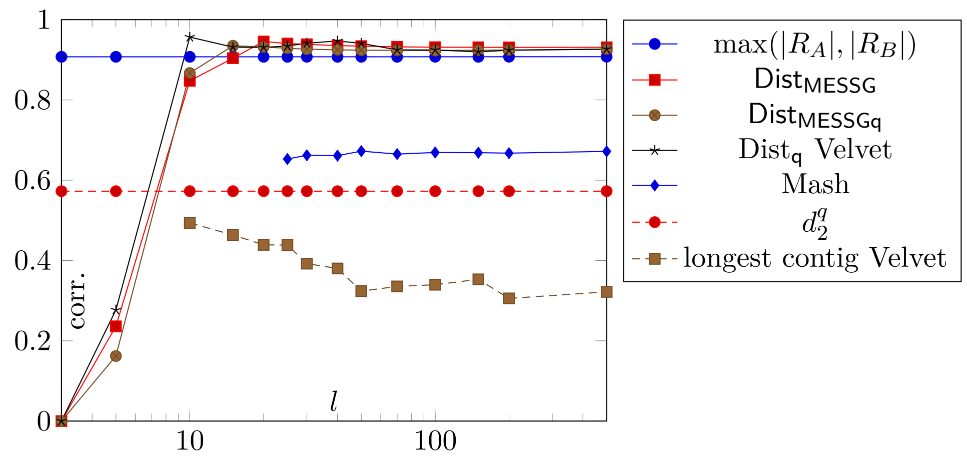


Figure 11.8: A plot of the average Pearson's correlation coefficient for several choices of read length on the *various* dataset.

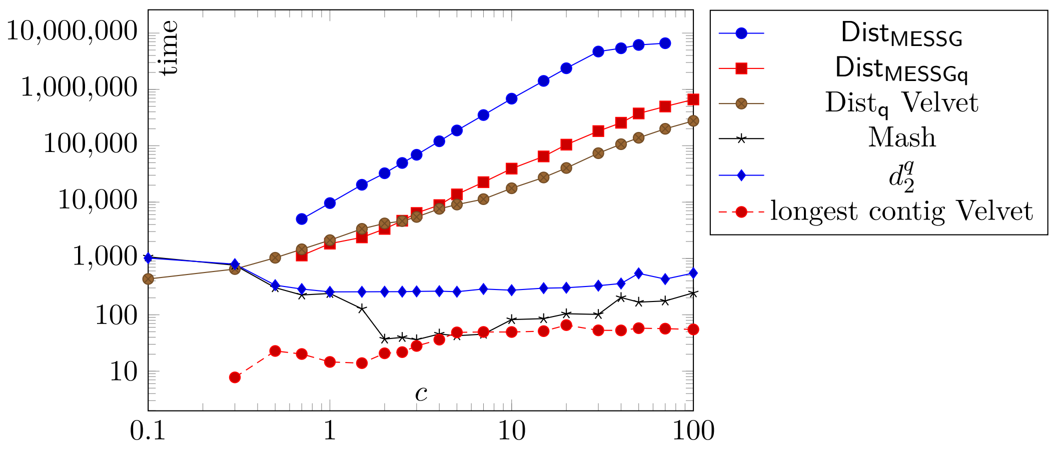


Figure 11.9: A plot of the distance matrix calculation time for several choices of coverage values on the *influenza* dataset.

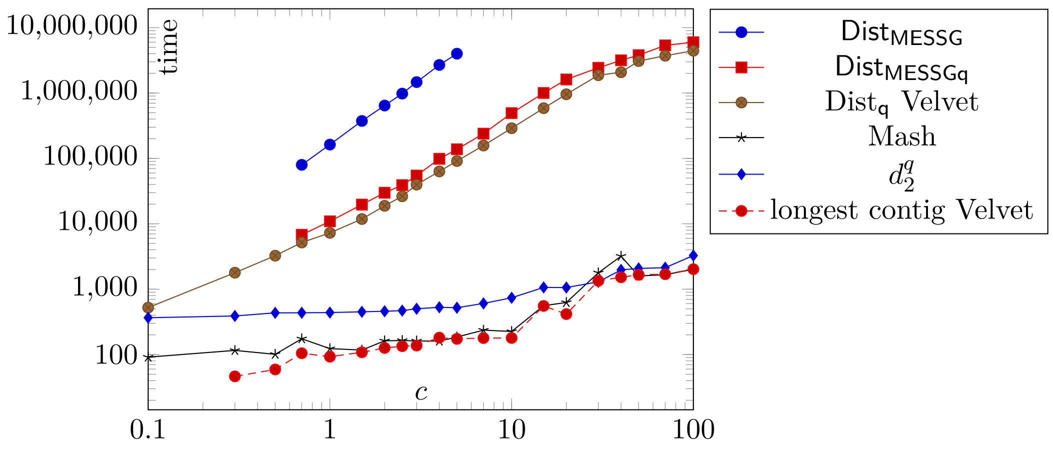


Figure 11.10: A plot of the distance matrix calculation time for several choices of coverage values on the *various* dataset.

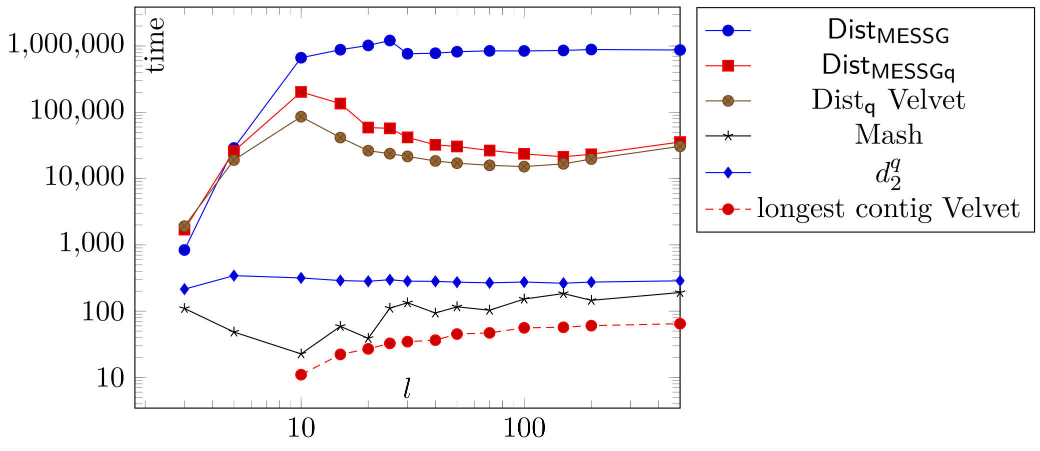


Figure 11.11: A plot of the distance matrix calculation time for several choices of read length on the *influenza* dataset.

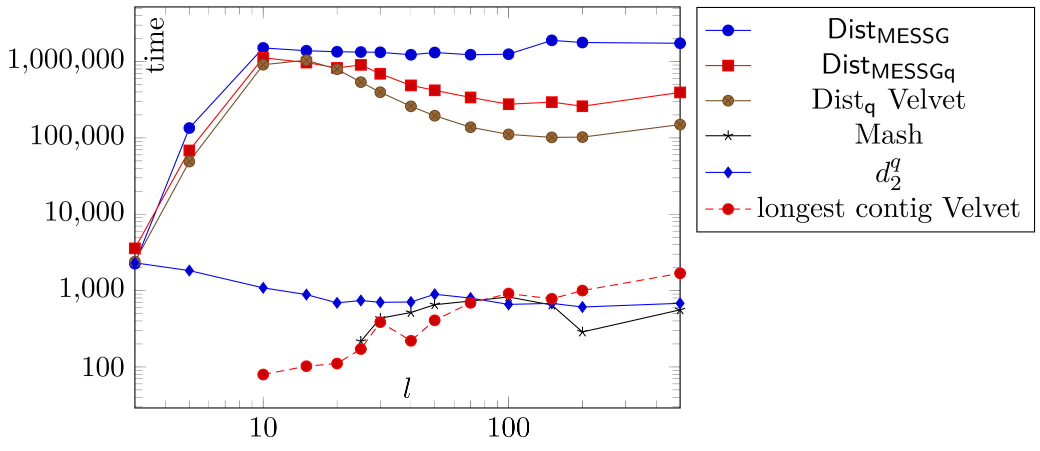


Figure 11.12: A plot of the distance matrix calculation time for several choices of read length on the *various* dataset.

provide a more detailed insight into the Fowlkes-Mallows values graphically for all the tree levels.

The *chromosomes* dataset shows how our method is dependent on the estimates of the lengths of the original sequences (obtained from (2.11)) are good. While method MES provides good estimates, the accuracy drops after applying the scaling from Section 5.1.3. Coverage is not constant on all read bags, and therefore, the results had to be provided with the information that all the original sequences had the same length.

11.3.2 Dependence on Read Length and Coverage

Figures 11.5 and 11.6 analyze the accuracy of different algorithms in dependence on coverage. We see that our approaches produce good results for coverage around 2. Therefore, on the *hepatitis* dataset, we sampled down to $c = 2$. The assembly algorithms, however, require several times higher coverage to produce results of the same accuracy. Figures 11.5 and 11.6 show that our method produces good estimates for shorter reads than the assembly methods.

Figures 11.9 and 11.10 show runtime dependence of the proposed methods on coverage in a logarithmic scale. It can be seen that the presented approach is slower than the alignment-free approaches; however, with sampling to $c = 2$, the difference is smaller. Similarly, Figures 11.11 and 11.12 show the dependence of runtime on the read length. The results; however, need to be compared with the context of the finished column of the result tables (Tables 11.2, 11.3, 11.4, and 11.5) - when a method does not finish at all, it is not counted towards average.

In the proposed methods, we can notice that there is a peak at a read length of around $l = 10$, and then the runtime drops. This is because of the comparison based on the q-gram distance. For very short sequences (around 10), calculating the Levenshtein distance means no overhead in the runtime. Therefore, the effect of embedding the reads (see Section 10.2 for details) into a smaller space does not improve the runtime. As the read length grows, the runtime of the Levenshtein distance grows quadratically while the runtime of the q-gram distance grows only linearly.

11.3.3 Runtime

Columns 4-5 of Tables 11.2, 11.3, 11.4, and 11.5 indicate that the exact variant (MESSG) of our approach is systematically slower concerning absolute runtime than the approaches based on sequence assembly, despite the NP-hard complexity of the latter task. However, the approximated version (MESSGq, including sampling) of our approach did produce results in a time comparable to the assembly time. The cost for this runtime improvement was only a small decrease in accuracy. Moreover, the MESSGq approach was faster than calculating the reference distance matrix on larger datasets. The numbers also show that our asymptotic complexity estimate in Section 5.2.1 is generally correct: the ratio between the time spent on calculating the distances on the one hand, and the runtime of the reference method, on the other hand, is approximately α^2. On the opposite, $\mathsf{Dist}_{\mathsf{CC}}(C_A, C_B)$ is approximately $10\times$ slower than the reference method. $\mathsf{Dist}_{\mathsf{CC}}(C_A, C_B)$ and the reference method have to fill dynamic programming tables of approximately the same sizes; therefore, their run times differ only by a multiplicative constant, as the proposed method has to fill entries in three tables instead of one.

11.3.4 Comparative Experiments on the *E. coli* Dataset

Table 11.6 shows the pairwise distance matrix correlations between the selected methods. As the reads come from the real-world dataset and are sequenced for several variants of the E. coli bacteria, there is no reference sequence for the samples. Therefore, pairwise comparison is used. From the quantitative perspective, Table 11.7 shows that the presented methods can calculate the distance matrix in a time comparable to the alignment-free methods and faster than the sequences can be assembled with default settings of the assembly algorithms.

11.3.5 Discussion

[240]

The results shown in Tables 11.2, 11.3, 11.4, and 11.5 indicate that the $\mathsf{Dist}_{\mathsf{q}}$ method, including the efficiency optimizations from Chapter 10 yields valid results in a broader range of cases than other methods. Compared to the alignment-free approaches, the method works well on both similar as well as dissimilar genomes. In contrast, the performance of the alignment-free approaches is worse on dissimilar genomes. Figures 11.5, 11.6, 11.7, and 11.8 further illustrate that the method is capable of producing reasonable results for low coverage data and very short reads.

Tables 11.2, 11.3, 11.4, and 11.5 show that compared to the simpler versions of the method ($\mathsf{Dist}_{\mathsf{MESSGq}\alpha}$, $\mathsf{Dist}_{\mathsf{CC}}$), the combination is faster. However, the method is, as expected, slower than the alignment-free methods. Concerning the correlation, the method performs better on one dataset and worse on one dataset than the read-read and contig-contig versions. Compared to the alignment-free methods, the Dist_q method did better on two datasets while worse on one dataset.

The changes proposed in Chapter 10 improve the runtime up to three magnitudes. Note that in the case of the hepatitis dataset, the exact variant of the method finishes only in two cases, while the faster variant finishes in all cases.

The real-world experiments show that our method successfully approximates the Levenshtein distance between the compared sequences.

The Folwkes-Mallows index does not always correlate with Pearson's correlation coefficient. There are reasons for this behavior. Firstly, the split of the clustered objects into clusters is presented only for selected depths, which means that the clustering might be good on a coarse level while bad on a fine level. In contrast, the correlation coefficient always compares whole matrices. Secondly, the data itself can affect the correlation. When many distances are almost the same, the correlation might be high, but small perturbations in the distances might cause the trees to look similar. In a different scenario, the trees might be the same, but some degree of freedom in distance values might decrease the correlations.

	$\mathsf{Dist}_{\mathsf{MESSGq}\alpha}$	$\mathsf{Dist}_{\mathsf{MESSGMq}\alpha}$	co-phylog	Mash	d_2	d_2^*	d_2^q	d_2^{q*}	D_2	D_2^*	D_2^q	D_2^{q*}	$Dist_q$ ABySS	Dist_q Edena	Dist_q SSAKE	Dist_q Velvet
$\mathsf{Dist}_{\mathsf{MESSGq}\alpha}$	1.000	1.000	0.831	0.943	0.269	0.302	0.269	0.304	0.683	0.867	0.684	0.868	1.000	1.000	1.000	1.000
$\mathsf{Dist}_{\mathsf{MESSGMq}\alpha}$	1.000	1.000	0.831	0.943	0.269	0.302	0.269	0.304	0.683	0.867	0.684	0.868	1.000	1.000	1.000	1.000
co-phylog	0.831	0.831	1.000	0.925	0.161	0.215	0.161	0.216	0.711	0.813	0.712	0.813	0.829	0.829	0.829	0.829
Mash	0.943	0.943	0.925	1.000	0.286	0.339	0.286	0.341	0.734	0.877	0.735	0.878	0.942	0.942	0.942	0.942
d_2	0.269	0.269	0.161	0.286	1.000	0.956	1.000	0.956	0.240	0.246	0.243	0.249	0.270	0.270	0.271	0.270
d_2^*	0.302	0.302	0.215	0.339	0.956	1.000	0.956	1.000	0.322	0.303	0.325	0.307	0.302	0.302	0.303	0.302
d_2^q	0.269	0.269	0.161	0.286	1.000	0.956	1.000	0.956	0.240	0.246	0.243	0.249	0.270	0.270	0.271	0.270
d_2^{q*}	0.304	0.304	0.216	0.341	0.956	1.000	0.956	1.000	0.323	0.305	0.326	0.308	0.304	0.304	0.305	0.304
D_2	0.683	0.683	0.711	0.734	0.240	0.322	0.240	0.323	1.000	0.949	1.000	0.949	0.683	0.683	0.683	0.683
D_2^*	0.867	0.867	0.813	0.877	0.246	0.303	0.246	0.305	0.949	1.000	0.950	1.000	0.866	0.866	0.866	0.866
D_2^q	0.684	0.684	0.712	0.735	0.243	0.325	0.243	0.326	1.000	0.950	1.000	0.949	0.684	0.684	0.684	0.684
D_2^{q*}	0.868	0.868	0.813	0.878	0.249	0.307	0.249	0.308	0.949	1.000	0.949	1.000	0.868	0.868	0.868	0.868
Dist_q ABySS	1.000	1.000	0.829	0.942	0.270	0.302	0.270	0.304	0.683	0.866	0.684	0.868	1.000	1.000	1.000	1.000
Dist_q Edena	1.000	1.000	0.829	0.942	0.270	0.302	0.270	0.304	0.683	0.866	0.684	0.868	1.000	1.000	1.000	1.000
Dist_q SSAKE	1.000	1.000	0.829	0.942	0.271	0.303	0.271	0.305	0.683	0.866	0.684	0.868	1.000	1.000	1.000	1.000
Dist_q Velvet	1.000	1.000	0.829	0.942	0.270	0.302	0.270	0.304	0.683	0.866	0.684	0.868	1.000	1.000	1.000	1.000

Table 11.6: Pairwise correlations of distance matrices on the *E. coli* dataset.

Method	Time (s, one thread)	Time (s, parallel)
$\mathsf{Dist}_{\mathsf{MESSGq}\alpha}$	8,908	NaN
$\mathsf{Dist}_{\mathsf{MESSGMq}\alpha}$	8,908	NaN
co-phylog	NaN	598
Mash	NaN	500
d_2	3,343	NaN
d_2^*	3,311	NaN
d_2^q	3,331	NaN
d_2^{q*}	3,346	NaN
D_2	3,289	NaN
D_2^*	3,329	NaN
D_2^q	3,302	NaN
D_2^{q*}	3,329	NaN
Dist_q ABySS	75,427	4,305
Dist_q Edena	68,326	4,382
Dist_q SPAdes	NaN	NaN
Dist_q SSAKE	82,604	4,575
Dist_q Velvet	88,156	4,276

Table 11.7: The runtime on the *E. coli* dataset. The assembly time (without the distance matrix calculation) on the same dataset is 24,980 s (ABySS), 17,514 s (Edena), 1021,184 s (SPAdes), 229,350 s (SSAKE), and 17,608 s (Velvet).

Chapter 12 Subsidiary Experiments

This chapter will show some experiments that thematically did not fit into the previous chapters nor are as important as those in the main experimental chapter (Chapter 11). Nevertheless, the experiments may illustrate some properties of the methods.

12.1 Dependence of $\mathsf{Dist}_{\mathsf{CC}}$ on Contig Quality

To further test the contig-contig distances, we generated idealized contigs for the influenza dataset. The artificial contigs are denoted as opt_θ. First, we started by randomly generating i.i.d. selected reads, similarly to Chapter 11. For each position, we calculated the per-base coverage and produced nine simulated assembly results opt_θ for $\theta = 10, 20, \ldots, 90$. To generate opt_θ, we chose the highest number c such that at least θ percent of nucleotides are covered by c or more reads. Those nucleotides with the high per-base coverage then formed individual contigs - the adjacent nucleotides were joined into contigs so that each contig could not be extended by a high-base coverage nucleotide.

Figure 12.1 shows the dependence of correlation on the percent of nucleotides in the simulated assembly, i.e., on parameter θ of opt_θ. For clarity, the figures do not plot all the methodological combinations. Figure 12.1 indicates that $\mathsf{Dist}_{\mathsf{CC}}(C_A, C_B)$ gives good estimates even if assembly identified only a fraction of the original sequence. The results are further improved when the reads are included in the calculations and distance $\mathrm{Dist}(T_A, T_B)$ is used instead. However, it needs to be said that when the contigs cover only 10 % to 20 % of the sequence, it is better to use $\mathsf{Dist}_{\mathsf{MESG}}$ instead as overlaps between those very fragmented and short contigs are more likely to be random than caused by the real overlaps in the alignment of the sequences. Also, no significant runtime improvement, in that case, can be expected, as shown in Figure 12.2.

12.2 Effect of Tries on the Monge-Elkan Distance Evaluation [243]

In this section, we briefly evaluate the effect of usage of tries for evaluation of (5.1). Our main evaluation criterion is runtime. We evaluated the algorithm for choices of read length $l \in \{8, 10, 12, \ldots, 30\}$ and for choices of read bag sizes $|R_A| = |R_B| \in \{100, 200, 300, \ldots, 2500\}$. The reads were sampled from DNA sequences of viruses wry ith accession AM712239 and AM712239 that were downloaded from the ENA repository [166]. We repeated each experiment 10 times and averaged the results for each choice of l and read bag size.

We compared the results of Algorithm 10.1 with the straightforward evaluation of (5.2) using the standard Wagner-Fischer dynamic programming algorithm. The time

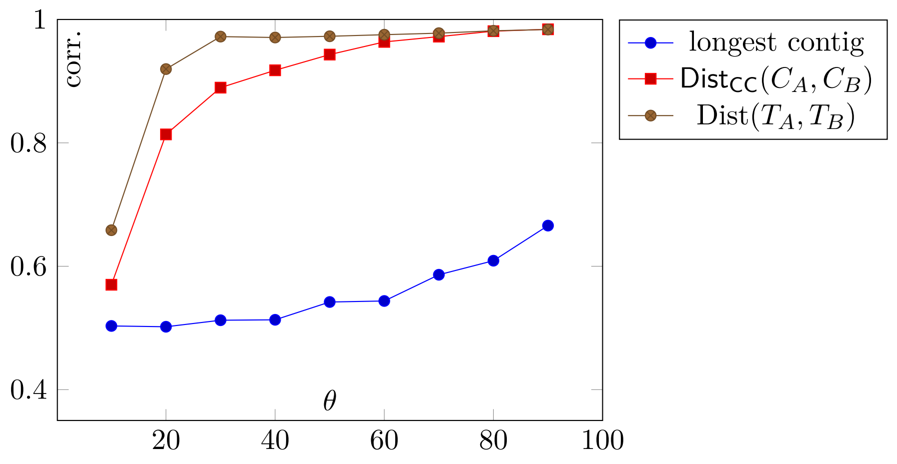


Figure 12.1: A plot of the average Pearson's correlation coefficient on θ parameter for simulated assembly opt_θ on the *influenza* dataset.

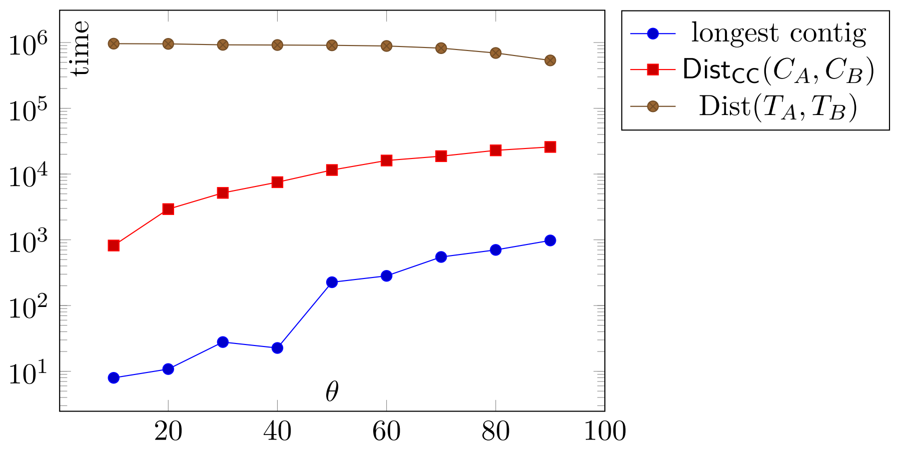


Figure 12.2: A plot of the average distance calculation time on θ parameter for simulated assembly opt_θ on the *influenza* dataset. Note that the Dist method is visualized without the embedding and sampling techniques.

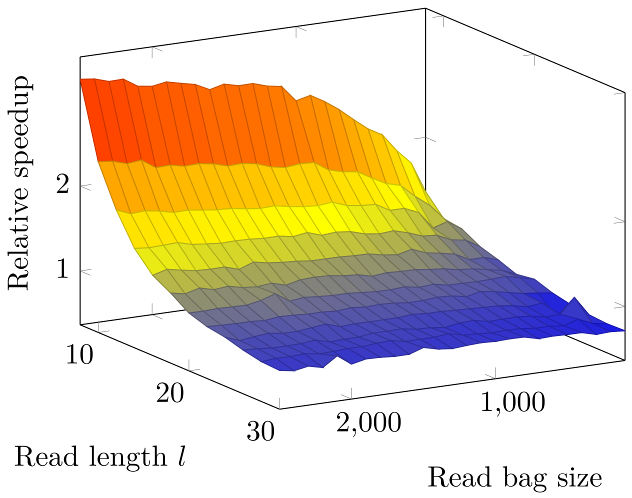


Figure 12.3: The relative speedup of Algorithm 10.1 on size of the read bags and read length l. The algorithm is compared to the direct evaluation of the Monge-Elkan distance with the Wagner-Fischer algorithm.

needed to build a trie is included in the runtime of Algorithm 10.1, because it is neglectable (approximately 0.1 %) compared to the time that is needed to calculate the distance in Formula (5.2).

The main experimental result is shown in Figure 12.3, which shows the dependency of relative speedup w.r.t. reference on read length l. From the figure, we see that for very short reads, Algorithm 10.1 is up to 3 times faster than the reference approach. However, for longer reads, the improvement is smaller than for shorter reads. We see that the overall speedup is higher for short reads and bigger read bags. This has a natural explanation in the structure of the tries. They can compress a shared prefix of the reads. Reads, unlike English words, are mostly random strings, although they are sampled from the same source. Therefore, the common prefixes are less common than in the case of dictionary search. Therefore, a trie can avoid redundant work on approximately $\log_{|\Sigma|} |R_A|$ nucleotides. For $|R_A| = 2500$ and $|\Sigma| = 4$ this number is approximately 6. Therefore, two reads are not likely to share more than 6 initial nucleotides. This shared prefix only saves a little work for longer reads, and a lower memory locality overtakes the savings.

12.3 Quality of Overlap Minimizing the Post-normalized Levenshtein Distance

In Algorithm 8.1, we presented an algorithm for finding the overlap of two contigs such that the post-normalized Levenshtein distance from (2.4) is minimized. We did not provide any guarantees for the quality of the result; we stated that the algorithm is only a heuristic that approximates the optimum. In Algorithm 8.2, we presented an algorithm that provides the exact result but is not fast enough.

In this section, we compare Algorithms 8.1 and 8.2 to verify that on the real data, the heuristic provides results that are usable in praxis. We selected contigs generated by the SPADES [210] assembly algorithm on the *hepatitis* dataset with coverage $c = 10$ and read length $l = 70$. From those contigs, we randomly selected one thousand pairs and calculated the value of (2.4) for both algorithms. We calculated the ratio to see how big is the relative error. The histogram of the results can be seen in Figure 12.4.

From the figure, we see that the large errors are relatively sparse. For 698 cases

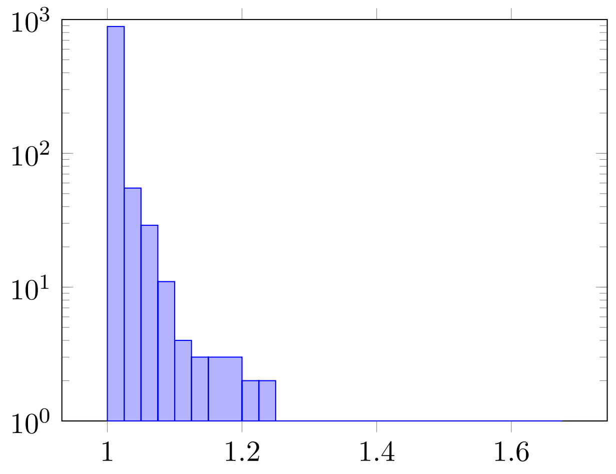


Figure 12.4: A histogram of the relative error of Algorithm 8.1 to its exact version in Algorithm 8.2. The plot shows the histogram of the relative error on 1000 sequences.

out of 1000, the heuristic in Algorithm 8.1 provided the same value of the optimized criterion as the exact algorithm in Algorithm 8.2.

Chapter 13

Discussion

As we have seen from the previous chapters, the proposed methods to compare read bags are somewhere in the middle between the alignment-free approaches and the alignment-based approaches. The methods do alignments of small subsequences of the original sequences in order to compare the original sequences.

The proposed methods are primarily applicable to viral or shorter bacteria genomes. We have shown the applicability of the methods on bacteria genomes (see the results on the *E. coli* dataset). The direct sequence alignment for genome comparison is more applicable for sequences thousands of nucleotides long, as for longer sequences, the larger genome events are more likely. For longer sequences, it is, therefore, more natural to count higher-order events as genome rearrangements [220]. The presented Monge-Elkan-based approaches are, similar to the alignment-free methods, less sensitive to such events. The applicability to viral sequences is more supported by the fact that viral genomes are fast evolving as their lifespan counts in days. This evolution rate can make reference alignment problematic. The fast evolution might be suppressed for some viruses, including the coronaviruses [325], but, for example, the influenza virus mutates 30 times faster than the recent covid-19 virus [189]. Therefore, the influenza virus is a good example of a target genetic sequence.

We have seen the applicability of the method on both similar and dissimilar sequences. The *influenza* dataset contained similar sequences; the *various* dataset contained dissimilar sequences. On those short genome sequences, the proposed method dominated the alignment-free approaches. This might come from the fact that for shorter sequences, the probability of shared k-mers is smaller.

The experiments show that the method requires coverage of only 2. Potential benefits, therefore, include a possible reduction of the wet lab sequencing, no need to use high coverages, or to sequence mate pairs. Figures 11.7 and 11.8 have also shown that the correlation is high for a read length of 10. This might be useful in some applications; for example, publication [196] justifies the need to use assembly-free, alignment-free, and reference-free tools by the MiSeq sequencing [229] of rapidly mutating RNA viruses. Nevertheless, Figures 11.7 and 11.8 show that the possible conditions when our method works are much broader. [240]

Our methods had two assumptions. The first one was that the read length l is the same for all reads. This can be justified by the fact that many sequencing technologies (including Illumina) read a single nucleotide in each iteration. As a result, all reads are sequenced with the same read length. The assumption that the coverage is equal for all samples is usually not met. Hence, in our publicly available implementation, this requirement is not enforced and can be replaced by either providing per read bag coverage or an estimate of the length of each genome. For our analyses, especially in the artificial datasets, we assumed that the reads are generated uniformly, which is usually not exactly true [308]. [240]

The proposed method is applicable not only on its own but might be used in combination with other methods. In metagenomics, researchers often do the sequencing of many organisms at once. For example, one milliliter of seawater contains around 10^9 viral particles [118]. Those particles are sequenced at once without knowing which read originated from which sequence. The resulting read sequences are then clustered based on criteria such as GC-content [121], read prefix-suffix overlaps, or k-mer counts. This clustering should identify individual organisms and is called binning [256]. Assuming that the binning was correct, our method is applicable to provide a phylogenetic tree as it might be difficult to select a correct reference for read mapping, especially if the sample is likely to contain many unknown or highly mutated viral particles.

Another natural usage might be in the supertree methods [62]. In more extensive phylogenetic studies, the resulting tree might contain hundreds or thousands of sequences and, at the same time, span over many taxonomic levels. In such a case, a different algorithm might be used to cluster higher-order taxa than the algorithm that clusters individual species or individual strands of the same species. In such a case, the approaches proposed in this paper are a possible choice for clustering of similar sequences.

Although we motivated the methods with unsupervised learning and hierarchical clustering, in particular, the distance estimates may be used in supervised learning. Classification using the k-NN [94] algorithm is a good example. The classification can then be used to identify taxons from their DNA sequences. Sometimes we are interested in which taxon a newly sequenced bacteria or virus belongs, and DNA sequence is the most straightforward way to classify the organism nowadays. Some studies use similar methods to do classification, for example, [297, 209, 310].

Other applications stem from the usage of sequence alignment. We might identify conserved regions by observing which reads had a low distance to their closest counterpart in Formula (5.1). If there is a high similarity between two parts of sequences, it indicates that in this region of DNA, the mutations are suppressed as they are likely to cause the death of the organism or turn out to cause evolutionary disadvantage. From the algorithmic perspective, we can claim that we do not only use global alignment to calculate the similarity of the sequences but local alignment as well when we select the closest reads in the other read bag.

Chapter 14

Towards RNA-Seq Similarities - Annotation of circRNAs

In this chapter, we will develop an algorithm to annotate circRNA molecules with annotation terms, and thus, we will provide a way to identify similarities between a different type of sequencing data — the RNA-seq data. The contents of this chapter will be mostly based on [242]. We will propose an algorithm, named circGPA (circRNA generating-polynomial annotator). To calculate the annotations, we exploit the interaction graph between circRNA-miRNA and miRNA-mRNA molecules. As we assume that the annotations of circRNAs are independent, we can process the individual circRNAs sequentially and restrict ourselves to a single circRNA molecule in our description.

The overall idea is that we will construct a statistic that is based on the interaction graph. Later, this statistic is further extended by some additional features. Moreover, we calculate the p-value of the statistic using dynamic programming and generating polynomials. The remainder of this chapter then contains experimental evaluation and comparison to the standard method of evaluating the p-value by Barnard's Monte-Carlo sampling [192].

14.1 Problem Definition [242]

Assume a fixed ordering on miRNA and mRNA molecules. Assume that the count of mRNAs (miRNAs) is $|m|$ ($|\mu|$). Formally, we can define the interaction graph between the selected circRNA and miRNAs using a vector $\mathbf{a}^{\mu,c} \in \{0,1\}^{|\mu|}$ where each field represents whether a particular miRNA interacts with the circRNA. Interactions between miRNAs and mRNAs are represented by an adjacency matrix $\mathbf{A}^{m,\mu} \in \{0,1\}^{|m| \times |\mu|}$ where each row is a vector indicating which miRNAs interact with a particular mRNA.[1] We assume that the graph edges are directed only from circRNA to miRNA and from miRNA to mRNA so that a directed path cannot connect two molecules of the same type. A simple network is shown in Figure 14.1.

In our notation, an annotation term will be defined by a set of mRNAs and miRNAs it annotates. The membership of mRNAs (miRNAs respectively) is formalized using binary vectors $\mathbf{g}^m \in \{0,1\}^{|m|}$ ($\mathbf{g}^\mu \in \{0,1\}^{|\mu|}$ respectively). As a shorthand notation, we will use the symbol g to denote the tuple of the latter, i.e., $g = (\mathbf{g}^m, \mathbf{g}^\mu)$. Having these

[1]We define $\mathbf{a}^{\mu,c}$ as a binary vector, and $\mathbf{A}^{m,\mu}$ as a binary matrix. However, the approach can be easily generalized to the situation when $\mathbf{a}^{\mu,c}$ and $\mathbf{A}^{m,\mu}$ contain natural numbers which might capture, for example, the strength of the interactions or, alternatively situations when a miRNA has two binding spots on a circRNA. The fields higher than 1 can be represented as parallel edges in the interaction graph, which becomes a multigraph in this case.

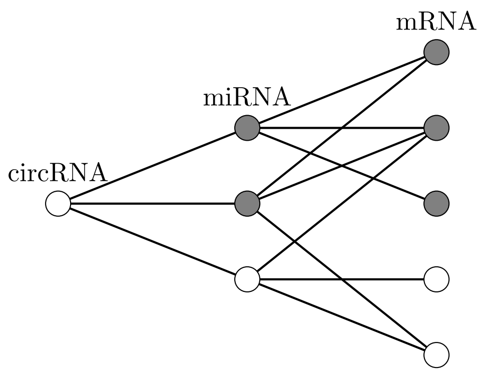


Figure 14.1: An example of a network. The grey nodes are part of the annotation term. The circRNA of interest interacts with all three miRNAs, out of which two are annotated with the term of interest. There are five mRNAs, three of them annotated with the term. In the graph, we might find three paths from the circRNA to a miRNA and nine paths from the circRNA to a mRNA. Out of those, $2 + 6$ terminate in an annotated miRNA/mRNA, resulting in $s(c, g) = 8$.

definitions on hand, we can define the problem to be solved in this chapter.

Definition 14.1 (CircRNA annotation problem) *For a circular RNA, let* $\mathbf{A}^{m,\mu}$, $\mathbf{a}^{\mu,c}$ *be its interaction graph. Decide whether the circRNA should be annotated with a term* $g = (\mathbf{g}^m, \mathbf{g}^\mu)$.

14.2 Proposed Statistic [242]

To solve the problem, we will develop a simple yet powerful statistic to annotate a circRNA. The concept is based on the ”guilt by association” principle [212, 175]. The circRNA should be annotated with a term if and only if this molecule frequently interacts with miRNAs (and through them indirectly with mRNAs) annotated with the term. We will capture this frequency in statistic s. This statistic will quantify the size of the neighborhood of the circRNA that is annotated with the term. As the complete tripartite graph is only a unification of two bipartite graphs and remains fixed for the circRNA, we might calculate this number precisely:

$$s(c, g) = (\mathbf{a}^{\mu,c})^T \mathbf{g}^\mu + (\mathbf{A}^{m,\mu}\mathbf{a}^{\mu,c})^T \mathbf{g}^m. \tag{14.1}$$

The first addend shows how many paths of length one end in a miRNA annotated with the term. The term $\mathbf{A}^{m,\mu}\mathbf{a}^{\mu,c}$ shows how many paths go from c to each mRNA. The second addend in Formula (14.1) calculates how many paths of length two terminate in an mRNA that is annotated with the term.

However, the frequency represented by s is hard to explain without knowing the entire neighborhood. Larger neighborhoods, as well as more abundant gene terms, tend to generate a larger frequency. The importance of the term could better be captured by a relative frequency. Assume that we start a random walk in circRNA c. We might calculate the probability that this random walk ends in an RNA annotated with the term g. For a fixed circRNA, the size of the neighborhood is fixed. Therefore, the aforementioned probability is equal to $s(c, g)$ but for a normalization factor.

We will continue to use the number of paths represented in $s(c, g)$, knowing that they are only linearly scaled, preserving the ordering of the above-mentioned probabilities.

To increase interpretability, we will further develop a normalized s as well as the p-value for the statistic so that standard statistical reasoning is applicable. In this chapter, we will avoid random Monte-Carlo sampling (used among others in [69, 86, 105, 161, 321]) and simulating the random walk and claim its low efficiency in our setting for the p-value calculation.

14.3 Normalization [242]

As the value of statistic s grows by definition with the size of the annotation term and the size of the interaction graph (it gives the number of distinct paths to mRNAs and miRNAs annotated with the term), we present the user with a more explainable output. We normalize the statistic (14.1) by its expected value

$$\mathsf{E}\, s(c,g) = \frac{\|\mathbf{g}^{\mu}\|_1}{|\mu|}(\mathbf{a}^{\mu,c})^T\mathbf{1} + \frac{\|\mathbf{g}^{m}\|_1}{|m|}(\mathbf{A}^{m,\mu}\mathbf{a}^{\mu,c})^T\mathbf{1} \tag{14.2}$$

where $\mathbf{1} = (1, 1, \ldots, 1)^T$ is the vector of ones and $\|\cdot\|_1$ denotes the L1-norm. The expected value gives the expected number of random walks that end in an RNA annotated with the term if the annotations were assigned randomly. The user is then presented with the ratio of the statistic and its expected value

$$\frac{s(c,g)}{\mathsf{E}\, s(c,g)}. \tag{14.3}$$

This ratio represents the normalized statistic. The value above 1 then stands for terms that tend to interact with the circRNA under observation more than expected as can be seen in Table 14.1.

14.4 Influence of Individual RNAs [242]

Once circGPA predicts that a circRNA should be annotated with a term, users might be interested in which miRNAs and mRNAs back up this annotation. In other words, knowledge of which RNAs connect the circRNA to the annotation term is important. Fortunately, it is possible to split $s(c,g)$ among individual molecules. A natural method of explaining how much the RNA adds to the statistic is to remove this RNA with all its incoming and outcoming edges from the graph. On such a modified graph, we recalculate the score and calculate the difference in the score value. We can calculate this difference for all miRNAs and mRNAs at once using linear algebra. We denote the vector of those differences $\boldsymbol{\Delta}^m$ ($\boldsymbol{\Delta}^\mu$) for mRNAs (miRNAs). For a vector $\mathbf{v}$, let $\operatorname{diag}(\mathbf{v})$ denote a diagonal matrix with elements of $\mathbf{v}$ on its diagonal. Then we derive that

$$\boldsymbol{\Delta}^{\mu} = \operatorname{diag}(\mathbf{a}^{\mu,c})\left(\mathbf{g}^{\mu} + (\mathbf{A}^{m,\mu})^T\mathbf{g}^{m}\right), \tag{14.4}$$

$$\boldsymbol{\Delta}^{m} = \operatorname{diag}(\mathbf{A}^{m,\mu}\mathbf{a}^{\mu,c})\mathbf{g}^{m}. \tag{14.5}$$

One can notice that the L1 norm of $\boldsymbol{\Delta}^\mu$ is equal to $s(c,g)$. We use values $\boldsymbol{\Delta}^\mu, \boldsymbol{\Delta}^m$ to sort mi/mRNAs in a report that shows the influence of individual RNAs. An example output will be seen in Section 14.11.

14.5 p-value Calculation [242]

To understand and compare the values of statistic s among different circRNAs and annotation terms, we need to calculate its p-value. The p-value cannot stem solely from

s itself as other circular RNAs have a different number of connections to the remaining RNAs. In addition, more frequent annotation terms will reach higher scores. Formally, statistic $s(c, g)$ is an outcome of the statistical test, whose null hypothesis is that *the given c, g pair is not related.* In other words, the null hypothesis is that circRNA c has no preference in interactions with miRNAs (or mediated interactions with mRNAs) annotated with term g. The alternative hypothesis states that *c should be annotated with g* as g is overrepresented in the neighborhood of c.

The p-value, in our case, represents the probability that a random annotation term of the same size in the same interaction graph reaches the same statistic s or higher. The literal implementation of the null distribution simulation would thus be empirical random sampling with replacement [188]. In our case, this Monte-Carlo approach would be based on enumerating the random subsets of m/miRNAs of the same size as the evaluated annotation term and calculating the statistic value based on Formula (14.1).

This chapter proposes an exact approach that does not depend on random trials but uses generating polynomials instead to compute the p-value. We should first reformulate the problem so that we can easily describe its mathematical solution. Denote $\|\mathbf{g}^\mu\|_1$ the number of miRNA molecules annotated by the term. Formally, $\|\mathbf{g}^\mu\|_1$ is the L1-norm of the $\mathbf{g}^\mu$ vector. Define $\|\mathbf{g}^m\|_1$ similarly. For each miRNA, there is a fixed number denoting its weight in the statistic (14.1). This weight is 1 if and only if the circRNA of interest is connected to the miRNA, zero otherwise. The weight is stored in the respective field of $\mathbf{a}^{\mu,c}$. Out of all miRNAs, we select $\|\mathbf{g}^\mu\|_1$. For mRNA, the weight can be seen in the respective field of $\mathbf{A}^{m,\mu}\mathbf{a}^{\mu,c}$. Out of all interacting mRNAs, $\|\mathbf{g}^m\|_1$ mRNAs are selected.

To calculate the p-value, the molecules of mRNA and miRNA are selected randomly given the weights and the fact that $\|\mathbf{g}^\mu\|_1$ and $\|\mathbf{g}^m\|_1$ need to be preserved. For now, we consider only miRNAs. Imagine a bag full of balls with numbers written on them. Each number is a field in $\mathbf{a}^{\mu,c}$ (one field equals one ball). Now we randomly select $\|\mathbf{g}^\mu\|_1$ balls from the bag and sum the numbers written on them. By repeating this procedure many times, we get the null distribution for the first part of the statistic (14.1). If we include a second bag with numbers taken from $\mathbf{A}^{m,\mu}\mathbf{a}^{\mu,c}$, we get the null distribution for the whole statistic.

Having built an informal intuition, we can proceed to introduce the generating polynomials by which we denote a polynomial which is a multiple of the well-established probability-generating functions [88]. Consider an mRNA that is connected by five paths to the circRNA. The weight of this mRNA is 5. In a random annotation term, this mRNA is either included or not. This gives two possibilities. We can formulate the generating polynomial for this mRNA as

$$1 + x^5y^1. \tag{14.6}$$

The variable x keeps track of weights, y keeps track of the number of selected mRNAs. Having a simple graph with only one mRNA, we have two options for building a random mRNA set: either we use zero mRNAs, and the sum of weights is zero (the term 1, which equals x^0y^0), or we use one, and the sum is 5 (the term x^5y^1). If we also consider a new mRNA with weight 3, the resulting polynomial that represents the extended graph is

$$(1 + x^5y^1) \cdot (1 + x^3y^1) = 1 + x^3y^1 + x^5y^1 + x^8y^2. \tag{14.7}$$

We immediately see that if we select no mRNAs, we can only get the sum of weights 0; by selecting one, the weights will be either 3 or 5, and by selecting two, the sum of the weights will be eight. The coefficients by terms with y^1 show a single possibility of getting a weight of three or five. Another helpful view on the formula above might be as on a dynamic programming algorithm in a 2D array where the power of x denotes a row, the power of y denotes a column, and the coefficient is the number at the particular position of the table. Now, we can define the generating polynomial for a weight vector.

Definition 14.2 (Generating polynomial) *Let $\mathbf{w}$ be a vector of weights (of mRNA or miRNA). Then the generating polynomial is*

$$\mathrm{genpoly}_{\mathbf{w}}(x,y) = \prod_{w \in \mathbf{w}} (1 + x^w y). \tag{14.8}$$

Next, we define an operator that restricts the polynomial only on a selected power of one or more variables. We will denote the operator $\mid x^n$ and use it to denote only terms that contain x^n. For example, for the polynomial (14.7), operator $\mid y^1$ will return $x^3 + x^5$. The following theorem allows us to calculate the null distribution of the statistic (14.1).

Theorem 14.3 *Consider statistic s for a fixed circRNA c, interaction graph $\mathbf{a}^{\mu,c}$, $\mathbf{A}^{m,\mu}$ and annotation term sizes $\|\mathbf{g}^{\mu}\|_1$, $\|\mathbf{g}^{m}\|_1$. Then coefficients of the polynomial*

$$\left(\mathrm{genpoly}_{\mathbf{a}^{\mu,c}}(x,y) \mid y^{\|\mathbf{g}^{\mu}\|_1}\right)\left(\mathrm{genpoly}_{\mathbf{A}^{m,\mu}\mathbf{a}^{\mu,c}}(x,y) \mid y^{\|\mathbf{g}^{m}\|_1}\right) \tag{14.9}$$

are the null distribution of statistic s up to a normalization factor.

Proof From what precedes, it can be seen that the first multiplicand coefficients are the number of ways to reach a particular value of the miRNA part of the statistic (14.1) by selecting a particular number of miRNAs. The restriction to the $y^{\|\mathbf{g}^{\mu}\|_1}$ ensures that the number of miRNAs in the annotation term is preserved. The same holds for the second multiplicand and mRNAs.

After multiplying the polynomials, the polynomial coefficients will hold the number of unique ways the value of the statistic can be achieved. The normalization to 1 then finishes the calculation of the null distribution. ■

Once the null distribution is calculated, the p-value is then obtained by a standard approach in which we sum probabilities of all statistic values greater than $s(c, g)$.

14.6 Computational Complexity [242]

If we focus on the computational complexity of circGPA, most of the work is done in the `Generating-Polynomial` function. The two inner loops depend on variables $maxx$ and $maxy$, where $maxx$ is, in the worst case, linearly dependent on the L1-norm of vector $\mathbf{w}$; $maxy$ is equal to the number of RNAs annotated with the terms. Their product, therefore, is linearly dependent on the product of $\|\mathbf{w}\|_1$ times the size of the annotation term. The two outer loops in function `Generating-Polynomial` do at most n operations for each unique non-zero entry in the vector of weights of count n. Sum of the fields of weight vector $\mathbf{w}$ is, therefore, the same as the number of evaluations of the two outer loops. The computational complexity of the two outer loops is in the worst case equal to $\|\mathbf{w}\|_1$. We may conclude that one call to the `Generating-Polynomial` function is in $\mathcal{O}(\|\mathbf{w}\|^2 \cdot maxy)$. Other terms in function `AnnotateCircRNA` are asymptotically smaller than the runtime of the `Generating-Polynomial` function. The overall runtime is, therefore, in

$$\mathcal{O}\left((\|\mathbf{a}^{\mu,c}\|_1)^2\|\mathbf{g}^{\mu}\|_1 + (\|\mathbf{A}^{m,\mu}\mathbf{a}^{\mu,c}\|_1)^2\|\mathbf{g}^{m}\|_1\right). \tag{14.10}$$

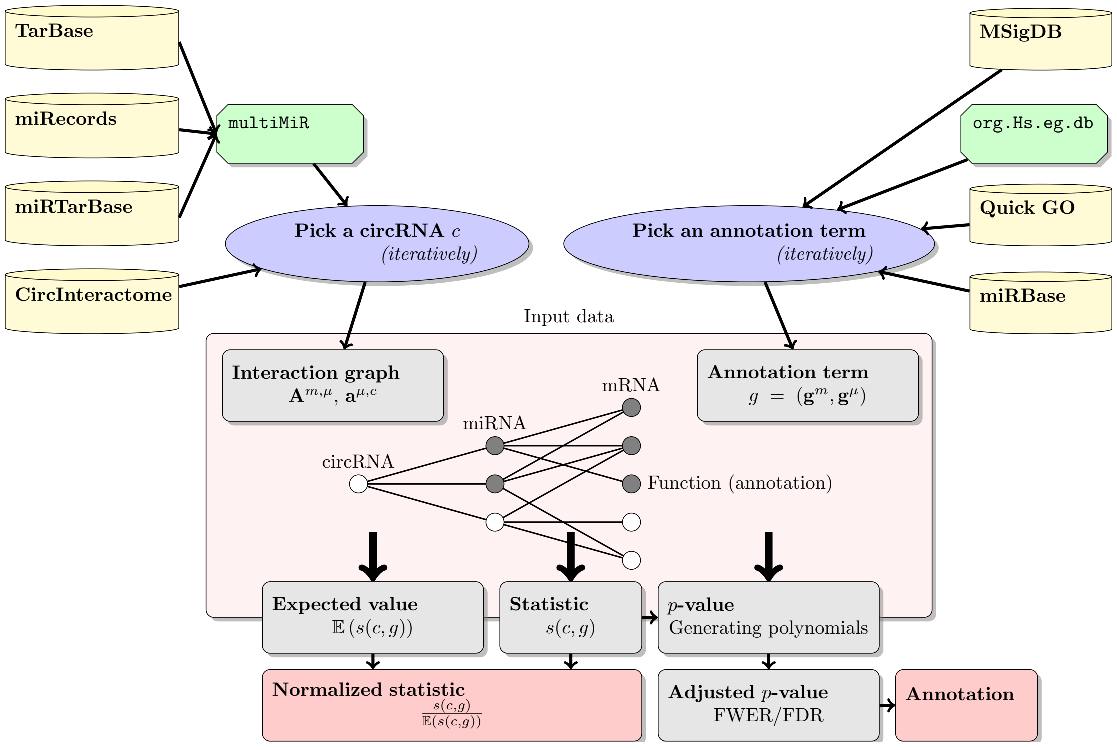

TarBase
miRecords
miRTarBase
CircInteractome
multiMiR
MSigDB
org.Hs.eg.db
Quick GO
miRBase
Pick a circRNA c (iteratively)
Pick an annotation term (iteratively)
Input data
Interaction graph $\mathbf{A}^{m,\mu}$, $\mathbf{a}^{\mu,c}$
Annotation term $g = (\mathbf{g}^m, \mathbf{g}^\mu)$
mRNA
miRNA
circRNA
Function (annotation)
Expected value $\mathbb{E}(s(c,g))$
Statistic $s(c,g)$
p-value Generating polynomials
Normalized statistic $\frac{s(c,g)}{\mathbb{E}(s(c,g))}$
Adjusted p-value FWER/FDR
Annotation

Figure 14.2: An illustration of the whole pipeline. The input graph is used for multiple annotation terms to calculate the statistic and its p-value. Later, the p-values are adjusted and used for annotation.

14.7 Referential Approach

[242]

A standard approach for p-value calculation would be to enumerate subsets of miRNAs/mRNAs as random annotation terms. The size of the term is preserved, and we count how many times the score is higher than $s(c, g)$. This sampling Monte-Carlo approach then allows estimation of the p-value using the biased estimator $\frac{r+1}{n+1}$, where r is the number of trials with a high enough score and n is the number of all trials [223].

14.8 Implementation

[242]

Using a naive implementation, we need at most $2^{|m|} \cdot 2^{|\mu|}$ multiplications to evaluate polynomial (14.9). This number is the theoretically possible maximum number of terms in polynomial (14.7). The real number is, however, much smaller. As polynomial exponents repeat, the bound can be tightened. The power of x goes from 0 to $\|\mathbf{a}^{\mu,c}\|_1$ in the case of miRNAs, and from 0 to $\|\mathbf{A}^{m,\mu}\mathbf{a}^{\mu,c}\|_1$ in the case of mRNAs. The y variable goes from 0 to $|\mu|$ (0 to $|m|$ in case of mRNA); however, relevant fields are only up to $\|\mathbf{g}^\mu\|_1$ ($\|\mathbf{g}^m\|_1$). The x variable can be trimmed similarly using the fact that only $\|\mathbf{g}^\mu\|_1$ (or $\|\mathbf{g}^m\|_1$) highest terms of $\mathbf{a}^{\mu,c}$ (or $\mathbf{A}^{m,\mu}\mathbf{a}^{\mu,c}$) may be used.

If the weight of a mi/mRNA occurs more than once, we can exploit the binomial expansion instead of term-by-term multiplication in Equation (14.8). The multiplication could be implemented using the dynamic programming approach and pointers. A similar approach was used for the fast p-value calculation of the unweighted GSEA [149]. Details can be seen in Algorithm 14.1. The whole pipeline is illustrated in Figure 14.2.

```
function AnnotateCircRNA($c$, $\mathbf{a}^{\mu,c}$, $\mathbf{A}^{m,\mu}$, $\mathbf{g}^{\mu}$, $\mathbf{g}^{m}$)
    $s(c,g) \leftarrow (\mathbf{a}^{\mu,c})^T\mathbf{g}^{\mu} + (\mathbf{A}^{m,\mu}\mathbf{a}^{\mu,c})^T\mathbf{g}^{m}$        ▷ Calculate the statistic
    $\mathsf{E}\, s(c,g) = \frac{\|\mathbf{g}^{\mu}\|_1}{|\mu|}(\mathbf{a}^{\mu,c})^T\mathbf{1} + \frac{\|\mathbf{g}^{m}\|_1}{|m|}(\mathbf{A}^{m,\mu}\mathbf{a}^{\mu,c})^T\mathbf{1}$
                                                    ▷ The expected value of the statistic

    $miRNAPoly \leftarrow$ Generating-Polynomial($\mathbf{a}^{\mu,c}$, $\|\mathbf{g}^{\mu}\|_1$)
    $mRNAPoly \leftarrow$ Generating-Polynomial($\mathbf{A}^{m,\mu}\mathbf{a}^{\mu,c}$, $\|\mathbf{g}^{m}\|_1$)
    $null \leftarrow$ Coefficients($miRNAPoly \cdot mRNAPoly$)
    $pval \leftarrow$ Sum($null[s(c,g):]$)/Sum($null$)
      ▷ $p$-value is the probability of reaching a higher value than $s(c,g)$ for a random term
    return $\frac{s(c,g)}{\mathsf{E}\, s(c,g)}$, $pval$
end function

function Generating-Polynomial($\mathbf{w}$, $maxy$)
                                    ▷ $maxy$ represents how many weights are selected
    $maxx \leftarrow$ Sum($maxy$ highest terms in $\mathbf{w}$)
                          ▷ Since we select only $maxy$ RNAs, higher powers of $x$ will be zero
    $curr, next \leftarrow$ 2D-arrays of 0 indexed from 0 to $maxx$ and from 0 to $maxy$
    $curr[0,0] = next[0,0] = 1$
                               ▷ Initialization, 1 way of obtaining sum 0 using 0 molecules

    for all unique $w$ in $\mathbf{w}$ do
        $n \leftarrow$ count of $w$ in $\mathbf{w}$
                                    ▷ We will use binomial expansion of $(1 + x^w y^1)^n$
        $maxPow \leftarrow$ Min($n, maxy$)
                               ▷ Only relevant powers of $y$ in expansion of $(1 + x^w y^1)^n$
        for $pow = 1...maxPow$ (incl.) do                          ▷ Power of $x^w y^1$
            for $y = pow...maxy$ (incl.) do
                for $x = w \cdot pow...maxx$ (incl.) do
                    $next[x,y] \leftarrow next[x,y] + curr[y - pow, x - w \cdot pow] \cdot \binom{n}{pow}$
                              ▷ Multiply the $curr$ poly. by $\binom{n}{pow}(x^w y^1)^{pow}$; add to $next$
                end for
            end for
        end for
        copy values from $next$ to $curr$                          ▷ This includes $\cdot x^0 y^0$
    end for
    return Polynomial in $x$ given by the last row, i.e., $curr[, maxy]$
end function
```

Algorithm 14.1: The circGPA algorithm.

14.9 Input Data and Used Libraries [242]

Our implementation combines R code and C++ code for the critical p-value calculation. To connect the C++ code and R code, we use the `Rcpp` package [78].

To construct the graph, we exploit several databases and R packages. The circRNA-miRNA interactions are downloaded from the CircInteractome [76] database that uses the TargetScan [170] interaction prediction algorithm. The miRNA-mRNA interactions are downloaded from the TarBase [145], miRecords [309], and miRTarBase [126] databases via the `multiMiR` package [235]. In the case of miRNA-mRNA interactions, we used verified interactions only.

The GO annotation for the miRNAs is downloaded using the miRBase [108] and DNA Quick GO [24] databases. Annotation of mRNAs is done via the `org.Hs.eg.db` R package. The annotation terms are obtained from the MSigDB database [176] C5 category using the `msigdbr` R package.

Other R packages used include `miRBaseConverter`, `GO.db`, `biomaRt`,`stringr`, `httr`, `openxlsx`, `geometry`, `tictoc`, `ggnet`, `network` and `polynom`.

The graph constructed in the presented way can annotate 3,009 circRNAs. There are 1,761 miRNAs connected by 81,391 edges. This means that one circRNA interacts with 29 miRNAs on average. One miRNA interacts with 50 circRNAs on average. The circRNA that interacts most with other molecules is `hsa_circ_0000005` with 307 interactions. The miRNA with the most frequent interactions is `hsa-miR-942`, with 799.

The graph contains 19,375 mRNAs with 465,741 known interactions with miRNAs. Therefore, one mRNA interacts with 24 miRNAs on average, while one miRNA interacts with 264 mRNAs on average. The most frequent miRNA is `hsa-miR-1-3p` with 7,491 interactions, the most frequent mRNA is `NUFIP2` with 331 interactions.

We work with 10,189 annotation terms. The average size of those is 82 mRNA or miRNA molecules. If we exclude annotation terms which are too narrow or too broad (see Section 14.11 for details), we end up with 7,075 annotation terms with 89 molecules on average. One RNA is annotated with 43 terms on average.

14.10 All in One Example [242]

Consider the interaction network depicted in Figure 14.1. In this figure, we have a circRNA of interest connected with 3 miRNAs that connect to 5 mRNAs. The edges in the graph can then be described by a vector and a matrix.

$$\mathbf{a}^{\mu,c} = (1,1,1)^T, \qquad \mathbf{A}^{m,\mu} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}.$$

The annotation term contains 2 miRNAs and 3 mRNAs. It is formalized as

$$\mathbf{g}^{\mu} = (1,1,0)^T, \qquad \mathbf{g}^{m} = (1,1,1,0,0)^T.$$

The weights of the miRNAs and mRNAs are

$$\mathbf{a}^{\mu,c} = (1,1,1)^T, \qquad \mathbf{A}^{m,\mu}\mathbf{a}^{\mu,c} = (2,3,1,1,2)^T.$$

Which gives a statistic of

$$s(c,g) = 2 + 6 = 8.$$

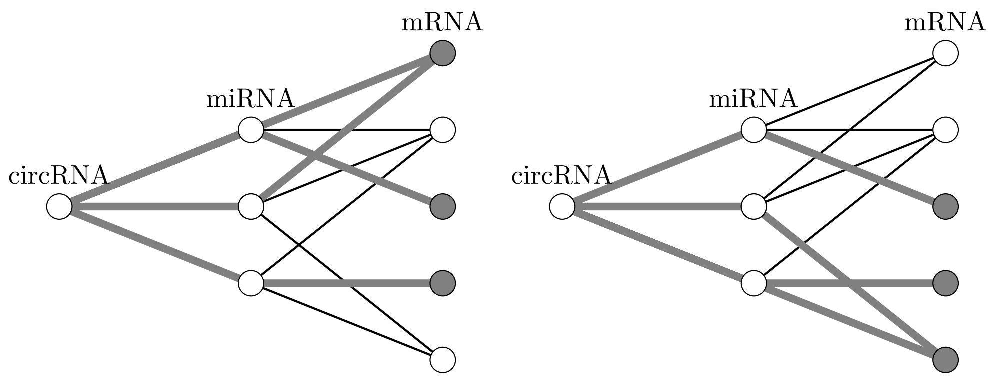


Figure 14.3: The generating polynomial of the mRNAs in this graph is equal to $2x^4 + 3x^5 + 4x^6 + x^7$. We see that there are two ways to annotate 3 mRNAs, so that there are 4 paths that end in an annotated mRNA. These two cases are illustrated in the figure, the paths are marked by bold gray.

The first addend is for miRNAs, the second for mRNAs. The expected value of the statistic is

$$\mathsf{E}\, s(c,g) = \frac{2}{3}(1+1+1) + \frac{3}{5}(2+3+1+1+2) = 7.4.$$

The normalized statistic is, therefore, $\frac{8}{7.4} \cong 1.08$. The generating polynomial for miRNAs is

$$(1 + x^1 y)^3 = 1 + 3xy + 3x^2y^2 + x^3y^3.$$

As we have two miRNAs that are annotated with the terms, we immediately see that there are three options for achieving two paths from the circRNA that end in an annotated miRNA (see term $3x^2y^2$ – variable y stands for the number of miRNAs, variable x stands for the number of paths). For mRNA, the generating polynomial is

$$(1 + x^1 y)^2(1 + x^2 y)^2(1 + x^3 y) = \cdots + (2x^4 + 3x^5 + 4x^6 + x^7)y^3 + \cdots.$$

The relevant terms contain the variable y to the power of three – the term annotates three mRNAs. And the corresponding terms show that there are two options for selecting mRNAs such that there are 4 paths that go from the circRNA to an annotated mRNA. Those two options are illustrated in Figure 14.3. Similarly, we can see that there are four options so that six paths end in an annotated mRNA and so on.

The generating polynomial for the whole statistic is, therefore,

$$3x^2(2x^4 + 3x^5 + 4x^6 + x^7) = 6x^6 + 9x^7 + 12x^8 + 3x^9.$$

From the polynomial, we see that there are 12 ways to obtain a statistic value equal to 8. One of these is the situation depicted in Figure 14.1 and solved in this example. The p-value is equal to the ratio of the number of combinations that reach a statistic value greater or equal to 8 to the number of all possible combinations. These can be seen from the polynomial or calculated as $\binom{3}{2}\binom{5}{3} = 30$. Hence, the p-value is $\frac{12+3}{6+9+12+3}$.

We can also see that the expected value is the same if calculated from the null distribution. There are six ways to get statistic equal to six, nine ways to get statistic equal to seven and so forth, i.e., the expected value is

$$\frac{6 \cdot 6 + 9 \cdot 7 + 12 \cdot 8 + 3 \cdot 9}{6 + 9 + 12 + 3} = \frac{222}{30} = 7.4.$$

14.11 Results

We ran circGPA on a graph constructed on the human genome as explained in Section 14.8. For presentation purposes, we filtered the annotation terms based on their sizes. We excluded annotation terms which are too broad or too narrow. The narrow terms are difficult to evaluate statistically, the general terms suffer from low interestingness to domain experts. Reimand et al. [230] argue that: *. . . we often recommend excluding pathway GO terms with* $< 10 - 15$ *genes and* $> 200 - 500$ *genes, although upper bounds of* $200 - 2{,}000$ *genes can be found in the literature.* We decided to stick with the bounds researched by the authors and exclude by default terms smaller than 10 genes and larger than 1,000 genes.

14.11.1 Outputs of circGPA

For each circular RNA, circGPA generates a table with the statistic (14.1), the normalized statistic (as described in Section 14.3), and the p-values. Since we deal with many annotation terms in parallel, we have to adjust p-values for multiple comparison. We provide both FWER (Bonferroni [77]) and FDR (Holm [20]) adjusted p-values. The runtime of the p-value calculation is measured. An example output is shown in Table 14.1.

For a visual presentation of the provided results, circGPA is able to generate an output in a form that can be processed by the EnrichmentMap plugin [197] of the Cytoscape program [260]. This tool visualizes multiple annotation terms found relevant for a single circRNA in a graph. The vertices correspond to the terms; their size corresponds to the number of genes in the term and the color is calculated from the p-value as in a heatmap. The edges are constructed so that their width represents the Jaccard index of the connected annotation terms – a wider line means a bigger overlap between the terms. An example of a produced output is presented in Figure 14.4. Therefore, the graph for a circRNA shows predicted annotations in the context of the other annotation terms. As a result, the user is presented with information about the term-term overlap and clustering of the predicted annotation. Figure 14.5 shows an example of the circGPA report in which miRNAs and mRNAs back up annotation.

To further test our method of p-value calculation, we implemented the usual sampling approach mentioned in Section 14.5. circGPA is deterministic and guarantees the exact p-values. The sampling approach is burdened with a random noise caused by the randomness in the selection of the subsets. Therefore, the p-values are not the same. At the very beginning, we thus configured the stochastic algorithm to approach the exact p-values. We worked with all the annotation terms relevant to `hsa_circ_0000228`. We found out that p-values closely match for 10^6 and more random trials. We follow recommendations of [223] and estimated the p-values as $\frac{r+1}{n+1}$, where r is the number of positive Monte-Carlo trials out of $n = 10^6$. The Spearman's ρ for the two p-value vectors were equal to 0.99991. We also calculated the relative difference with a mean equal to 0.015 (i.e., on average, the p-values differ by less than 2 %), standard deviation equal to 0.041, the maximum deviation equal to 0.82, and median equal to 0.006. If we eliminate the first 20 annotation terms with a p-value $1.6 \cdot 10^{-5}$ and smaller, the mean of the relative p-value difference is 0.014, standard deviation 0.027, maximum deviation 0.372, and median 0.006.

14.11.2 Runtime

Then, we compared the circGPA runtime with the sampling approach using 10^6 trials. The sampling was configured to allow for Monte-Carlo p-values to approach the exact

Set id	**Size**	$s(c,g)$	$\frac{s(c,g)}{\mathsf{E}\, s(c,g)}$	time (s)	p**-value**	**Bonferroni**	**FDR**
HP_SOFT_TISSUE_SARCOMA	115	38	3.61	0.04	1.40E-08	1.40E-04	1.39E-04
GOMF_MRNA_BINDING	287	64	2.44	0.16	2.78E-08	2.78E-04	1.39E-04
HP_GENITAL_NEOPLASM	142	41	3.16	0.05	5.78E-08	5.78E-04	1.93E-04
HP_SARCOMA	165	44	2.91	0.06	1.03E-07	1.03E-03	2.57E-04
HP_NEOPLASM_BY_HISTOLOGY	320	66	2.25	0.19	1.60E-07	1.60E-03	3.21E-04
HP_THIN_VERMILION_BORDER	329	66	2.19	0.20	3.48E-07	3.48E-03	5.80E-04
HP_THIN_UPPER_LIP_VERMILION	234	52	2.43	0.11	5.82E-07	5.82E-03	8.32E-04
GOMF_UBIQUITIN_LIKE_PROTEIN_LIGASE_BINDING	309	62	2.19	0.18	7.74E-07	7.75E-03	9.69E-04
HP_URINARY_TRACT_NEOPLASM	133	36	2.96	0.04	1.10E-06	1.10E-02	1.22E-03
GOBP_REGULATION_OF_MRNA_METABOLIC_PROCESS	334	64	2.09	0.22	1.75E-06	1.76E-02	1.63E-03

Table 14.1: A sample circGPA output. The table shows ten annotation terms with the smallest p-value that were obtained by annotating circRNA `hsa_circ_0000228`.

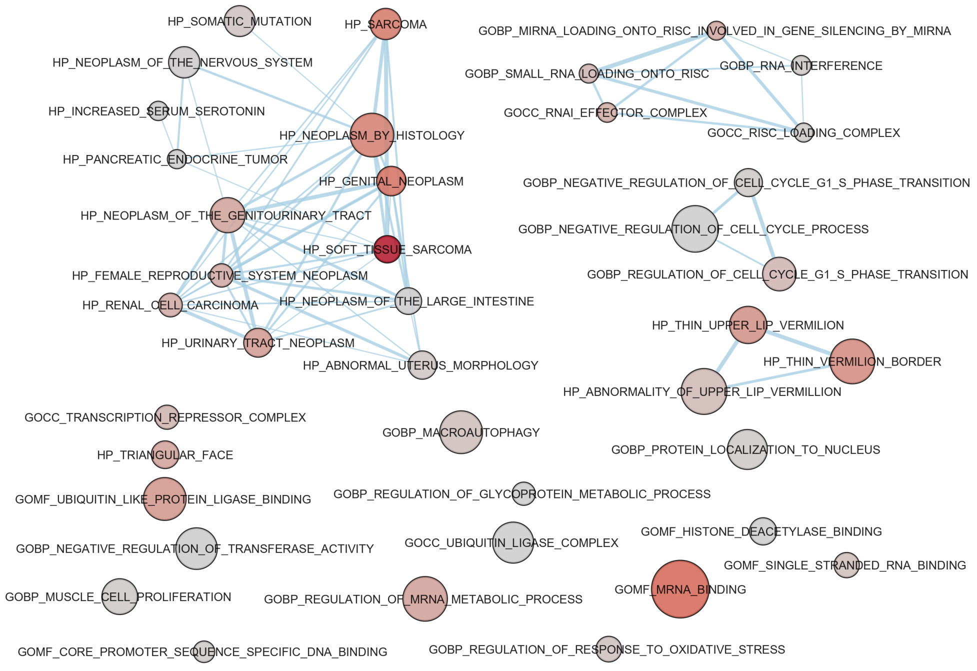

HP_SOMATIC_MUTATION
HP_SARCOMA
HP_NEOPLASM_OF_THE_NERVOUS_SYSTEM
HP_INCREASED_SERUM_SEROTONIN
HP_NEOPLASM_BY_HISTOLOGY
HP_PANCREATIC_ENDOCRINE_TUMOR
HP_GENITAL_NEOPLASM
HP_NEOPLASM_OF_THE_GENITOURINARY_TRACT
HP_SOFT_TISSUE_SARCOMA
HP_FEMALE_REPRODUCTIVE_SYSTEM_NEOPLASM
HP_RENAL_CELL_CARCINOMA
HP_NEOPLASM_OF_THE_LARGE_INTESTINE
HP_URINARY_TRACT_NEOPLASM
HP_ABNORMAL_UTERUS_MORPHOLOGY
GOBP_MIRNA_LOADING_ONTO_RISC_INVOLVED_IN_GENE_SILENCING_BY_MIRNA
GOBP_SMALL_RNA_LOADING_ONTO_RISC
GOBP_RNA_INTERFERENCE
GOCC_RNAI_EFFECTOR_COMPLEX
GOCC_RISC_LOADING_COMPLEX
GOBP_NEGATIVE_REGULATION_OF_CELL_CYCLE_G1_S_PHASE_TRANSITION
GOBP_NEGATIVE_REGULATION_OF_CELL_CYCLE_PROCESS
GOBP_REGULATION_OF_CELL_CYCLE_G1_S_PHASE_TRANSITION
HP_THIN_UPPER_LIP_VERMILION
HP_THIN_VERMILION_BORDER
HP_ABNORMALITY_OF_UPPER_LIP_VERMILLION
GOCC_TRANSCRIPTION_REPRESSOR_COMPLEX
GOBP_MACROAUTOPHAGY
HP_TRIANGULAR_FACE
GOBP_PROTEIN_LOCALIZATION_TO_NUCLEUS
GOMF_UBIQUITIN_LIKE_PROTEIN_LIGASE_BINDING
GOBP_REGULATION_OF_GLYCOPROTEIN_METABOLIC_PROCESS
GOMF_HISTONE_DEACETYLASE_BINDING
GOBP_NEGATIVE_REGULATION_OF_TRANSFERASE_ACTIVITY
GOCC_UBIQUITIN_LIGASE_COMPLEX
GOMF_SINGLE_STRANDED_RNA_BINDING
GOBP_MUSCLE_CELL_PROLIFERATION
GOBP_REGULATION_OF_MRNA_METABOLIC_PROCESS
GOMF_MRNA_BINDING
GOMF_CORE_PROMOTER_SEQUENCE_SPECIFIC_DNA_BINDING
GOBP_REGULATION_OF_RESPONSE_TO_OXIDATIVE_STRESS

Figure 14.4: An example of a graph produced by the EnrichmentMap plugin of the Cytoscape program using the 40 most likely annotations of `hsa_circ_0000228`. The labels were moved manually so that they do not overlap. Several red circles that correspond to annotation terms with lower p-values can be noticed. Besides that, there are several clusters of terms that share genes. The biggest is located in the top left corner showing a set of terms connected with the reproductive and urinary systems. This indicates that the circular RNA might be connected with those systems. According to circBase [106], the sequence of `hsa_circ_0000228` is located on the `ZEB1` gene. According to the NCBI summary of publication [83], the `ZEB1` gene shows the highest expression in the endometrium (out of 27 tissues presented) and is also highly expressed in the urinary bladder, placenta and prostatic tissues. Also, a recent publication has shown a connection between `hsa_circ_0000228` and cervical cancer [179]. circGPA predicts a link of the circRNA to cancer as well, given the fact that `HP_SOFT_TISSUE_SARCOMA` is the term with the lowest p-value.

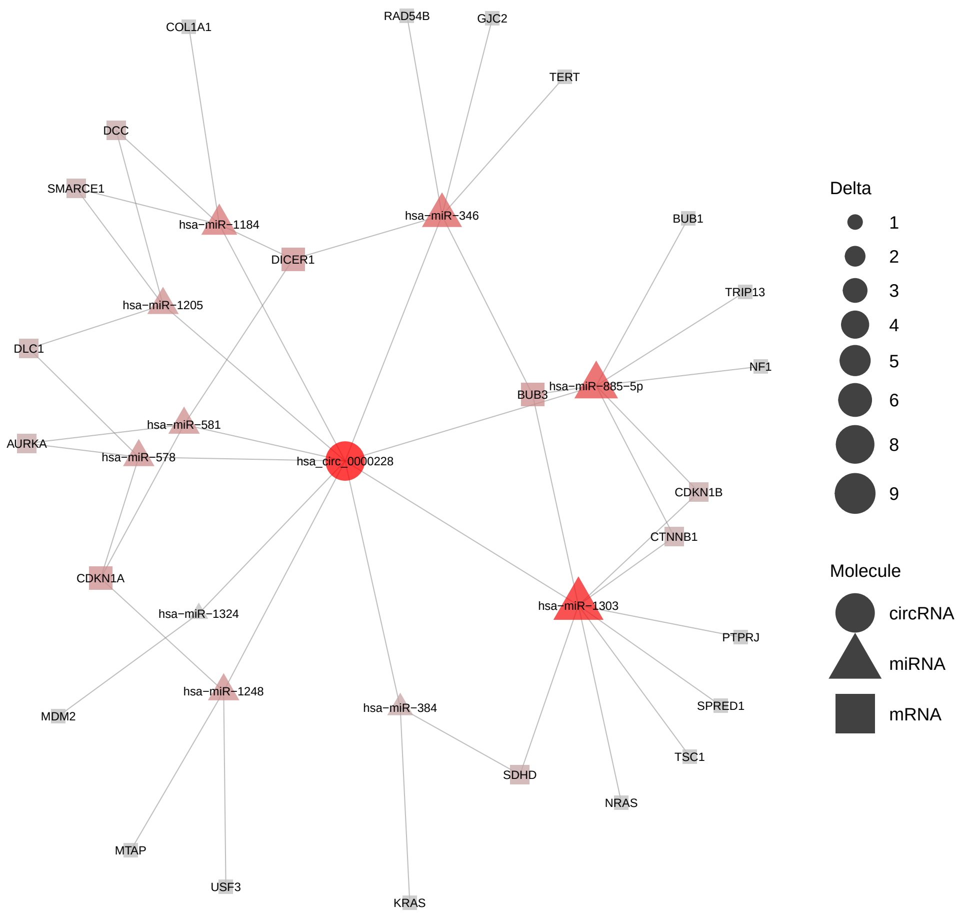


Figure 14.5: A network of miRNAs and mRNAs that back up the annotation of `hsa_circ_0000228` by the term with the lowest p-value — term `HP_SOFT_TISSUE_SARCOMA`. The size and color of each RNA shows how much the statistic (equal to 38) drops if the RNA is excluded from the graph together with all incident edges, i.e., the $\boldsymbol{\Delta}^{\mu}$ and $\boldsymbol{\Delta}^{m}$ values. See Section 14.4 for details.

deterministic ones calculated with generating polynomials. The comparison of runtimes is summarized in Figure 14.6. In those experiments, we used the sample of circRNAs summarized in Table 14.2. It is obvious that circGPA overcomes the stochastic algorithm for all the tested annotation terms with speedups that vary from 0.16 to 48,670. The average speedup per single p-value calculation proved to be 3,150. If we compare the overall runtime requirements of 59 hours needed to calculate the p-values exactly on the testing circRNAs (see Table 14.2) and 1967 hours using the sampling approach, we can see that our approach is 33 times faster as a whole. Extrapolating this number means that our approach to calculating the p-values can save 65 years of CPU-time to annotate all circRNAs in our database (compared to approximately two years needed to calculate the p-values using the exact approach). However, the results of circGPA are worse on densely connected circRNAs that are over-represented in our dataset (see Table 14.2). The real measurements have shown that the circGPA requires 20 months to annotate all circRNAs in the database, including graph construction and disk access.

Further insights into the speedups provided by the algorithm are in Figure 14.7, which demonstrates the runtime complexity of Algorithm 14.1. If we eliminate all terms that are constant for a given circular RNA from Formula (14.10) and re-evaluate the runtime of the two outer loops, the computational complexity is square of the size of the gene ontology term. As we see from Figure 14.7, the runtime measurements are almost a line. The slope of the line is circRNA-dependent. For more connected circular RNAs, the slope is higher.

Figure 14.9 demonstrates that the decreasing number of trials in the sampling approach leads to a decrease in the accuracy of p-value estimation. It is impossible to straightforwardly decrease the number of trials and still approach the exact p-values. To reach runtimes observed in circGPA, the sampling algorithm would need to work with no more than 350 trials. However, the p-values then do not match the exact ones. We consider `hsa_circ_0000228` with 352 sampling trials to illustrate the gap. The Spearman's ρ of the p-value vectors that capture those annotation terms that pass the 0.05 p-value significance threshold (the most likely to be truly relevant) is equal only to 0.93. The average relative difference between those is equal to 19, with a standard deviation of 178. The maximum relative difference is 2793, and the median 0.33. Excluding the first 20 terms with the lowest p-value decreases the maximum deviation to 176, the mean to 2.6, the standard deviation to 11.2, and the median stays 0.32. It needs to be said, however, that we followed the recommendation of [223] to use biased estimates.

14.11.3 The p-values Distribution

The applicability of calculated p-values is demonstrated in Figure 14.8. The histograms shown are mostly bimodal. The peak close to 0 p-values represents the cases where alternative hypotheses truly apply. The peak close to 1 p-value represents the cases where the network size and interactivity are not sufficient. Clearly, this peak is large, especially for the low-interacting circRNAs (see Table 14.2). We can notice that the most interacting circRNAs tend to attract more low-p-value annotation terms. On the contrary, for circRNAs with few interactions, there are only a few annotation terms with low p-values. This is actually the desired behavior as circRNAs that interact a lot may influence many other genes and pathways; however, circRNAs that have only a few interactions influence only a specific part of the cellular machinery. This situation is common in nature and can be explained by the well-known ”80-20 rule” [216].

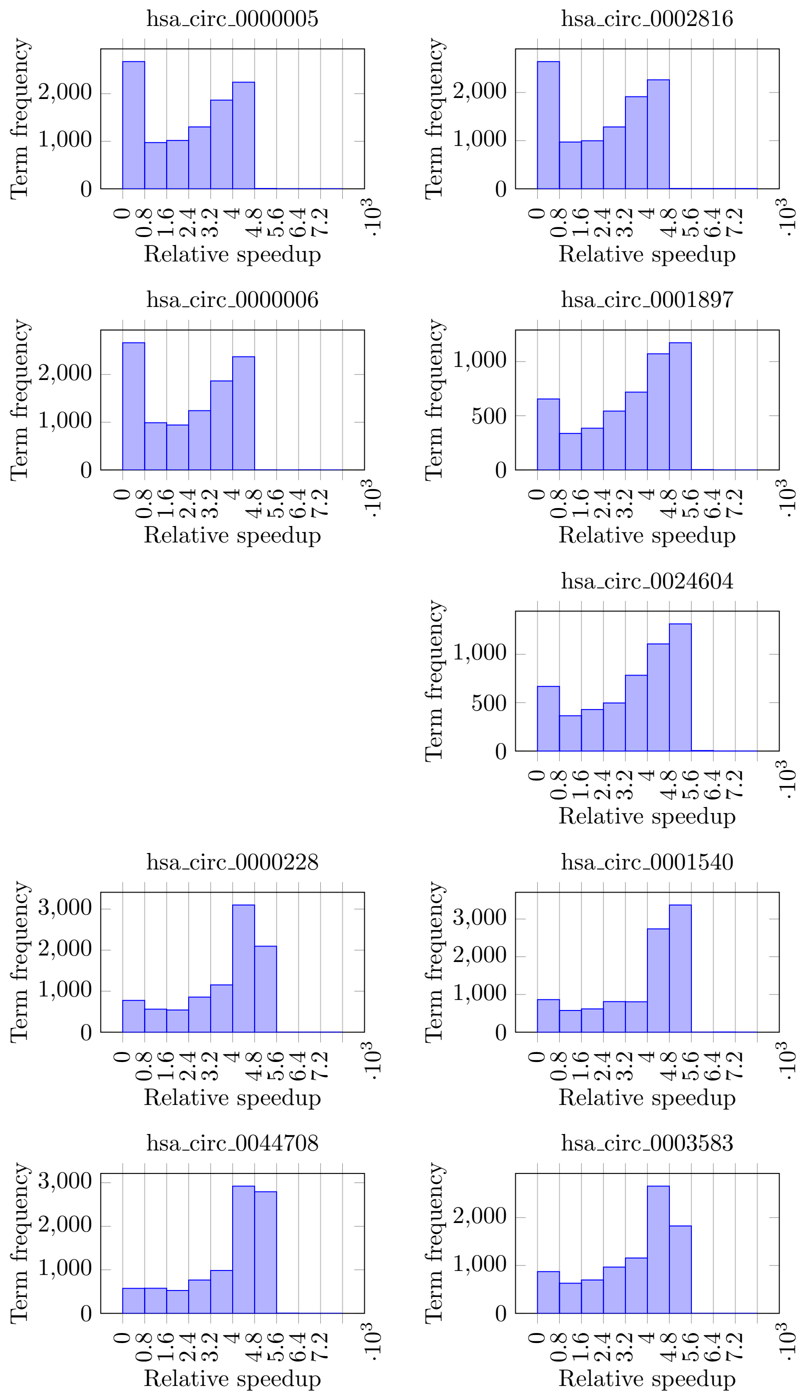


Figure 14.6: A histogram of the relative speedup of the p-value calculation using the generating polynomials compared to the sampling approach with the sample size 10^6. The speedup is shown only for the case when the p-value is not equal to 1 (i.e., when $s(c, g) = 0$). CircRNA `hsa_circ_0004624` is excluded from the plot as all p-values are equal to 1, meaning that the circRNA is not connected with any term.

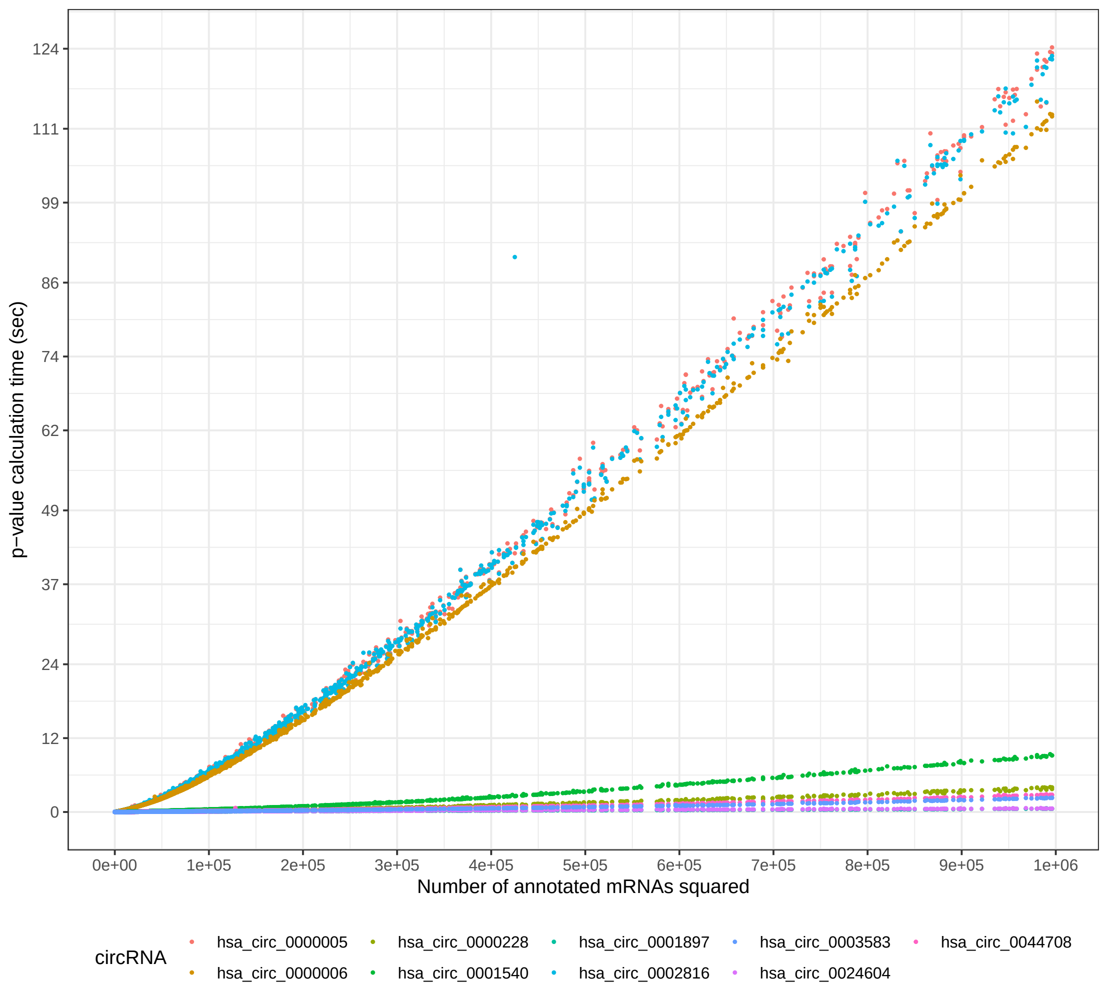


Figure 14.7: A plot of the time needed to calculate the p-value on $\|\mathbf{g}^m\|_1^2$. The plot was generated under the same conditions as Figure 14.6.

circRNA	Interacting miRNAs	Paths to mRNAs	Reason to include
`hsa_circ_0000005`	307	34043	top-interacting
`hsa_circ_0002816`	305	34113	top-interacting
`hsa_circ_0000006`	295	33470	top-interacting
`hsa_circ_0001897`	2	302	least-interacting
`hsa_circ_0004624`	2	0	least-interacting
`hsa_circ_0024604`	2	325	least-interacting
`hsa_circ_0000228`	26	1771	used in development
`hsa_circ_0001540`	22	5037	random choice
`hsa_circ_0044708`	11	1837	random choice
`hsa_circ_0003583`	12	1234	random choice

Table 14.2: CircRNAs used in the experiments, along with the reason we included them. As the p-value calculation using the sampling approach takes up to two weeks on a single circRNA, we limited the experiments only to those circRNAs.

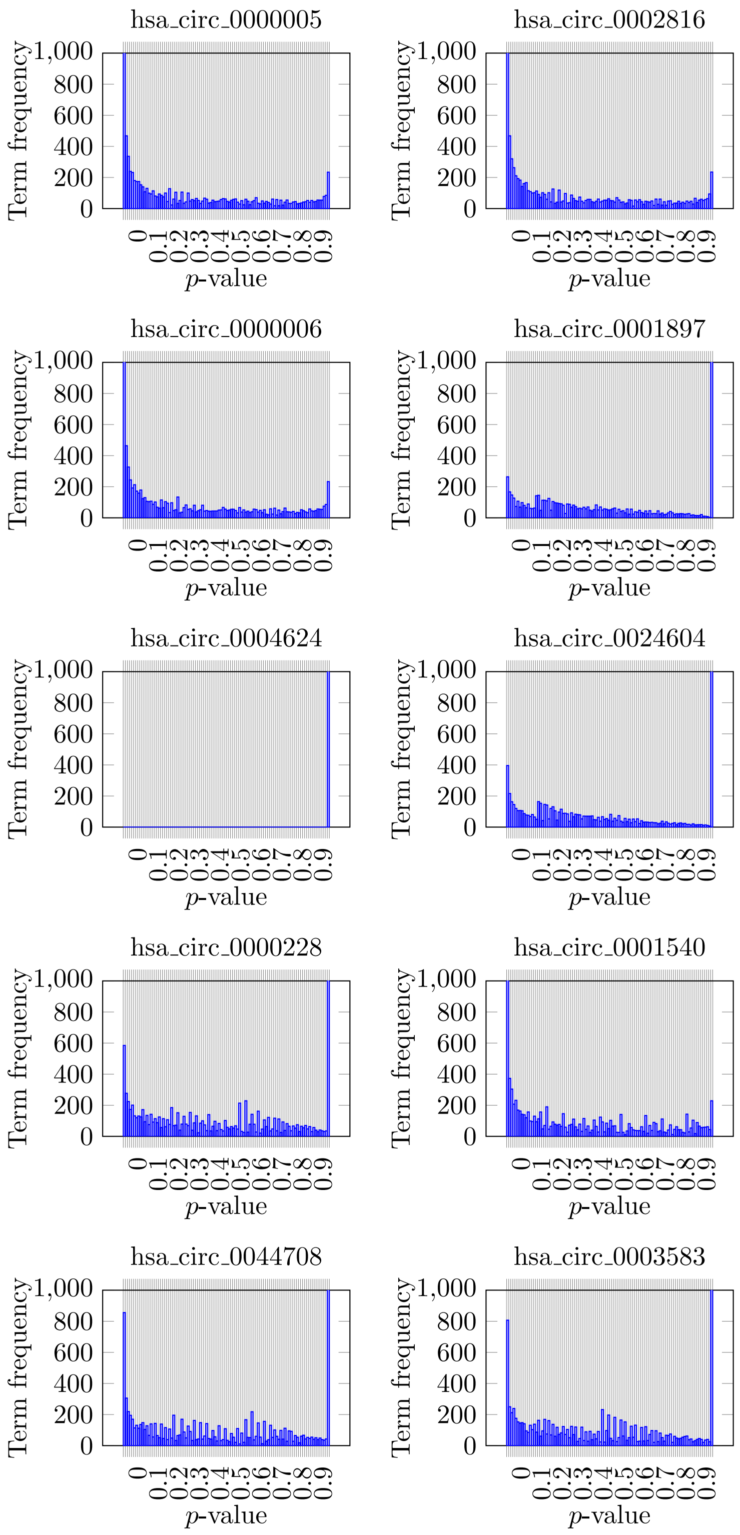


Figure 14.8: Histograms of the p-values before multiple hypothesis testing correction. The y axis is trimmed to 1,000 annotation terms.

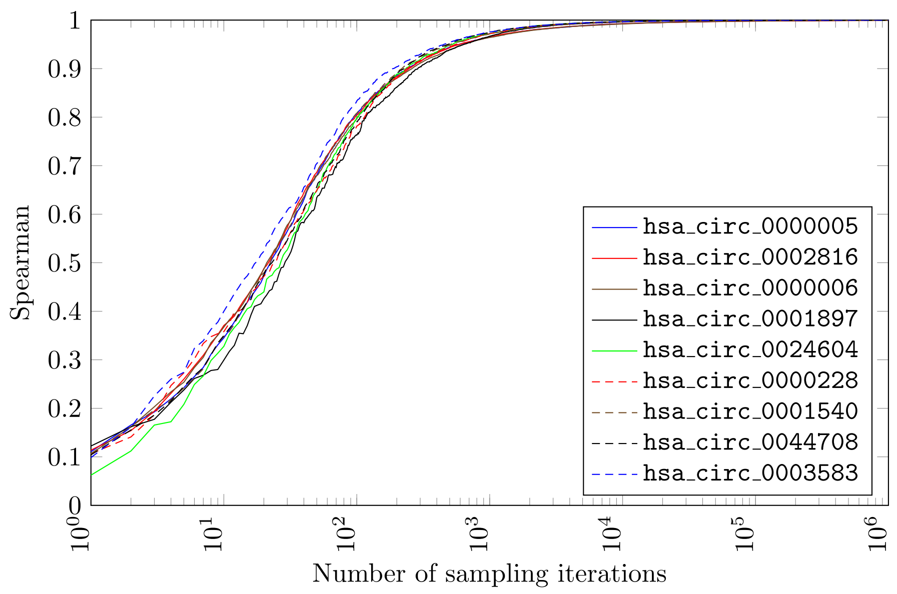


Figure 14.9: Dependence of Spearman's correlation between the p-values calculated by Algorithm 14.1 and the sampling approach. The correlation is calculated for the annotation terms with p-value smaller than 0.05 (i.e., those that are likely to be checked manually). The plot excludes `hsa_circ_0004624` as all p-values for this circRNA were equal to 1.

14.12 Discussion and Related Work [242]

The previous section clearly shows that circGPA is an efficient tool for circRNA functional annotation. Let us compare it conceptually with similar existing tools. The closest tool is Cerina [37] which also employs the circRNA-miRNA-mRNA interaction network for circRNA functional annotation, including GO terms. Their stochastic approach is based on permuting the connections between a given circRNA and its interacting miRNAs/mRNAs. To reduce the size of the interaction network, Cerina binds the interaction and expression data and uses the Pareto-front-based algorithm for their integrative analysis. This step increases the efficiency of the stochastic algorithm and gives a chance to work with tissue-specific interactions only. We mention the possibilities of circGPA for integrative analysis in Section 14.13.

The utilization of random-walk algorithms for functional annotation is wide. One of its early applications was the EnrichNet tool [105] for integrative analysis and gene annotation. Another usage is in [161], where the authors use random walks and circRNA similarity to predict circRNA-disease association. A similar approach utilized random walk for drug association prediction [323, 319]. The circRNA-disease association prediction problem was tackled using random walk with restarts [294, 69]. Fang et al. used random walks to predict miRNA-circRNA associations [86]. Close to the random walk with restarts algorithm is the PageRank algorithm [29] that has been developed for internet hyperlinks. If applied to the presented graph with circRNA-miRNA-mRNA interactions, both random walk with restarts and PageRank algorithms would lead to the same results as there is only a single source circRNA.

From the methodological point of view, the circRNA-disease association prediction is very similar to the problem solved by circGPA. The main condition for circGPA application in this task is the knowledge of miRNA/mRNA-disease annotations in the circRNA interaction network. These annotation databases exist. The miRNA-disease association databases include miR2Disease [137] with 3,273 associations, and HMDDv3

[129], which contains 35,547 manually collected miRNA-disease associations. Those associations were collected from 19,280 scientific papers. For gene-disease associations, we can mention the DisGeNET database [224]. The accuracy of prediction could be verified against the known circRNA disease annotations. The CircR2Cancer [158] provides a list of 1,439 manually curated entries. The Circ2Disease [314] is the most extensive database and contains 5,368 manually curated entries. The CircR2Disease database [84] contains 725 associations.

One of the early papers from the circRNA-disease association prediction field is the Circ2Traits database [103]. The authors worked with two data sources to predict the associations. Firstly, it was a miRNA-circRNA interaction graph together with statistical tests. Secondly, the algorithm incorporates knowledge about single nucleotide polymorphisms. Recent tools for circRNA-disease association prediction usually take advantage of similarities between the circRNA pairs and disease pairs to predict the associations. For example, a tool named PWCDA developed in [162] constructs three graphs. The first graph represents circRNA similarity; the second graph represents disease similarity. Those two graphs are used to form a disease-circRNA association graph, out of which association scores are calculated. A similar approach was used in [324], where a Gaussian interacting profile is used to evaluate the similarities. There were frequent machine learning applications to circRNA-disease association prediction too. The work of Lei et al. [139] employs recommender systems to overcome the sparsity of validated annotations. The authors of [183] used convolutionary networks on the k-mer representation of the circRNA sequences. The disease similarities are captured using the disease ontology terms. The iGRLCDA tool [317] uses Gaussian interacting profiles, convolutional networks, and graph factorization. The authors of this paper developed a similar tool for drug-disease association prediction named HINGRL [322] too. The tool DWNCPCDA [172] uses DeepWalk. DWNCPCDA is based on the work by a similar set of authors – the NCPCDA tool [171] based on network consistency projection. This brief overview is far from complete. Other tools include, among others, [320, 66, 299, 326].

A clear advantage of circGPA compared to the aforementioned circRNA-disease prediction algorithms is the existence of a p-value that allows filtering the predictions in a more advanced manner than selecting top k predictions. The runtime requirements of circGPA allow us to bulk annotate all known circRNAs with gene-ontology terms and provide the annotated results on `https://ida.fel.cvut.cz/~rysavy/circgpa/`.

14.13 Conclusion [242]

In this chapter, we proposed an annotation algorithm circGPA that identifies prospective links between circRNAs and annotation terms. The algorithm is deterministic and based on generating polynomials. We show that this approach is both more effective and efficient than the alternative stochastic approach frequently applied in a similar context.

Our approach could easily be generalized for related tasks. Besides circRNAs, the long non-coding RNAs can act as miRNA sponges, and their annotation could be predicted too. As a whole, the approach is generalizable on any interactions which can be represented by a directed acyclic graph where leaves are annotated with binary concepts (annotation terms). Our goal is to decide upon the annotation of the roots of the graph (ncRNAs whose annotation is unknown). There are, however, computational limits to our approach. The p-value calculation is limited by the fact that we need to fit a potentially large table into the memory. The size of the table is the number of paths from the vertex of interest multiplied by the size of the annotation term.

When we compared circGPA with the sampling approach, we set the number of sampling trials as a constant. The only optimization we did was when the p-value was equal to 1. However, for high p-values annotation terms, it would be possible to stop the sampling earlier, knowing that the p-value will not be smaller than a threshold with a high-enough probability. Such approaches were proposed in [264, 265] and used in `simctest` R package. Similar ideas could apply to circGPA. We might use the generating polynomial to say that the p-value will not be smaller than a threshold without evaluating all polynomial coefficients. In the same manner, the weights in the loop of the `Generation-Polynomial` function could be sorted so that a bound on the p-value could be provided in the middle of computation.

In future work, we will look at the integrative analysis that deals with additional data modalities. So far, we have only examined interaction graphs. In the future, the annotation should stem from sequential data too in order not to rely on binary interaction records only. Also, tissue-specific expression data can help to minimize the impact of false-positive interactions with negligible expression and focus our analysis. The hierarchy of annotation terms could serve to regularize the eventual annotation records.

14.14 Future Work - Integration of Expression Data

Once the circGPA algorithm works for the general graph, it is natural to propose an extension of the approach so that other data modalities are captured. A straightforward one can be based on the expression profiles of the interacting molecules. For any circRNA, miRNA, and mRNA, their expression pattern among samples can be calculated. Then, the absolute value of the correlation between each pair with known interaction can be calculated. By this way, we obtain a correlation vector $\boldsymbol{\rho}^{\mu,c} \in [0,1]^{|\mu|}$ and a correlation matrix $\boldsymbol{\rho}^{m,\mu} \in [0,1]^{|m|\times|\mu|}$. In $\boldsymbol{\rho}^{\mu,c}$, each field represents the absolute value of the correlation between the circRNA of interest, and particular miRNA, in $\boldsymbol{\rho}^{m,\mu}$, each field represents the absolute value of the correlation between the particular mRNA and miRNA.

Correlation vector $\boldsymbol{\rho}^{\mu,c}$ and matrix $\boldsymbol{\rho}^{m,\mu}$ can then be used as weights of the edges in the interaction graph. In such a case, each edge is weighted by the respective correlation instead of default 1. A path to a miRNA is then weighted by the correlation between the miRNA and circRNA c. A path to a mRNA is weighted by the product of the correlation from circRNA c to a miRNA and the correlation from the miRNA to the mRNA. In such a case, a straightforward modification of the statistic in (14.1) is

$$s'(c,g) = (\mathbf{a}^{\mu,c} \odot \boldsymbol{\rho}^{\mu,c})^T \mathbf{g}^{\mu} + ((\mathbf{A}^{m,\mu} \odot \boldsymbol{\rho}^{m,\mu})(\mathbf{a}^{\mu,c} \odot \boldsymbol{\rho}^{\mu,c}))^T \mathbf{g}^{m}. \tag{14.11}$$

The symbol $\odot$ denotes the Hadamar product, i.e., the field-wise multiplication of vectors and matrices of the same dimensions in which $(\mathbf{A} \odot \mathbf{B})_{ij} = (\mathbf{A})_{ij} \cdot (\mathbf{B})_{ij}$. By this modification, statistic s' reflects the importance of individual interactions between the RNAs in the current measurement, thus allowing us to observe which annotation terms are currently active with respect to the current circRNA. The pre-selection of the differentially expressed circular RNAs can then be used to obtain deeper insight into pathways that modulate the differential gene expression between analyzed samples and control samples.

For the p-value calculation, there are several options. The first and most straightforward option is to use Algorithm 14.1. However, this algorithm was designed to work with integer weights of miRNAs and mRNAs. Naturally, we will be interested only in a fraction of circRNAs and annotation terms which in turn allows us to spend more computational time on those circRNAs. However, still, the full evaluation of the generating polynomial is not feasible. Recall that, for example, for a mRNA with two paths from the circRNA, i.e., weight 2, the term in the generating polynomial multiplication was $1 + x^2y^1$. When those paths have weights (obtained from the correlations) of 0.457 and 0.312, then the weight of the mRNA is 0.769, and the respective multiplicand is $1 + x^{0.769}y^1$. When the term was $1 + x^2y^1$, we were able to exploit the binomial theorem as there were likely more multiplicands in the form of $1 + x^2y^1$. In such a case, the algorithm benefited from the repeated powers by x. When the weight is $1 + x^{0.769}y^1$, this repetition is extremely unlikely (and with double precision almost impossible), hence, the algorithm is burdened by the increased number of terms with grows exponentially with each mRNA reachable from the circRNA in the interaction graph with non-zero weights.

A possible solution to the problem is to use discretization. With precision up to 0.1 and rounding, the expected time and memory complexity grow only ten times compared to the original problem. In such a case, the algorithm still remains faster than Barnard's Monte-Carlo sampling algorithm [192]. Precision up to 0.01 means 100 times slower calculation which is still admissible in the context that fewer circRNAs will be evaluated. Precision up to 0.001 then causes that the sampling would be preferred to Algorithm 14.1.

In future work, we will aim to overcome those obstacles with the p-value calculation, and aim for a more precise and faster calculation, so that we can generalize circGPA to the integrative analysis.

Chapter 15 Conclusion and Future Work

In Chapters 5, 8, and 9, we have seen methods that can exploit reads in unsupervised learning, namely in phylogenetic tree reconstruction. Those two methods can produce reasonable estimates of the Levenshtein distance [167], which is, in many applications, one of the possible choices of distance measures in the alignment-based approaches for phylogeny.

Although we motivated our approach by approximating the Levenshtein distance, the method proposed in Chapter 5 calculates the alignment only locally. The alignment is calculated between very short sequences, which makes it applicable to larger genomes than the original Levenshtein distance. The approximated distance does not reflect the possibility of transpositions and inversions, while this is reflected in the proposed distance.

The proposed approaches require lower coverage c and read length l, which makes them applicable in more situations than the conventional assembly-based methods. By avoiding the necessity of complete assembly and avoiding the scaffolding and gap filling, it is possible to reduce the coverage and/or read length in order to obtain a similar phylogenetic tree. Therefore, the proposed methods may result in less wet-lab work sequencing.

Later, in Chapter 14, we proposed the circGPA algorithm that captures similarities between the circRNAs by annotation terms, i.e., for example, the gene ontology terms. As this method stands on its alone, we presented it, together with the experiments, in a separate chapter. At the end of the chapter, we discussed possibilities to incorporate RNA-seq data into the similarities by capturing the correlations on the interactions and the necessity of further discretization. As a result, the method can capture similarities via the gene ontology terms, which then, in turn, can be used in learning methods, especially in symbolic learning.

For both methods, we provided the p-value and other statistical and theoretical background and experimental evaluation.

15.1 Future Work - Insights into the Methods

Despite the fact that the proposed methods have been published, there are still many chances for future development. First, there are technical things that can be improved. For example, some of the method parameters need better insight. In Chapter 8, we selected only overlaps longer than 20 symbols, which we justified theoretically by the common read lengths and t used in the margin gap methods. Nevertheless, it would be good to show the experimental sensitivity of the method to this parameter. The thresholding proposed in Section 5.1.5 may sometimes be harmful. For this reason, we excluded it from the measure in Chapter 9. For low-coverage data, a read may miss its match in the other read bag due to randomness. Those possibilities could be addressed in additional experiments.

The theoretical background was mostly developed for the initial method in Chapter 5. However, there are still possibilities for theoretical insights into the method in Chapter 8 or the combination of the latter. It is, for example, trivial to show the claim that under the assumption that all overlaps were properly identified in Section 8.2, the sum of distances is not overestimated. Similarly, Chapter 7, which provides a lower-bound view on the Levenshtein distance, still contains some possibility to develop better guarantees when two read bags are considered.

15.2 Future Work - Improving Runtime and Quality

The embedding used in Section 10.2 is simple. In practical applications, it would be better to avoid the quadratic search by using a more enhanced data structure as k-d trees [21]. Different embeddings, such as [39] or [214], should be tested in the matter of quality and accuracy. Because both read bags are supposed to be approximately of the same size, it may be worth traversing both k-d trees of their embeddings simultaneously to avoid the quadratic work. This may be similar to the problem we tried to solve with tries in Section 10.3.

Another way to improve the runtime may include pruning based on upper bounds and lower bounds on the Levenshtein distance. Papers [259] and [57] propose examples of such bounds.

A further idea to reduce the runtime may consist in providing a non-accurate metric that requires only a very low coverage ($c = 0.5$ or less). Such a metric would provide low-quality results, however, very fast (recall that the runtime is quadratic in c as shown in Section 5.2.1. However, from multiple low-quality estimates, it may be possible to obtain a good ensemble [97], while avoiding the c^2 factor and having a dependency only on c instead.

15.3 Future Work - Extending the Methods

Further, we might be interested in incorporating the quality scores into the measure. Illumina, the major sequencing technology nowadays, provides in FASTQ files estimates of the quality of the sequenced data. These quality scores were used, for example, in [55]. The quality scores could be used to weight possible mismatches by their probabilities so that the measure accounts for the possibility of sequencing errors.

Another direction of future research is to focus on Nanopore sequencing [46]. This method relies on passing a DNA molecule through a protein which is used to measure the electric characteristics of the molecule. As a result, the method is capable of reading long reads spanning over a thousand or more symbols. A clear disadvantage is a high number of sequencing errors, especially indels, that are caused by the limitations in the regulation of the speed by which the DNA molecule passes through the protein. As a result, the Nanopore methods are able to report that a nucleotide has passed through the protein, but they cannot be sure how many times the nucleotide was repeated in the sequence.

List of Figures

List of Tables

Bibliography

[1] 1000 Genomes Project Consortium et al. A global reference for human genetic variation. *Nature*, 526(7571):68–74, 2015. doi:10.1038/nature15393. Cited on pages 6 and 93.

[2] Mostefai Abdelkader. A method based on WordNet and Monge-Elkan distance for business process model matching. *Int. J. Inf. Syst. Model. Des.*, 9(4):37–48, oct 2018. doi:10.4018/IJISMD.2018100103. Cited on page 52.

[3] Saulo Alves Aflitos, Edouard Severing, Gabino Sanchez-Perez, Sander Peters, Hans de Jong and Dick de Ridder. Cnidaria: fast, reference-free clustering of raw and assembled genome and transcriptome NGS data. *BMC Bioinformatics*, 16(1):352, Nov 2015. doi:10.1186/s12859-015-0806-7. Cited on pages 28 and 30.

[4] Nima Aghaeepour, Radina Nikolic, Holger H. Hoos and Ryan R. Brinkman. Rapid cell population identification in flow cytometry data. *Cytometry Part A*, 79A(1):6–13, 2011. doi:10.1002/cyto.a.21007. Cited on page 4.

[5] Alfred V. Aho and Margaret J. Corasick. Efficient string matching: An aid to bibliographic search. *Communications of the ACM*, 18(6):333–340, June 1975. doi:10.1145/360825.360855. Cited on page 81.

[6] H. Raza Ali, Hartland W. Jackson, Vito R. T. Zanotelli, Esther Danenberg, Jana R. Fischer et al. Imaging mass cytometry and multiplatform genomics define the phenogenomic landscape of breast cancer. *Nature Cancer*, 1(2):163–175, Feb 2020. doi:10.1038/s43018-020-0026-6. Cited on page 4.

[7] S F Altschul. Generalized affine gap costs for protein sequence alignment. *Proteins*, 32 (1):88–96, July 1998. Cited on page 22.

[8] Stephen F. Altschul. Amino acid substitution matrices from an information theoretic perspective. *Journal of Molecular Biology*, 219(3):555 – 565, 1991. doi:10.1016/0022-2836(91)90193-A. Cited on pages 11 and 22.

[9] Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers and David J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990. doi:10.1016/S0022-2836(05)80360-2. Cited on pages 12, 22, and 85.

[10] Stephen F. Altschul, Thomas L. Madden, Alejandro A. Schäffer, Jinghui Zhang, Zheng Zhang et al. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, 09 1997. doi:10.1093/nar/25.17.3389. Cited on page 22.

[11] Simon Anders, Paul Theodor Pyl and Wolfgang Huber. HTSeq – a python framework to work with high-throughput sequencing data. *bioRxiv*, 2014. doi:10.1101/002824. Cited on page 4.

[12] Stephen Anderson. Shotgun DNA sequencing using cloned DNase I-generated fragments. *Nucleic Acids Research*, 9(13):3015–3027, 1981. doi:10.1093/nar/9.13.3015. Cited on page 2.

[13] Oswald T. Avery, Colin M. MacLeod and Maclyn McCarty. Studies on the chemical nature of the substance inducing transformation of pneumococcal types. *Journal of Experimental Medicine*, 79(2):137–158, 1944. doi:10.1084/jem.79.2.137. Cited on page 1.

[14] Ricardo Baeza-Yates. Efficient text searching. PhD thesis, Department of Computer Science, University of Waterloo, Waterloo, Ontario, 1989. Cited on page 23.

[15] Ergude Bao, Tao Jiang, Isgouhi Kaloshian and Thomas Girke. SEED: efficient clustering of next-generation sequences. *Bioinformatics*, 27(18):2502–2509, 2011. doi:10.1093/bioinformatics/btr447. Cited on page 24.

[16] Barthel Barlogie, Martin N. Raber, Johannes Schumann, Tod S. Johnson, Benjamin Drewinko et al. Flow cytometry in clinical cancer research. *Cancer Research*, 43(9): 3982–3997, 1983. Cited on page 4.

[17] Geoffrey J. Barton and Michael J.E. Sternberg. A strategy for the rapid multiple alignment of protein sequences: Confidence levels from tertiary structure comparisons. *Journal of Molecular Biology*, 198(2):327–337, 1987. doi:10.1016/0022-2836(87)90316-0. Cited on page 11.

[18] Serafim Batzoglou, David B. Jaffe, Ken Stanley, Jonathan Butler, Sante Gnerre et al. ARACHNE: A whole-genome shotgun assembler. *Genome Research*, 12(1):177–189, 2002. doi:10.1101/gr.208902. Cited on page 32.

[19] David C. Bell, W. Kelley Thomas, Katelyn M. Murtagh, Cheryl A. Dionne, Adam C. Graham et al. DNA base identification by electron microscopy. *Microscopy and Microanalysis*, 18(5):1049–1053, 2012. doi:10.1017/S1431927612012615. Cited on page 4.

[20] Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(1):289–300, 1995. Cited on page 122.

[21] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, September 1975. doi:10.1145/361002.361007. Cited on page 136.

[22] Hal Berghel and David Roach. An extension of Ukkonen's enhanced dynamic programming ASM algorithm. *ACM Trans. Inf. Syst.*, 14(1):94–106, January 1996. doi:10.1145/214174.214183. Cited on pages 22 and 87.

[23] Philip S Bernard and Carl T Wittwer. Real-Time PCR Technology for Cancer Diagnostics. *Clinical Chemistry*, 48(8):1178–1185, 08 2002. doi:10.1093/clinchem/48.8.1178. Cited on page 4.

[24] David Binns, Emily Dimmer, Rachael Huntley, Daniel Barrell, Claire O'Donovan and Rolf Apweiler. QuickGO: a web-based tool for Gene Ontology searching. *Bioinformatics*, 25(22):3045–3046, 09 2009. doi:10.1093/bioinformatics/btp536. Cited on page 120.

[25] B. Edwin Blaisdell. A measure of the similarity of sets of sequences not requiring sequence alignment. *Proceedings of the National Academy of Sciences*, 83(14):5155–5159, 1986. Cited on pages 27, 28, and 91.

[26] Marcus Boden, Martin Schöneich, Sebastian Horwege, Sebastian Lindner, Chris Leimeister and Burkhard Morgenstern. Alignment-free sequence comparison with spaced k-mers. In Tim Beißbarth, Martin Kollmar, Andreas Leha, Burkhard Morgenstern, Anne-Kathrin Schultz et al, editors, *German Conference on Bioinformatics 2013*, volume 34 of *OpenAccess Series in Informatics (OASIcs)*, pages 24–34, Dagstuhl, Germany, 2013. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-939897-59-0. doi:10.4230/OASIcs.GCB.2013.24. Cited on pages 28 and 30.

[27] Oliver Bonham-Carter, Joe Steele and Dhundy Bastola. Alignment-free genetic sequence comparisons: a review of recent approaches by word analysis. *Briefings in Bioinformatics*, 15(6):890–905, 07 2013. doi:10.1093/bib/bbt052. Cited on page 29.

[28] Ivan Borozan, Stuart Watt and Vincent Ferretti. Integrating alignment-based and alignment-free sequence similarity measures for biological sequence classification. *Bioinformatics*, 31(9):1396–1404, 01 2015. doi:10.1093/bioinformatics/btv006. Cited on page 29.

[29] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1):107–117, 1998. doi:10.1016/S0169-7552(98)00110-X. Proceedings of the Seventh International World Wide Web Conference. Cited on page 130.

[30] Roy J. Britten, Lee Rowen, John Williams and R. Andrew Cameron. Majority of divergence between closely related DNA samples is due to indels. *Proceedings of the National Academy of Sciences*, 100(8):4661–4665, 2003. doi:10.1073/pnas.0330964100. Cited on page 10.

[31] R. L. Brooks. On colouring the nodes of a network. *Mathematical Proceedings of the Cambridge Philosophical Society*, 37(2):194–197, 1941. doi:10.1017/S030500410002168X. Cited on page 63.

[32] Ivan Brukner, Yves Longtin, Matthew Oughton, Vincenzo Forgetta and Andre Dascal. Assay for estimating total bacterial load: relative qPCR normalisation of bacterial load with associated clinical implications. *Diagnostic Microbiology and Infectious Disease*, 83 (1):1–6, 2015. doi:10.1016/j.diagmicrobio.2015.04.005. Cited on page 6.

[33] Peter Buneman. A note on the metric properties of trees. *Journal of Combinatorial Theory, Series B*, 17(1):48 – 50, 1974. doi:10.1016/0095-8956(74)90047-1. Cited on page 26.

[34] Natascha Bushati and Stephen M. Cohen. microRNA functions. *Annual Review of Cell and Developmental Biology*, 23(1):175–205, 2007. doi:10.1146/annurev.cellbio.23.090506.123406. PMID: 17506695. Cited on page 1.

[35] Jonathan Butler, Iain MacCallum, Michael Kleber, Ilya A. Shlyakhter, Matthew K. Belmonte et al. ALLPATHS: De novo assembly of whole-genome shotgun microreads. *Genome Research*, 18(5):810–820, 2008. doi:10.1101/gr.7337908. Cited on page 33.

[36] David G. Cantor and Erich Kaltofen. On fast multiplication of polynomials over arbitrary algebras. *Acta Informatica*, 28(7):693–701, Jul 1991. doi:10.1007/BF01178683. Cited on page 52.

[37] Jacob Cardenas, Uthra Balaji and Jinghua Gu. Cerina: systematic circRNA functional annotation based on integrative analysis of ceRNA interactions. *Scientific Reports*, 10(1): 22165, Dec 2020. doi:10.1038/s41598-020-78469-x. Cited on page 130.

[38] Selahattin Baris Cay, Yusuf Ulas Cinar, Selim Can Kuralay, Behcet Inal, Gokmen Zararsiz et al. Genome skimming approach reveals the gene arrangements in the chloroplast genomes of the highly endangered ¨Crocus L. species: Crocus istanbulensis (B.Mathew) rukšāns. *PLOS ONE*, 17(6):1–19, 06 2022. doi:10.1371/journal.pone.0269747. Cited on page 31.

[39] Diptarka Chakraborty, Elazar Goldenberg and Michal Koucký. Streaming algorithms for embedding and computing edit distance in the low distance regime. In *Proceedings of the Forty-eighth Annual ACM Symposium on Theory of Computing*, STOC '16, pages 712–725, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4132-5. doi:10.1145/2897518.2897577. Cited on pages 23 and 136.

[40] Michelle Cheatham and Pascal Hitzler. String similarity metrics for ontology alignment. In Harith Alani, Lalana Kagal, Achille Fokoue, Paul Groth, Chris Biemann et al, editors, *The Semantic Web – ISWC 2013*, pages 294–309, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-41338-4. doi:10.1007/978-3-642-41338-4_19. Cited on page 53.

[41] Geng Chen, Baitang Ning and Tieliu Shi. Single-cell RNA-seq technologies and related computational data analysis. *Frontiers in Genetics*, 10, 2019. doi:10.3389/fgene.2019.00317. Cited on page 4.

[42] Nae-Chyun Chen, Brad Solomon, Taher Mun, Sheila Iyer and Ben Langmead. Reference flow: reducing reference bias using multiple population genomes. *Genome Biology*, 22(1): 8, Jan 2021. doi:10.1186/s13059-020-02229-3. Cited on page 16.

[43] Nicos Christofides. Worst-case analysis of a new heuristic for the traveling salesman problem. Technical report, Carnegie-Mellon University, Management Sciences Research Group, 02 1976. URL `http://www.dtic.mil/cgi-bin/GetTRDoc?Location=U2&doc=GetTRDoc.pdf&AD=ADA025602`. Cited on page 32.

[44] Justin Chu, Hamid Mohamadi, Emre Erhan, Jeffery Tse, Readman Chiu et al. Mismatch-tolerant, alignment-free sequence classification using multiple spaced seeds and multiindex bloom filters. *Proceedings of the National Academy of Sciences*, 117(29): 16961–16968, 2020. doi:10.1073/pnas.1903436117. Cited on pages 28 and 31.

[45] Vacláv Chvátal and David Sankoff. Longest common subsequences of two random sequences. *Journal of Applied Probability*, 12(2):306–315, 1975. doi:10.2307/3212444. Cited on page 43.

[46] James Clarke, Hai-Chen Wu, Lakmal Jayasinghe, Alpesh Patel, Stuart Reid and Hagan Bayley. Continuous base identification for single-molecule nanopore DNA sequencing. *Nature Nanotechnology*, 4(4):265–270, Apr 2009. doi:10.1038/nnano.2009.12. Cited on pages 3 and 136.

[47] Ian C. Clift. Diagnostic Flow Cytometry and the AIDS Pandemic. *Laboratory Medicine*, 46(3):e59–e64, 08 2015. doi:10.1309/LMKHW2C86ZJDRTFE. Cited on page 4.

[48] Guy Cochrane, Ilene Karsch-Mizrachi and Yasukazu Nakamura. The International Nucleotide Sequence Database Collaboration. *Nucleic Acids Research*, 39(suppl_1): D15–D18, 2011. doi:10.1093/nar/gkq1150. Cited on page 6.

[49] William W. Cohen, Pradeep Ravikumar and Stephen E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *Proceedings of the 2003 International Conference on Information Integration on the Web*, IIWEB'03, page 73–78. AAAI Press, 2003. Cited on page 52.

[50] M. Comin and D. Verzotto. Whole-genome phylogeny by virtue of unic subwords. In *2012 23rd International Workshop on Database and Expert Systems Applications*, pages 190–194, Sept 2012. doi:10.1109/DEXA.2012.10. Cited on page 29.

[51] Matteo Comin and Michele Schimd. Assembly-free genome comparison based on next-generation sequencing reads and variable length patterns. *BMC Bioinformatics*, 15 (9):S1, Sep 2014. doi:10.1186/1471-2105-15-S9-S1. Cited on pages 28 and 29.

[52] Matteo Comin and Michele Schimd. Assembly-free techniques for NGS data. In Mourad Elloumi, editor, *Algorithms for Next-Generation Sequencing Data: Techniques, Approaches, and Applications*, pages 327–355. Springer International Publishing, 2017. doi:10.1007/978-3-319-59826-0_14. Cited on page 30.

[53] Matteo Comin and Davide Verzotto. Alignment-free phylogeny of whole genomes using underlying subwords. *Algorithms for Molecular Biology*, 7(1):34, Dec 2012. doi:10.1186/1748-7188-7-34. Cited on pages 28 and 29.

[54] Matteo Comin, Andrea Leoni and Michele Schimd. Qcluster: Extending alignment-free measures with quality values for reads clustering. In Dan Brown and Burkhard Morgenstern, editors, *Algorithms in Bioinformatics: 14th International Workshop, WABI 2014, Wroclaw, Poland, September 8-10, 2014. Proceedings*, pages 1–13. Springer Berlin Heidelberg, 2014. doi:10.1007/978-3-662-44753-6_1. Cited on pages 28 and 30.

[55] Matteo Comin, Andrea Leoni and Michele Schimd. Clustering of reads with alignment-free measures and quality values. *Algorithms for Molecular Biology*, 10(1):4, Jan 2015. doi:10.1186/s13015-014-0029-x. Cited on pages 28, 30, 91, and 136.

[56] The Gene Ontology Consortium. The Gene Ontology project in 2008. *Nucleic Acids Research*, 36(suppl_1):D440–D444, 11 2007. doi:10.1093/nar/gkm883. Cited on page 7.

[57] Graham Cormode and Shan Muthukrishnan. The string edit distance matching problem with moves. *ACM Transactions on Algorithms Algorithms*, 3(1):2:1–2:19, February 2007. doi:10.1145/1186810.1186812. Cited on page 136.

[58] Francis HC Crick. On protein synthesis. In *Symp Soc Exp Biol*, volume 12, page 8, 1958. Cited on page 1.

[59] Ralf Dahm. Discovering DNA: Friedrich Miescher and the early years of nucleic acid research. *Human Genetics*, 122(6):565–581, Jan 2008. doi:10.1007/s00439-007-0433-0. Cited on page 1.

[60] Qi Dai, Yanchun Yang and Tianming Wang. Markov model plus k-word distributions: a synergy that produces novel statistical measures for sequence comparison. *Bioinformatics*, 24(20):2296–2302, 2008. doi:10.1093/bioinformatics/btn436. Cited on pages 28 and 29.

[61] Fred J. Damerau. A technique for computer detection and correction of spelling errors. *Commun. ACM*, 7(3):171–176, March 1964. doi:10.1145/363958.363994. Cited on pages 11 and 23.

[62] Vincent Daubin and Manolo Gouy. Bacterial molecular phylogeny using supertree approach. *Genome Informatics*, 12:155–164, 2001. doi:10.11234/gi1990.12.155. Cited on page 112.

[63] William H.E. Day, David S. Johnson and David Sankoff. The computational complexity of inferring rooted phylogenies by parsimony. *Mathematical Biosciences*, 81(1):33 – 42, 1986. doi:10.1016/0025-5564(86)90161-6. Cited on page 25.

[64] Margaret O. Dayhoff and Bruce C. Orcutt Robert M. Schwartz. A model of evolutionary change in proteins. In *Atlas of Protein Sequence and Structure*, volume 5, pages 345–352. National Biomedical Research Foundation, 1978. Cited on page 22.

[65] Rene De La Briandais. File searching using variable length keys. In *Papers Presented at the the March 3-5, 1959, Western Joint Computer Conference*, IRE-AIEE-ACM '59 (Western), pages 295–298, New York, NY, USA, 1959. ACM. doi:10.1145/1457838.1457895. Cited on page 23.

[66] K. Deepthi and A.S. Jereesh. An ensemble approach for circRNA-disease association prediction based on autoencoder and deep neural network. *Gene*, 762:145040, 2020. doi:10.1016/j.gene.2020.145040. Cited on page 131.

[67] A. P. Dempster, N. M. Laird and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977. Cited on page 31.

[68] Thomas G. Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems*, pages 1–15, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. ISBN 978-3-540-45014-6. Cited on page 70.

[69] Yulian Ding, Bolin Chen, Xiujuan Lei, Bo Liao and Fang-Xiang Wu. Predicting novel CircRNA-disease associations based on random walk and logistic regression model. *Computational Biology and Chemistry*, 87:107287, 2020. doi:10.1016/j.compbiolchem.2020.107287. Cited on pages 115 and 130.

[70] Zhihao Ding, Massimo Mangino, Abraham Aviv, UK10K Consortium, Tim Spector and Richard Durbin. Estimating telomere length from whole genome sequence data. *Nucleic Acids Research*, 42(9):e75–e75, 03 2014. doi:10.1093/nar/gku181. Cited on page 18.

[71] Alexander Dobin, Carrie A. Davis, Felix Schlesinger, Jorg Drenkow, Chris Zaleski et al. STAR: ultrafast universal RNA-seq aligner. *Bioinformatics*, 29(1):15–21, 2013. doi:10.1093/bioinformatics/bts635. Cited on pages 4 and 16.

[72] 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, 2007. doi:10.1101/gr.6435207. Cited on page 32.

[73] Alexander P Douglass, Caoimhe E O'Brien, Benjamin Offei, Aisling Y Coughlan, Raúl A Ortiz-Merino et al. Coverage-Versus-Length Plots, a Simple Quality Control Step for de Novo Yeast Genome Sequence Assemblies. *G3 Genes—Genomes—Genetics*, 9(3): 879–887, 03 2019. doi:10.1534/g3.118.200745. Cited on pages 14 and 17.

[74] Radoje Drmanac. The advent of personal genome sequencing. *Genetics in Medicine*, 13 (3):188–190, Mar 2011. doi:10.1097/GIM.0b013e31820f16e6. Cited on page 2.

[75] Radoje Drmanac, Andrew B. Sparks, Matthew J. Callow, Aaron L. Halpern, Norman L. Burns et al. Human genome sequencing using unchained base reads on self-assembling DNA nanoarrays. *Science*, 327(5961):78–81, 2010. doi:10.1126/science.1181498. Cited on page 3.

[76] Dawood B. Dudekula, Amaresh C. Panda, Ioannis Grammatikakis, Supriyo De, Kotb Abdelmohsen and Myriam Gorospe. CircInteractome: A web tool for exploring circular RNAs and their interacting proteins and microRNAs. *RNA Biology*, 13(1):34–42, 2016. doi:10.1080/15476286.2015.1128065. PMID: 26669964. Cited on page 120.

[77] Olive Jean Dunn. Multiple comparisons among means. *Journal of the American Statistical Association*, 56(293):52–64, 1961. doi:10.1080/01621459.1961.10482090. Cited on page 122.

[78] Dirk Eddelbuettel and James Joseph Balamuta. Extending R with C++: A brief introduction to Rcpp. *The American Statistician*, 72(1):28–36, 2018. doi:10.1080/00031305.2017.1375990. Cited on page 120.

[79] John R. Edwards, Hameer Ruparel and Jingyue Ju. Mass-spectrometry DNA sequencing. *Mutation Research/Fundamental and Molecular Mechanisms of Mutagenesis*, 573(1): 3–12, 2005. doi:10.1016/j.mrfmmm.2004.07.021. Single Nucleotide Polymorphisms (SNPs): Detection, Interpretation, and Applications. Cited on page 4.

[80] Nicole Eger, Laura Schoppe, Susanne Schuster, Ulrich Laufs and Jes-Niels Boeckel. *Circular RNA Splicing*, pages 41–52. Springer Singapore, Singapore, 2018. ISBN 978-981-13-1426-1. doi:10.1007/978-981-13-1426-1_4. Cited on page 1.

[81] Michael Eisenstein. Oxford nanopore announcement sets sequencing sector abuzz. *Nature Biotechnology*, 30(4):295–296, Apr 2012. doi:10.1038/nbt0412-295. Cited on page 3.

[82] B Ewing and P Green. Base-calling of automated sequencer traces using phred. II. error probabilities. *Genome Res*, 8(3):186–194, March 1998. Cited on page 17.

[83] Linn Fagerberg, Björn M. Hallström, Per Oksvold, Caroline Kampf, Dijana Djureinovic et al. Analysis of the human tissue-specific expression by genome-wide integration of transcriptomics and antibody-based proteomics. *Molecular & Cellular Proteomics*, 13(2): 397–406, Feb 2014. doi:10.1074/mcp.M113.035600. Cited on page 124.

[84] Chunyan Fan, Xiujuan Lei, Zengqiang Fang, Qinghua Jiang and Fang-Xiang Wu. CircR2Disease: a manually curated database for experimentally supported circular RNAs associated with various diseases. *Database*, 2018, 05 2018. doi:10.1093/database/bay044. bay044. Cited on page 131.

[85] Huan Fan, Anthony R. Ives, Yann Surget-Groba and Charles H. Cannon. An assembly and alignment-free method of phylogeny reconstruction from next-generation sequencing data. *BMC Genomics*, 16(1):522, Jul 2015. doi:10.1186/s12864-015-1647-5. Cited on page 30.

[86] Zengqiang Fang and Xiujuan Lei. Prediction of miRNA-circRNA associations based on *k*-nn multi-label with random walk restart on a heterogeneous network. *Big Data Mining and Analytics*, 2(4):261–272, 2019. doi:10.26599/BDMA.2019.9020010. Cited on pages 115 and 130.

[87] James S. Farris. Outgroups and parsimony. *Systematic Biology*, 31(3):328–334, 1982. doi:10.1093/sysbio/31.3.328. Cited on page 27.

[88] William Feller. *Introduction to probability theory and its applications.* 1966. ISBN 978-0-471-25709-7. Cited on pages 48 and 116.

[89] Joseph Felsenstein. Evolutionary trees from DNA sequences: A maximum likelihood approach. *Journal of Molecular Evolution*, 17(6):368–376, Nov 1981. doi:10.1007/BF01734359. Cited on page 22.

[90] Joseph Felsenstein. Evolutionary trees from DNA sequences: A maximum likelihood approach. *Journal of Molecular Evolution*, 17(6):368–376, Nov 1981. doi:10.1007/BF01734359. Cited on page 25.

[91] Da-Fei Feng and Russell F. Doolittle. Progressive sequence alignment as a prerequisitetto correct phylogenetic trees. *Journal of Molecular Evolution*, 25(4):351–360, Aug 1987. doi:10.1007/BF02603120. Cited on page 25.

[92] Hans Fischer. *A history of the central limit theorem: from classical to modern probability theory.* Springer Science & Business Media, 2010. ISBN 978-0-387-87857-7. doi:10.1007/978-0-387-87857-7. Cited on page 70.

[93] Walter M. Fitch. Toward defining the course of evolution: Minimum change for a specific tree topology. *Systematic Zoology*, 20(4):406–416, 1971. doi:10.2307/2412116. Cited on page 25.

[94] Evelyn Fix and Joseph L. Hodges Jr. Discriminatory analysis-nonparametric discrimination: consistency properties. Technical report, University of California, Berkeley, 1951. URL `http://www.dtic.mil/dtic/tr/fulltext/u2/a800276.pdf`. Cited on page 112.

[95] Richard B. Flavell, Michael D. Bennett, J. B. Smith and Derek B. Smith. Genome size and the proportion of repeated nucleotide sequence DNA in plants. *Biochemical Genetics*, 12(4):257–269, Oct 1974. doi:10.1007/BF00485947. Cited on page 18.

[96] Edward B. Fowlkes and Colin L. Mallows. A method for comparing two hierarchical clusterings. *Journal of the American Statistical Association*, 78(383):553–569, 1983. doi:10.1080/01621459.1983.10478008. Cited on page 92.

[97] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55 (1):119–139, 1997. doi:10.1006/jcss.1997.1504. Cited on pages 70 and 136.

[98] Carl W. Fuller, Lyle R. Middendorf, Steven A. Benner, George M. Church, Timothy Harris et al. The challenges of sequencing by synthesis. *Nature Biotechnology*, 27(11): 1013–1023, Nov 2009. doi:10.1038/nbt.1585. Cited on page 17.

[99] P. Andrew Futreal, Lachlan Coin, Mhairi Marshall, Thomas Down, Timothy Hubbard et al. A census of human cancer genes. *Nature Reviews Cancer*, 4(3):177–183, Mar 2004. doi:10.1038/nrc1299. Cited on page 7.

[100] Najlah Gali, Radu Mariescu-Istodor and Pasi Fränti. Similarity measures for title matching. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 1548–1553, 2016. doi:10.1109/ICPR.2016.7899857. Cited on page 52.

[101] John Gallant, David Maier and James A. Storer. On finding minimal length superstrings. *Journal of Computer and System Sciences*, 20(1):50 – 58, 1980. doi:10.1016/0022-0000(80)90004-5. Cited on page 32.

[102] Genomics England. The 100,000 genomes project by numbers, December 2018. URL `https://www.genomicsengland.co.uk/the-100000-genomes-project-by-numbers/`. [Online; accessed 24-January-2022]. Cited on page 6.

[103] Suman Ghosal, Shaoli Das, Rituparno Sen, Piyali Basak and Jayprokas Chakrabarti. Circ2Traits: a comprehensive database for circular RNA potentially associated with disease and traits. *Frontiers in Genetics*, 4, 2013. doi:10.3389/fgene.2013.00283. Cited on page 131.

[104] Stephen K. Gire, Augustine Goba, Kristian G. Andersen, Rachel S. G. Sealfon, Daniel J. Park et al. Genomic surveillance elucidates Ebola virus origin and transmission during the 2014 outbreak. *Science*, 345(6202):1369–1372, 2014. doi:10.1126/science.1259657. Cited on page 7.

[105] Enrico Glaab, Anaïs Baudot, Natalio Krasnogor, Reinhard Schneider and Alfonso Valencia. EnrichNet: network-based gene set enrichment analysis. *Bioinformatics*, 28 (18):i451–i457, 09 2012. doi:10.1093/bioinformatics/bts389. Cited on pages 115 and 130.

[106] Petar Glažar, Panagiotis Papavasileiou and Nikolaus Rajewsky. circBase: a database for circular RNAs. *RNA*, 20(11):1666–1670, September 2014. Cited on page 124.

[107] Sara Goodwin, John Mcpherson and W. Richard Mccombie. Coming of age: Ten years of next-generation sequencing technologies. *Nature Reviews Genetics*, 17:333–351, 05 2016. doi:10.1038/nrg.2016.49. Cited on pages 2 and 5.

[108] Sam Griffiths-Jones, Russell J. Grocock, Stijn van Dongen, Alex Bateman and Anton J. Enright. miRBase: microRNA sequences, targets and gene nomenclature. *Nucleic Acids Research*, 34(suppl_1):D140–D144, 01 2006. doi:10.1093/nar/gkj112. Cited on page 120.

[109] gui11aume, qenvio and user24872. Distribution of the Levenshtein distance between two random strings. `https://stats.stackexchange.com/questions/136265/distribution-of-the-levenshtein-distance-between-two-random-strings`. Accessed: Friday 9th April, 2021. Cited on page 43.

[110] Giulia Guidi, Marquita Ellis, Daniel Rokhsar, Katherine Yelick and Aydın Buluç. Bella: Berkeley efficient long-read to long-read aligner and overlapper. *bioRxiv*, 2020. doi:10.1101/464420. Cited on page 32.

[111] Dan Gusfield. Efficient methods for multiple sequence alignment with guaranteed error bounds. *Bulletin of Mathematical Biology*, 55(1):141–154, 1993. doi:10.1016/S0092-8240(05)80066-7. Cited on page 24.

[112] Marc Haber, Claude Doumet-Serhal, Christiana Scheib, Yali Xue, Petr Danecek et al. Continuity and admixture in the last five millennia of levantine history from ancient canaanite and present-day lebanese genome sequences. *The American Journal of Human Genetics*, 101(2):274 – 282, 2017. doi:10.1016/j.ajhg.2017.06.013. Cited on page 7.

[113] James Hadfield, Colin Megill, Sidney M Bell, John Huddleston, Barney Potter et al. Nextstrain: real-time tracking of pathogen evolution. *Bioinformatics*, 34(23):4121–4123, 05 2018. doi:10.1093/bioinformatics/bty407. Cited on pages 7 and 8.

[114] B David Hames. *Gel electrophoresis of proteins: a practical approach*, volume 197. OUP Oxford, 1998. Cited on page 2.

[115] Richard W. Hamming. Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2):147–160, April 1950. doi:10.1002/j.1538-7305.1950.tb00463.x. Cited on pages 11, 23, 47, 54, and 57.

[116] Bernhard Haubold, Fabian Klötzl and Peter Pfaffelhuber. andi: Fast and accurate estimation of evolutionary distances between closely related genomes. *Bioinformatics*, 31 (8):1169–1175, 12 2014. doi:10.1093/bioinformatics/btu815. Cited on pages 28 and 31.

[117] Erika Check Hayden. The $1,000 genome. *Nature*, 507(7492):294, 2014. doi:10.1038/507294a. Cited on page 6.

[118] Roger W. Hendrix. Bacteriophages: Evolution of the majority. *Theoretical Population Biology*, 61(4):471–480, 2002. doi:10.1006/tpbi.2002.1590. Cited on page 112.

[119] M.D. Hendy and David Penny. Branch and bound algorithms to determine minimal evolutionary trees. *Mathematical Biosciences*, 59(2):277 – 290, 1982. doi:10.1016/0025-5564(82)90027-X. Cited on page 25.

[120] Steven Henikoff and Jorja G. Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Science*, 89:10915–10919, 11 1992. doi:10.1073/pnas.89.22.10915. Cited on page 22.

[121] Damayanthi Herath, Sen-Lin Tang, Kshitij Tandon, David Ackland and Saman Kumara Halgamuge. CoMet: a workflow using contig coverage and composition for binning a metagenomic sample with high precision. *BMC Bioinformatics*, 18(16):571, Dec 2017. doi:10.1186/s12859-017-1967-3. Cited on page 112.

[122] David Hernandez, Patrice François, Laurent Farinelli, Magne Østerås and Jacques Schrenzel. De novo bacterial genome sequencing: Millions of very short reads assembled on a desktop computer. *Genome Research*, 18(5):802–809, 2008. doi:10.1101/gr.072033.107. Cited on pages 32 and 91.

[123] Desmond G. Higgins and Paul M. Sharp. Clustal: a package for performing multiple sequence alignment on a microcomputer. *Gene*, 73(1):237–244, 1988. doi:10.1016/0378-1119(88)90330-7. Cited on page 24.

[124] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, pages 278–282 vol.1, 1995. doi:10.1109/ICDAR.1995.598994. Cited on page 70.

[125] Sebastian Horwege, Sebastian Lindner, Marcus Boden, Klas Hatje, Martin Kollmar et al. Spaced words and kmacs: fast alignment-free sequence comparison based on inexact word matches. *Nucleic Acids Research*, 42(W1):W7–W11, 2014. doi:10.1093/nar/gku398. Cited on page 30.

[126] Sheng-Da Hsu, Feng-Mao Lin, Wei-Yun Wu, Chao Liang, Wei-Chih Huang et al. miRTarBase: a database curates experimentally validated microRNA–target interactions. *Nucleic Acids Research*, 39(suppl_1):D163–D169, 11 2010. doi:10.1093/nar/gkq1107. Cited on page 120.

[127] Weichun Huang, Leping Li, Jason R. Myers and Gabor T. Marth. ART: a next-generation sequencing read simulator. *Bioinformatics*, 28(4):593–594, 2012. doi:10.1093/bioinformatics/btr708. Cited on page 93.

[128] Xiaoqiu Huang and Shiaw-Pyng Yang. Generating a genome assembly with PCAP. In *Current Protocols in Bioinformatics.* John Wiley & Sons, Inc., 2002. doi:10.1002/0471250953.bi1103s11. Cited on page 32.

[129] Zhou Huang, Jiangcheng Shi, Yuanxu Gao, Chunmei Cui, Shan Zhang et al. HMDD v3.0: a database for experimentally supported human microRNA–disease associations. *Nucleic Acids Research*, 47(D1):D1013–D1017, 10 2018. doi:10.1093/nar/gky1010. Cited on page 131.

[130] Tim Hubbard, Daniel Barker, Ewan Birney, Graham Cameron, Yuan Chen et al. The Ensembl genome database project. *Nucleic Acids Research*, 30(1):38–41, 2002. doi:10.1093/nar/30.1.38. Cited on page 93.

[131] William N. Hunter, Tom Brown, Naveen N. Anand and Olga Kennard. Structure of an adenine˙cytosine base pair in DNA and its implications for mismatch repair. *Nature*, 320 (6062):552–555, Apr 1986. doi:10.1038/320552a0. Cited on page 1.

[132] Illumina, Inc. Data sheet: Sequencing, October 2010. URL `https://www.illumina.com/documents/products/datasheets/datasheet_genomic_sequence.pdf`. [Online; accessed 20-November-2017]. Cited on page 15.

[133] Karel Jalovec. Assembly of discriminatory superstrings from sequencing data, June 2015. Cited on page 31.

[134] Karel Jalovec and Filip Železný. Binary classification of metagenomic samples using discriminative DNA superstrings. In *MLSB 2014: 8th International Workshop on Machine Learning in Systems Biology*, pages 44–47, 2014. Cited on page 31.

[135] Sabarinath Jayaseelan, Francis Doyle and Scott A. Tenenbaum. Profiling post-transcriptionally networked mRNA subsets using RIP-Chip and RIP-Seq. *Methods*, 67(1):13–19, 2014. doi:10.1016/j.ymeth.2013.11.001. Genomic approaches for studying transcriptional and post-transcriptional processes. Cited on page 6.

[136] William R. Jeck, Josephine A. Reinhardt, David A. Baltrus, Matthew T. Hickenbotham, Vincent Magrini et al. Extending assembly of short DNA sequences to handle error. *Bioinformatics*, 23(21):2942–2944, 2007. doi:10.1093/bioinformatics/btm451. Cited on page 32.

[137] Qinghua Jiang, Yadong Wang, Yangyang Hao, Liran Juan, Mingxiang Teng et al. miR2Disease: a manually curated database for microRNA deregulation in human disease. *Nucleic Acids Research*, 37(suppl_1):D98–D104, 10 2008. doi:10.1093/nar/gkn714. Cited on page 130.

[138] Sergio Jimenez, Claudia Becerra, Alexander Gelbukh and Fabio Gonzalez. Generalized Mongue-Elkan method for approximate text string comparison. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, pages 559–570, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-00382-0. Cited on page 53.

[139] Xiu juan Lei, Zengqiang Fang and Ling Guo. Predicting circRNA–disease associations based on improved collaboration filtering recommendation system with multiple data. *Frontiers in Genetics*, 10, 2019. doi:10.3389/fgene.2019.00897. Cited on page 131.

[140] Thomas H. Jukes and Charles R. Cantor. Evolution of protein molecules. In H.N. Munro, editor, *Mammalian Protein Metabolism*, pages 21 – 132. Academic Press, 1969. doi:10.1016/B978-1-4832-3211-9.50009-7. Cited on pages 22, 24, 27, 43, 44, and 45.

[141] Dieter Jungnickel. *Graphs, networks and algorithms*, volume 3. Springer, 2007. ISBN 978-3-642-09186-5. doi:10.1007/978-3-540-72780-4. URL `https://link.springer.com/book/10.1007/978-3-540-72780-4`. Cited on page 33.

[142] André Kahles, Cheng Soon Ong, Yi Zhong and Gunnar Rätsch. SplAdder: identification, quantification and testing of alternative splicing events from RNA-Seq data. *Bioinformatics*, 32(12):1840–1847, 02 2016. doi:10.1093/bioinformatics/btw076. Cited on page 4.

[143] Miriam R. Kantorovitz, Gene E. Robinson and Saurabh Sinha. A statistical method for alignment-free comparison of regulatory sequences. *Bioinformatics*, 23(13):i249–i255, 2007. doi:10.1093/bioinformatics/btm211. Cited on pages 27 and 28.

[144] Aleksandar Kaplar, Aleksandra Aleksić, Milan Stošović, Radomir Naumović, Voin Brković and Aleksandar Kovačević. Evaluating string distance metrics for approximate dictionary matching: A case study in serbian electronic health records, 2019. Cited on page 52.

[145] Dimitra Karagkouni, Maria D Paraskevopoulou, Serafeim Chatzopoulos, Ioannis S Vlachos, Spyros Tastsoglou et al. DIANA-TarBase v8: a decade-long collection of experimentally supported miRNA–gene interactions. *Nucleic Acids Research*, 46(D1): D239–D245, 11 2017. doi:10.1093/nar/gkx1141. Cited on page 120.

[146] Cihan Kaya, Princesca Dorsaint, Stephanie Mercurio, Alexander M. Campbell, Kenneth Wha Eng et al. Limitations of detecting genetic variants from the RNA sequencing data in tissue and fine-needle aspiration samples. *Thyroid*, 31(4):589–595, 2021. doi:10.1089/thy.2020.0307. PMID: 32948110. Cited on page 4.

[147] Mehdi Kchouk and Mourad Elloumi. A clustering approach for denovo assembly using next generation sequencing data. In *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 1909–1911. IEEE, Dec 2016. doi:10.1109/BIBM.2016.7822812. Cited on page 24.

[148] Ziqi Ke and Haris Vikalo. Graph-based reconstruction and analysis of disease transmission networks using viral genomic data. *bioRxiv*, 2022. doi:10.1101/2022.07.28.501873. Cited on pages 28 and 31.

[149] Andreas Keller, Christina Backes and Hans-Peter Lenhof. Computation of significance scores of unweighted gene set enrichment analyses. *BMC Bioinformatics*, 8(1):290, Aug 2007. doi:10.1186/1471-2105-8-290. Cited on page 118.

[150] Young-Ji Kim, Kun-Ho Seo, Seolhui Kim and Songmee Bae. Phylogenetic comparison and characterization of an mcr-1-harboring complete plasmid genome isolated from enterobacteriaceae. *Microbial Drug Resistance*, 28(4):492–497, 2022. Cited on page 31.

[151] Motoo Kimura. A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *Journal of Molecular Evolution*, 16 (2):111–120, Jun 1980. doi:10.1007/BF01731581. Cited on page 22.

[152] Anastasiya D. Kirichenko, Anastasiya A. Poroshina, Dmitry Yu. Sherbakov, Michael G. Sadovsky and Konstantin V. Krutovsky. Comparative analysis of alignment-free genome clustering and whole genome alignment-based phylogenomic relationship of coronaviruses. *PLOS ONE*, 17(3):1–26, 03 2022. doi:10.1371/journal.pone.0264640. Cited on page 31.

[153] Jon Kleinberg and Eva Tardos. *Algorithm Design.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005. ISBN 0321295358. Cited on pages 35 and 77.

[154] Eugene V. Koonin. Orthologs, paralogs, and evolutionary genomics. *Annual Review of Genetics*, 39(1):309–338, 2005. doi:10.1146/annurev.genet.39.073003.114725. PMID: 16285863. Cited on page 18.

[155] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79 – 86, 1951. doi:10.1214/aoms/1177729694. Cited on page 55.

[156] Comin Lab. C2q, 2021. URL `https://github.com/CominLab/C2Q`. Cited on page 91.

[157] Klára Labská, Michaela Špačková, Ondřej Daniel, Josef Včelák, Veronika Vlasáková et al. A cross-border outbreak of salmonella bareilly cases confirmed by whole genome sequencing, Czech republic and Slovakia, 2017 to 2018. *Eurosurveillance*, 26(14):2000131, 2021. doi:10.2807/1560-7917.ES.2021.26.14.2000131. Cited on page 7.

[158] Wei Lan, Mingrui Zhu, Qingfeng Chen, Baoshan Chen, Jin Liu et al. CircR2Cancer: a manually curated database of associations between circRNAs and cancers. *Database*, 2020, 11 2020. doi:10.1093/database/baaa085. baaa085. Cited on page 131.

[159] Eric S. Lander, Lauren M. Linton, Bruce Birren, Chad Nusbaum, Michael C. Zody et al. Initial sequencing and analysis of the human genome. *Nature*, 409(6822):860–921, 2001. doi:10.1038/35057062. Cited on page 6.

[160] Hayan Lee, James Gurtowski, Shinjae Yoo, Maria Nattestad, Shoshana Marcus et al. Third-generation sequencing and the future of genomics. *bioRxiv*, 2016. doi:10.1101/048603. Cited on page 4.

[161] Xiujuan Lei and Chen Bian. Integrating random walk with restart and k-nearest neighbor to identify novel circRNA-disease association. *Scientific Reports*, 10(1):1943, Feb 2020. doi:10.1038/s41598-020-59040-0. Cited on pages 115 and 130.

[162] Xiujuan Lei, Zengqiang Fang, Luonan Chen and Fang-Xiang Wu. PWCDA: Path weighted method for predicting circRNA-disease associations. *International Journal of Molecular Sciences*, 19(11):3410, Oct 2018. doi:10.3390/ijms19113410. Cited on page 131.

[163] Chris-André Leimeister. A novel pseudo-alignment approach to fast genomic sequence comparison. Master's thesis, Institut für Mikrobiologie und Genetik, Abteilung für Bioinformatik, 2015. URL `https://webhelper.informatik.uni-goettingen.de/editor/media/theses/ZAI-MSC-2015-19-leimeister.pdf`. Cited on page 30.

[164] Chris-Andre Leimeister, Marcus Boden, Sebastian Horwege, Sebastian Lindner and Burkhard Morgenstern. Fast alignment-free sequence comparison using spaced-word frequencies. *Bioinformatics*, 30(14):1991–1999, 2014. doi:10.1093/bioinformatics/btu177. Cited on page 30.

[165] Chris-André Leimeister, Salma Sohrabi-Jahromi and Burkhard Morgenstern. Fast and accurate phylogeny reconstruction using filtered spaced-word matches. *Bioinformatics*, 33(7):971–979, 2017. doi:10.1093/bioinformatics/btw776. Cited on pages 28 and 30.

[166] Rasko Leinonen, Ruth Akhtar, Ewan Birney, Lawrence Bower, Ana Cerdeno-Tárraga et al. The European Nucleotide Archive. *Nucleic Acids Research*, 39(suppl_1):D28–D31, 2011. doi:10.1093/nar/gkq967. Cited on pages 6, 69, 93, and 107.

[167] Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics doklady*, 10(8):707, 1966. Cited on pages 10, 21, 47, 54, 56, 59, 87, and 135.

[168] Elizaveta Levina and Peter Bickel. The earth mover's distance is the Mallows distance: Some insights from statistics. volume 2, pages 251 – 256 vol.2, 02 2001. ISBN 0-7695-1143-0. doi:10.1109/ICCV.2001.937632. Cited on page 31.

[169] Harris A. Lewin, Gene E. Robinson, W. John Kress, William J. Baker, Jonathan Coddington et al. Earth biogenome project: Sequencing life for the future of life. *Proceedings of the National Academy of Sciences*, 115(17):4325–4333, 2018. doi:10.1073/pnas.1720115115. Cited on page 6.

[170] Benjamin P. Lewis, Christopher B. Burge and David P. Bartel. Conserved seed pairing, often flanked by adenosines, indicates that thousands of human genes are MicroRNA targets. *Cell*, 120(1):15–20, Jan 2005. doi:10.1016/j.cell.2004.12.035. Cited on page 120.

[171] Guanghui Li, Yingjie Yue, Cheng Liang, Qiu Xiao, Pingjian Ding and Jiawei Luo. NCPCDA: network consistency projection for circRNA–disease association prediction. *RSC advances*, 9(57):33222–33228, 2019. doi:10.1039/C9RA06133A. Cited on page 131.

[172] Guanghui Li, Jiawei Luo, Diancheng Wang, Cheng Liang, Qiu Xiao et al. Potential circRNA-disease association prediction using DeepWalk and network consistency projection. *Journal of Biomedical Informatics*, 112:103624, 2020. doi:10.1016/j.jbi.2020.103624. Cited on page 131.

[173] Ruiqiang Li, Hongmei Zhu, Jue Ruan, Wubin Qian, Xiaodong Fang et al. De novo assembly of human genomes with massively parallel short read sequencing. *Genome Research*, 20(2):265–272, 2010. doi:10.1101/gr.097261.109. Cited on page 33.

[174] Yongping Li, Cheng Dai, Chungen Hu, Zhongchi Liu and Chunying Kang. Global identification of alternative splicing via comparative analysis of SMRT- and Illumina-based RNA-seq in strawberry. *The Plant Journal*, 90(1):164–176, 2017. doi:10.1111/tpj.13462. Cited on page 4.

[175] Yongsheng Li, Juan Xu, Tingting Shao, Yunpeng Zhang, Hong Chen and Xia Li. *Functional Genomics: Methods and Protocols*, chapter RNA Function Prediction, pages 17–28. Springer New York, New York, NY, 2017. ISBN 978-1-4939-7231-9. doi:10.1007/978-1-4939-7231-9_2. Cited on page 114.

[176] Arthur Liberzon, Aravind Subramanian, Reid Pinchback, Helga Thorvaldsdóttir, Pablo Tamayo and Jill P. Mesirov. Molecular signatures database (MSigDB) 3.0. *Bioinformatics*, 27(12):1739–1740, 05 2011. doi:10.1093/bioinformatics/btr260. Cited on page 120.

[177] Daniel Lichtblau. Alignment-free genomic sequence comparison using FCGR and signal processing. *BMC Bioinformatics*, 20(1):742, Dec 2019. doi:10.1186/s12859-019-3330-3. Cited on pages 28 and 31.

[178] J. Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, 1991. doi:10.1109/18.61115. Cited on page 30.

[179] Shimei Liu, Bingqing Li, Ying Li and Huaihua Song. Circular RNA circ_0000228 promotes the malignancy of cervical cancer via miRNA-195-5p/ lysyl oxidase-like protein 2 axis. *Bioengineered*, 12(1):4397–4406, 2021. doi:10.1080/21655979.2021.1954846. PMID: 34308761. Cited on page 124.

[180] Rosalyn Lo, Katherine E Dougan, Yibi Chen, Sarah Shah, Debashish Bhattacharya and Cheong Xin Chan. Alignment-Free analysis of Whole-Genome sequences from symbiodiniaceae reveals different phylogenetic signals in distinct regions. *Front Plant Sci*, 13:815714, April 2022. Cited on page 31.

[181] Daniel Lopresti and Andrew Tomkins. Block edit models for approximate string matching. *Theoretical Computer Science*, 181(1):159 – 179, 1997. doi:10.1016/S0304-3975(96)00268-X. Cited on page 11.

[182] Michael I. Love, Wolfgang Huber and Simon Anders. Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2. *Genome Biology*, 15(12):550, Dec 2014. doi:10.1186/s13059-014-0550-8. Cited on page 4.

[183] Chengqian Lu, Min Zeng, Fang-Xiang Wu, Min Li and Jianxin Wang. Improving circRNA–disease association prediction by sequence and ontology representations with convolutional and recurrent neural networks. *Bioinformatics*, 36(24):5656–5664, 12 2020. doi:10.1093/bioinformatics/btaa1077. Cited on page 131.

[184] Chengwei Luo, Despina Tsementzi, Nikos Kyrpides, Timothy Read and Konstantinos T. Konstantinidis. Direct comparisons of Illumina vs. Roche 454 sequencing technologies on the same microbial community DNA sample. *PLOS ONE*, 7(2):1–12, 02 2012. doi:10.1371/journal.pone.0030087. Cited on page 3.

[185] David Maier and James A. Storer. A note on the complexity of the superstring problem. Technical Report 233, Dept. of Electrical Engineering and Computer Science, Priceton University, 1977. Cited on pages 15 and 32.

[186] Satya N. Majumdar and Sergei Nechaev. Exact asymptotic results for the bernoulli matching model of sequence alignment. *Phys. Rev. E*, 72:020901, Aug 2005. doi:10.1103/PhysRevE.72.020901. Cited on page 43.

[187] Raunaq Malhotra, Daniel Elleder, Le Bao, David R. Hunter, Raj Acharya and Mary Poss. Clustering pipeline for determining consensus sequences in targeted next-generation sequencing. *ArXiv preprint*, October 2014. Cited on page 24.

[188] Bryan Manly and Jorge Navarro Alberto. *Randomization, Bootstrap and Monte Carlo Methods in Biology (4th ed.)*. Chapman and Hall/CRC, 07 2020. ISBN 9780429329203. doi:10.1201/9780429329203. Cited on page 116.

[189] Laura D Manzanares-Meza and Oscar Medina-Contreras. SARS-CoV-2 and influenza: a comparative overview and treatment implications. *Boletín médico del Hospital Infantil de México*, 77(5):262–273, 2020. doi:10.24875/BMHIM.20000183. Cited on page 111.

[190] Daniel Mapleson, Gonzalo Garcia Accinelli, George Kettleborough, Jonathan Wright and Bernardo J Clavijo. KAT: a K-mer analysis toolkit to quality control NGS datasets and genome assemblies. *Bioinformatics*, 33(4):574–576, 11 2016. doi:10.1093/bioinformatics/btw663. Cited on page 14.

[191] P. Markoulatos, N. Siafakas and M. Moncany. Multiplex polymerase chain reaction: A practical approach. *Journal of Clinical Laboratory Analysis*, 16(1):47–51, 2002. doi:10.1002/jcla.2058. Cited on page 6.

[192] F. H. C. Marriott. Barnard's Monte Carlo tests: How many simulations? *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):75–77, 1979. Cited on pages 113 and 133.

[193] Andres Marzal and Enrique Vidal. Computation of normalized edit distance and applications. *IEEE transactions on pattern analysis and machine intelligence*, 15(9): 926–932, Sep 1993. doi:10.1109/34.232078. Cited on page 11.

[194] William J. Masek and Michael S. Paterson. A faster algorithm computing string edit distances. *Journal of Computer and System Sciences*, 20(1):18 – 31, 1980. doi:10.1016/0022-0000(80)90002-1. Cited on page 22.

[195] Sebastian Maurer-Stroh, Vithiagaran Gunalan, Wing-Cheong Wong and Frank Eisenhaber. A simple shortcut to unsupervised alignment-free phylogenetic genome groupings, even from unassembled sequencing reads. *Journal of Bioinformatics and Computational Biology*, 11(06):1343005, 2013. doi:10.1142/S0219720013430051. PMID: 24372034. Cited on pages 28 and 30.

[196] Andrew Melnyk, Sergey Knyazev, Fredrik Vannberg, Leonid Bunimovich, Pavel Skums and Alex Zelikovsky. Using earth mover's distance for viral outbreak investigations. *BMC Genomics*, 21(5):582, Dec 2020. doi:10.1186/s12864-020-06982-4. Cited on pages 28, 31, and 111.

[197] Daniele Merico, Ruth Isserlin, Oliver Stueker, Andrew Emili and Gary D. Bader. Enrichment map: A network-based method for gene-set enrichment visualization and interpretation. *PLOS ONE*, 5(11):1–12, 11 2010. doi:10.1371/journal.pone.0013984. Cited on page 122.

[198] Jason R. Miller, Sergey Koren and Granger Sutton. Assembly algorithms for next-generation sequencing data. *Genomics*, 95(6):315–327, 2010. doi:10.1016/j.ygeno.2010.03.001. Cited on page 33.

[199] Jason R. Miller, Sergey Koren and Granger Sutton. Assembly algorithms for next-generation sequencing data. *Genomics*, 95(6):315 – 327, 2010. doi:10.1016/j.ygeno.2010.03.001. Cited on page 32.

[200] Alvaro E. Monge and Charles P. Elkan. The field matching problem: Algorithms and applications. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 267–270. AAAI Press, 1996. Cited on pages 11, 12, 37, 52, 59, 64, and 81.

[201] Burkhard Morgenstern, Bingyao Zhu, Sebastian Horwege and Chris André Leimeister. Estimating evolutionary distances between genomic sequences from spaced-word matches. *Algorithms for Molecular Biology*, 10(1):5, Feb 2015. doi:10.1186/s13015-015-0032-x. Cited on page 30.

[202] Donald R. Morrison. PATRICIA — practical algorithm to retrieve information coded in alphanumeric. *Journal of the ACM*, 15(4):514–534, October 1968. doi:10.1145/321479.321481. Cited on page 24.

[203] Kary B. Mullis. The unusual origin of the polymerase chain reaction. *Scientific American*, 262(4):56–65, 1990. Cited on page 4.

[204] Kevin D. Murray, Christfried Webers, Cheng Soon Ong, Justin Borevitz and Norman Warthmann. kwip: The *k*-mer weighted inner product, a de novo estimator of genetic similarity. *PLOS Computational Biology*, 13(9):1–17, 09 2017. doi:10.1371/journal.pcbi.1005727. Cited on pages 28 and 31.

[205] Eugene W. Myers, Granger G. Sutton, Art L. Delcher, Ian M. Dew, Dan P. Fasulo et al. A whole-genome assembly of drosophila. *Science*, 287(5461):2196–2204, 2000. doi:10.1126/science.287.5461.2196. Cited on page 32.

[206] Darren A. Natale, Cecilia N. Arighi, Judith A. Blake, Jonathan Bona, Chuming Chen et al. Protein Ontology (PRO): enhancing and scaling up the representation of protein entities. *Nucleic Acids Research*, 45(D1):D339–D346, 11 2016. doi:10.1093/nar/gkw1075. Cited on page 7.

[207] Gonzalo Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88, March 2001. doi:10.1145/375360.375365. Cited on pages 12, 13, 23, 85, and 87.

[208] Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453, 1970. doi:10.1016/0022-2836(70)90057-4. Cited on page 21.

[209] Ngoc G Nguyen, Vu Anh Tran, Dau Phan, Favorisen R Lumbanraja, Mohammad Reza Faisal et al. DNA sequence classification by convolutional neural network. *Journal Biomedical Science and Engineering*, 9(5):280–286, 2016. Cited on page 112.

[210] Sergey Nurk, Anton Bankevich et al. Assembling genomes and mini-metagenomes from highly chimeric reads. In Minghua Deng, Rui Jiang, Fengzhu Sun and Xuegong Zhang, editors, *Research in Computational Molecular Biology: 17th Annual International Conference, RECOMB 2013, Beijing, China, April 7-10, 2013. Proceedings*, pages 158–170. Springer Berlin Heidelberg, 2013. doi:10.1007/978-3-642-37195-0_13. Cited on pages 33, 91, and 109.

[211] Stephen J. O'Brien, William G. Nash, David E. Wildt, Mitchell E. Bush and Raoul E. Benveniste. A molecular solution to the riddle of the giant panda's phylogeny. *Nature*, 317(6033):140–144, 1985. doi:10.1038/317140a0. Cited on page 7.

[212] Stephen Oliver. Guilt-by-association goes global. *Nature*, 403(6770):601–602, Feb 2000. doi:10.1038/35001165. Cited on page 114.

[213] Brian D. Ondov, Todd J. Treangen, Páll Melsted, Adam B. Mallonee, Nicholas H. Bergman et al. Mash: fast genome and metagenome distance estimation using MinHash. *Genome Biology*, 17(1):132, Jun 2016. doi:10.1186/s13059-016-0997-x. Cited on pages 28, 30, and 91.

[214] Rafail Ostrovsky and Yuval Rabani. Low distortion embeddings for edit distance. *Journal of the ACM*, 54(5), October 2007. doi:10.1145/1284320.1284322. Cited on pages 23 and 136.

[215] Amaresh Chandra Panda. *Circular RNAs: Biogenesis and Functions*, chapter Circular RNAs Act as miRNA Sponges, pages 67–79. Springer Singapore, Singapore, 2018. ISBN 978-981-13-1426-1. doi:10.1007/978-981-13-1426-1_6. URL `https://doi.org/10.1007/978-981-13-1426-1_6`. Cited on page 1.

[216] Vilfredo Pareto. *Cours d'économie politique*, volume 1. Librairie Droz, 1964. Cited on page 126.

[217] Peter J. Park. ChIP–seq: advantages and challenges of a maturing technology. *Nature Reviews Genetics*, 10(10):669–680, Oct 2009. doi:10.1038/nrg2641. Cited on page 6.

[218] Rob Patro, Geet Duggal, Michael I. Love, Rafael A. Irizarry and Carl Kingsford. Salmon provides fast and bias-aware quantification of transcript expression. *Nature Methods*, 14 (4):417–419, Apr 2017. doi:10.1038/nmeth.4197. Cited on page 31.

[219] Bradley A. Perkins, C. Thomas Caskey, Pamila Brar, Eric Dec, David S. Karow et al. Precision medicine screening using whole-genome sequencing and advanced imaging to identify disease risk in adults. *Proceedings of the National Academy of Sciences*, 115(14): 3686–3691, 2018. doi:10.1073/pnas.1706096114. Cited on page 2.

[220] Pavel Pevzner and Glenn Tesler. Genome rearrangements in mammalian evolution: lessons from human and mouse genomes. *Genome research*, 13(1):37–45, 2003. doi:10.1101/gr.757503. Cited on page 111.

[221] Pavel A. Pevzner and Haixu Tang. Fragment assembly with double-barreled data. *Bioinformatics*, 17(suppl_1):S225–S233, 2001. doi:10.1093/bioinformatics/17.suppl_1.S225. Cited on page 15.

[222] Pavel A. Pevzner, Haixu Tang and Michael S. Waterman. An eulerian path approach to DNA fragment assembly. *Proceedings of the National Academy of Sciences*, 98(17): 9748–9753, 2001. doi:10.1073/pnas.171285098. Cited on page 33.

[223] Belinda Phipson and Gordon K Smyth. Permutation p-values should never be zero: Calculating exact p-values when permutations are randomly drawn. *Statistical Applications in Genetics and Molecular Biology*, 9(1), 2010. doi:10.2202/1544-6115.1585. Cited on pages 118, 122, and 126.

[224] Janet Piñero, Núria Queralt-Rosinach, Alex Bravo, Jordi Deu-Pons, Anna Bauer-Mehren et al. DisGeNET: a discovery platform for the dynamical exploration of human diseases and their genes. *Database*, 2015, 04 2015. doi:10.1093/database/bav028. bav028. Cited on page 131.

[225] Maria S. Poptsova, Irina A. Il'icheva, Dmitry Yu. Nechipurenko, Larisa A. Panchenko, Mingian V. Khodikov et al. Non-random DNA fragmentation in next-generation sequencing. *Scientific Reports*, 4(1):4532, Mar 2014. doi:10.1038/srep04532. Cited on page 17.

[226] Ji Qi, Hong Luo and Bailin Hao. CVTree: a phylogenetic tree reconstruction tool based on whole genomes. *Nucleic Acids Research*, 32(suppl_2):W45–W47, 2004. doi:10.1093/nar/gkh362. Cited on pages 28 and 29.

[227] Ji Qi, Bin Wang and Bai-Iin Hao. Whole proteome prokaryote phylogeny without sequence alignment: A k-string composition approach. *Journal of Molecular Evolution*, 58(1):1–11, Jan 2004. doi:10.1007/s00239-003-2493-7. Cited on page 29.

[228] Emma M. Quinn, Paul Cormican, Elaine M. Kenny, Matthew Hill, Richard Anney et al. Development of strategies for SNP detection in RNA-seq data: Application to lymphoblastoid cell lines and evaluation using 1000 genomes data. *PLOS ONE*, 8(3):1–9, 03 2013. doi:10.1371/journal.pone.0058815. Cited on page 4.

[229] Rupesh Kanchi Ravi, Kendra Walton and Mahdieh Khosroheidari. *MiSeq: A Next Generation Sequencing Platform for Genomic Analysis*, pages 223–232. Springer New York, New York, NY, 2018. ISBN 978-1-4939-7471-9. doi:10.1007/978-1-4939-7471-9_12. Cited on pages 31 and 111.

[230] Jüri Reimand, Ruth Isserlin, Veronique Voisin, Mike Kucera, Christian Tannus-Lopes et al. Pathway enrichment analysis and visualization of omics data using g:Profiler, GSEA, Cytoscape and EnrichmentMap. *Nature Protocols*, 14(2):482–517, Feb 2019. doi:10.1038/s41596-018-0103-9. Cited on page 122.

[231] Gesine Reinert, David Chew, Fengzhu Sun and Michael S. Waterman. Alignment-free sequence comparison (I): statistics and power. *Journal of Computational Biology*, 16(12): 1615–1634, 2009. doi:10.1089/cmb.2009.0198. Cited on pages 27, 28, 29, and 91.

[232] Noa Rivlin, Ran Brosh, Moshe Oren and Varda Rotter. Mutations in the p53 tumor suppressor gene: Important milestones at the various steps of tumorigenesis. *Genes & Cancer*, 2(4):466–474, 2011. doi:10.1177/1947601911408889. PMID: 21779514. Cited on page 7.

[233] Michael G. Ross, Carsten Russ, Maura Costello, Andrew Hollinger, Niall J. Lennon et al. Characterizing and measuring bias in sequence data. *Genome biology*, 14(5):R51, 2013. doi:10.1186/gb-2013-14-5-r51. Cited on pages 14 and 17.

[234] Tanmoy Roychowdhury, Anchal Vishnoi and Alok Bhattacharya. Next-generation anchor based phylogeny (NexABP): constructing phylogeny from next-generation sequencing data. *Scientific reports*, 3, 2013. doi:10.1038/srep02634. Cited on pages 28 and 30.

[235] Yuanbin Ru, Katerina J. Kechris, Boris Tabakoff, Paula Hoffman, Richard A. Radcliffe et al. The multiMiR R package and database: integration of microRNA–target interactions along with their disease and drug associations. *Nucleic Acids Research*, 42 (17):e133, 2014. doi:10.1093/nar/gku631. Cited on page 120.

[236] Stuart Russell and Peter Norvig. Artificial intelligence: a modern approach, global edition 4th. *Foundations*, 19:23, 2021. Cited on page 44.

[237] Petr Ryšavý and Filip Železný. Estimating sequence similarity from read sets for clustering sequencing data. In Henrik Boström, Arno Knobbe, Carlos Soares and Panagiotis Papapetrou, editors, *Advances in Intelligent Data Analysis XV: 15th International Symposium, IDA 2016, Stockholm, Sweden, October 13-15, 2016, Proceedings*, pages 204–214. Springer International Publishing, 2016. doi:10.1007/978-3-319-46349-0_18. Cited on pages 37, 53, 79, 80, 81, 91, 92, and 165.

[238] Petr Ryšavý and Filip Železný. Estimating sequence similarity from contig sets. In Niall Adams, Allan Tucker and David Weston, editors, *Advances in Intelligent Data Analysis XVI: 16th International Symposium, IDA 2017, London, UK, October 26–28, 2017, Proceedings*, pages 272–283, Cham, 2017. Springer International Publishing. ISBN 978-3-319-68765-0. doi:10.1007/978-3-319-68765-0_23. Cited on pages 73, 76, 77, 79, 80, 91, 92, and 165.

[239] Petr Ryšavý and Filip Železný. Estimating sequence similarity from read sets for clustering next-generation sequencing data. *Data Mining and Knowledge Discovery*, 33 (1):1–23, Jan 2019. doi:10.1007/s10618-018-0584-8. Cited on pages 13, 37, 38, 39, 40, 41, 53, 80, 85, 91, 92, and 165.

[240] Petr Ryšavý and Filip Železný. Reference-free phylogeny from sequencing data. Accepted to BMC BioData Mining, in publication. Cited on pages 35, 79, 81, 82, 86, 91, 92, 104, 111, and 166.

[241] Petr Ryšavý and Filip Železný. An algorithm to calculate the p-value of the Monge-Elkan distance. Accepted to ISBRA 2022. Cited on pages 46, 47, 48, 49, 52, 53, 54, and 166.

[242] Petr Ryšavý, Jiří Kléma and Michaela Dostálová Merkerová. circGPA: circRNA functional annotation based on probability-generating functions. *BMC Bioinformatics*, 23(1):392, Sep 2022. doi:10.1186/s12859-022-04957-8. Cited on pages 113, 114, 115, 117, 118, 120, 122, 130, 131, and 165.

[243] Petr Ryšavý. Using tries for evaluating monge-elkan distance on genomic sequences. In Libor Husník, editor, *Proceedings of the International Student Scientific Conference Poster – 21/2017*. Czech Technical University in Prague, 2017. ISBN 978-80-01-06153-4. Cited on pages 24, 87, 88, 91, 107, and 165.

[244] Petr Ryšavý. On sequence overlaps minimizing post-normalized edit distance. In Libor Husník, editor, *Proceedings of the International Student Scientific Conference Poster – 22/2018*. Czech Technical University in Prague, 2018. ISBN 978-80-01-06428-3. Cited on pages 76, 91, and 165.

[245] Petr Ryšavý. Approximate search in genomic data. In Libor Husník, editor, *Proceedings of the International Student Scientific Conference Poster – 23/2019*. Czech Technical University in Prague, 2019. ISBN 978-80-01-06581-5. Cited on pages 59, 60, 67, 69, 91, and 166.

[246] Petr Ryšavý and Greg Hamerly. Geometric methods to accelerate k-means algorithms. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pages 324–332, May 2016. doi:10.1137/1.9781611974348.37. Cited on page 166.

[247] Gustavo AT Sacomoto, Janice Kielbassa, Rayan Chikhi, Raluca Uricaru, Pavlos Antoniou et al. Kis splice: de-novo calling alternative splicing events from RNA-seq data. *BMC Bioinformatics*, 13(6):S5, Apr 2012. doi:10.1186/1471-2105-13-S6-S5. Cited on page 4.

[248] Randall K. Saiki, Stephen Scharf, Fred Faloona, Kary B. Mullis, Glenn T. Horn et al. Enzymatic amplification of β-globin genomic sequences and restriction site analysis for diagnosis of sickle cell anemia. *Science*, 230(4732):1350–1354, 1985. doi:10.1126/science.2999980. Cited on page 4.

[249] Naruya Saitou and Masatoshi Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4(4):406–425, 1987. doi:10.1093/oxfordjournals.molbev.a040454. Cited on pages 7, 18, 24, 26, and 92.

[250] Frederick Sanger and Hans Tuppy. The amino-acid sequence in the phenylalanyl chain of insulin. 1. the identification of lower peptides from partial hydrolysates. *Biochemical Journal*, 49(4):463–481, 1951. doi:10.1042/bj0490463. Cited on page 2.

[251] Frederick Sanger, S. Nicklen and A. R. Coulson. DNA sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Sciences*, 74(12):5463–5467, 1977. doi:10.1073/pnas.74.12.5463. Cited on page 2.

[252] David Sankoff. Minimal mutation trees of sequences. *SIAM Journal on Applied Mathematics*, 28(1):35–42, 1975. doi:10.1137/0128004. Cited on page 25.

[253] Rui Santos, Patricia Murrieta-Flores and Bruno Martins. Learning to combine multiple string similarity metrics for effective toponym matching. *International Journal of Digital Earth*, 11(9):913–938, 2018. doi:10.1080/17538947.2017.1371253. Cited on page 52.

[254] Ajay Kumar Saw, Garima Raj, Manashi Das, Narayan Chandra Talukdar, Binod Chandra Tripathy and Soumyadeep Nandi. Alignment-free method for DNA sequence clustering using fuzzy integral similarity. *Scientific Reports*, 9(1):3753, Mar 2019. doi:10.1038/s41598-019-40452-6. Cited on pages 28 and 31.

[255] Eric E. Schadt, Steve Turner and Andrew Kasarskis. A window into third-generation sequencing. *Human Molecular Genetics*, 19(R2):R227–R240, 09 2010. doi:10.1093/hmg/ddq416. Cited on page 4.

[256] Karel Sedlar, Kristyna Kupkova and Ivo Provaznik. Bioinformatics strategies for taxonomy independent binning and visualization of sequences in shotgun metagenomics. *Computational and Structural Biotechnology Journal*, 15:48–55, 2017. doi:10.1016/j.csbj.2016.11.005. Cited on page 112.

[257] Peter H Sellers. The theory and computation of evolutionary distances: Pattern recognition. *Journal of Algorithms*, 1(4):359 – 373, 1980. doi:10.1016/0196-6774(80)90016-4. Cited on page 59.

[258] Heping Shang and Tim H. Merrettal. Tries for approximate string matching. *IEEE Transactions on Knowledge and Data Engineering*, 8(4):540–547, Aug 1996. doi:10.1109/69.536247. Cited on pages 24 and 87.

[259] Jingbo Shang, Jian Peng and Jiawei Han. MACFP: Maximal approximate consecutive frequent pattern mining under edit distance. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pages 558–566. Society for Industrial and Applied Mathematics, 2016. doi:10.1137/1.9781611974348.63. Cited on page 136.

[260] Paul Shannon, Andrew Markiel, Owen Ozier, Nitin S. Baliga, Jonathan T. Wang et al. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome research*, 13(11):2498–2504, Nov 2003. doi:10.1101/gr.1239303. Cited on page 122.

[261] Yuelong Shu and John McCauley. GISAID: Global initiative on sharing all influenza data – from vision to reality. *Eurosurveillance*, 22(13):30494, 2017. doi:10.2807/1560-7917.ES.2017.22.13.30494. Cited on page 7.

[262] Weiva Sieh, Joseph H. Rothstein, Valerie McGuire and Alice S. Whittemore. The role of genome sequencing in personalized breast cancer prevention. *Cancer Epidemiology and Prevention Biomarkers*, 23(11):2322–2327, 2014. doi:10.1158/1055-9965.EPI-14-0559. Cited on page 2.

[263] Marina Siljic, Dubravka Salemovic, Djordje Jevtovic, Ivana Pesic-Pavlovic, Sonja Zerjav et al. Forensic application of phylogenetic analysis – exploration of suspected epidemiological linkage. *BMC Infectious Diseases*, 14(4):O21, May 2014. doi:10.1186/1471-2334-14-S4-O21. Cited on page 7.

[264] I. Silva, R. Assunção and M. Costa. Power of the sequential Monte Carlo test. *Sequential Analysis*, 28(2):163–174, 2009. doi:10.1080/07474940902816601. Cited on page 132.

[265] Ivair R. Silva and Renato M. Assunção. Optimal generalized truncated sequential monte carlo test. *Journal of Multivariate Analysis*, 121:33–49, 2013. doi:10.1016/j.jmva.2013.06.003. Cited on page 132.

[266] 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, 2009. doi:10.1101/gr.089532.108. Cited on pages 33 and 91.

[267] David Sims, Ian Sudbery, Nicholas E. Ilott, Andreas Heger and Chris P. Ponting. Sequencing depth and coverage: key considerations in genomic analyses. *Nature Reviews Genetics*, 15(2):121–132, Feb 2014. doi:10.1038/nrg3642. Cited on page 14.

[268] Temple F. Smith and Michael S. Waterman. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981. doi:10.1016/0022-2836(81)90087-5. Cited on pages 21, 59, and 74.

[269] Robert R. Sokal and Charles D. Michener. A statistical method for evaluating systematic relationships. *University of Kansas Science Bulletin*, 38:1409–1438, 1958. Cited on pages 7, 18, 24, 25, and 92.

[270] Kai Song, Jie Ren, Zhiyuan Zhai, Xuemei Liu, Minghua Deng and Fengzhu Sun. Alignment-free sequence comparison based on next generation sequencing reads: Extended abstract. In Benny Chor, editor, *Research in Computational Molecular Biology: 16th Annual International Conference, RECOMB 2012, Barcelona, Spain, April 21-24, 2012. Proceedings*, pages 272–285. Springer Berlin Heidelberg, 2012. doi:10.1007/978-3-642-29627-7_29. Cited on page 29.

[271] Kai Song, Jie Ren, Zhiyuan Zhai, Xuemei Liu, Minghua Deng and Fengzhu Sun. Alignment-free sequence comparison based on next-generation sequencing reads. *Journal of computational biology*, 20(2):64–79, 2013. doi:10.1089/cmb.2012.0228. Cited on pages 28, 29, and 91.

[272] Kai Song, Jie Ren, Gesine Reinert, Minghua Deng, Michael S. Waterman and Fengzhu Sun. New developments of alignment-free sequence comparison: measures, statistics and next-generation sequencing. *Briefings in Bioinformatics*, 15(3):343–353, 2014. doi:10.1093/bib/bbt067. Cited on page 29.

[273] Min Song and Alex Rudniy. Detecting duplicate biological entities using markov random field-based edit distance. *Knowledge and Information Systems*, 25(2):371–387, Nov 2010. doi:10.1007/s10115-009-0254-7. Cited on page 53.

[274] R. Staden. A strategy of DNA sequencing employing computer programs. *Nucleic Acids Research*, 6(7):2601–2610, 1979. doi:10.1093/nar/6.7.2601. Cited on page 2.

[275] Rodger Staden. A mew computer method for the storage and manipulation of DNA gel reading data. *Nucleic Acids Research*, 8(16):3673–3694, 1980. doi:10.1093/nar/8.16.3673. Cited on page 15.

[276] Zachary D. Stephens, Skylar Y. Lee, Faraz Faghri, Roy H. Campbell, Chengxiang Zhai et al. Big data: Astronomical or genomical? *PLOS Biology*, 13(7):1–11, 07 2015. doi:10.1371/journal.pbio.1002195. Cited on page 6.

[277] Giorgos Stoilos, Giorgos Stamou and Stefanos Kollias. A string metric for ontology alignment. In Yolanda Gil, Enrico Motta, V. Richard Benjamins and Mark A. Musen, editors, *The Semantic Web – ISWC 2005*, pages 624–637, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-32082-1. doi:10.1007/s10115-009-0254-7. Cited on page 53.

[278] Erkki Sutinen and Jorma Tarhio. Filtration with *q*-samples in approximate string matching. In Dan Hirschberg and Gene Myers, editors, *Combinatorial Pattern Matching: 7th Annual Symposium, CPM 96 Laguna Beach, California, June 10–12, 1996 Proceedings*, pages 50–63. Springer Berlin Heidelberg, 1996. doi:10.1007/3-540-61258-0_4. Cited on page 12.

[279] Ann-Christine Syvänen. Accessing genetic variation: genotyping single nucleotide polymorphisms. *Nature Reviews Genetics*, 2(12):930–942, Dec 2001. doi:10.1038/35103535. Cited on page 6.

[280] Alireza Tahamtan and Abdollah Ardebili. Real-time RT-PCR in COVID-19 detection: issues affecting the results. *Expert Review of Molecular Diagnostics*, 20(5):453–454, 2020. doi:10.1080/14737159.2020.1757437. PMID: 32297805. Cited on page 4.

[281] Kiochiro Tamura and Masatoshi Nei. Estimation of the number of nucleotide substitutions in the control region of mitochondrial DNA in humans and chimpanzees. *Molecular Biology and Evolution*, 10(3):512–526, 1993. doi:10.1093/oxfordjournals.molbev.a040023. Cited on page 22.

[282] Kujin Tang, Jie Ren and Fengzhu Sun. Afann: bias adjustment for alignment-free sequence comparison based on sequencing data using neural network regression. *Genome Biology*, 20(1):266, Dec 2019. doi:10.1186/s13059-019-1872-3. Cited on pages 28 and 31.

[283] E. Taub, Floyd, James M. DeLeo and E. Brad Thompson. Sequential comparative hybridizations analyzed by computerized image processing can identify and quantitate regulated RNAs. *DNA*, 2(4):309–327, 1983. doi:10.1089/dna.1983.2.309. PMID: 6198132. Cited on page 4.

[284] The UniProt Consortium. UniProt: the universal protein knowledgebase. *Nucleic Acids Research*, 45(D1):D158–D169, 2017. doi:10.1093/nar/gkw1099. Cited on page 6.

[285] Ngoc Hieu Tran and Xin Chen. Comparison of next-generation sequencing samples using compression-based distances and its application to phylogenetic reconstruction. *BMC Research Notes*, 7(1):320, May 2014. doi:10.1186/1756-0500-7-320. Cited on pages 28 and 30.

[286] Michael Uhl, Torsten Houwaart, Gianluca Corrado, Patrick R. Wright and Rolf Backofen. Computational analysis of CLIP-seq data. *Methods*, 118-119:60–72, 2017. doi:10.1016/j.ymeth.2017.02.006. Protein-RNA: Structure Function and Recognition. Cited on page 6.

[287] Esko Ukkonen. On approximate string matching. In Marek Karpinski, editor, *Foundations of Computation Theory: Proceedings of the 1983 International FCT-Conference Borgholm, Sweden, August 21–27, 1983*, pages 487–495. Springer Berlin Heidelberg, 1983. doi:10.1007/3-540-12689-9_129. Cited on pages 22 and 23.

[288] Esko Ukkonen. Algorithms for approximate string matching. *Information and Control*, 64(1):100 – 118, 1985. doi:10.1016/S0019-9958(85)80046-2. International Conference on Foundations of Computation Theory. Cited on pages 22, 23, and 87.

[289] Esko Ukkonen. Finding approximate patterns in strings. *Journal of Algorithms*, 6(1):132 – 137, 1985. doi:10.1016/0196-6774(85)90023-9. Cited on page 23.

[290] Esko Ukkonen. Approximate string-matching with q-grams and maximal matches. *Theoretical computer science*, 92(1):191–211, 1992. doi:10.1016/0304-3975(92)90143-4. Cited on pages 11, 12, and 23.

[291] Anton Valouev, Jeffrey Ichikawa, Thaisan Tonthat, Jeremy Stuart, Swati Ranade et al. A high-resolution, nucleosome position map of c. elegans reveals a lack of universal sequence-dictated positioning. *Genome research*, 18(7):1051–1063, 2008. Cited on page 3.

[292] Jo Vandesompele, M Kubista, MW Pfaffl, Julie Logan, Kirstin Edwards and N Saunders. Real-time PCR: current technology and applications. *Reference gene validation software for improved normalization*, 2:47–64, 2009. Cited on page 4.

[293] Anchal Vishnoi, Rahul Roy, Hanumanthappa K. Prasad and Alok Bhattacharya. Anchor-based whole genome phylogeny (ABWGP): A tool for inferring evolutionary relationship among closely related microorganims. *PLOS ONE*, 5(11):1–11, 11 2010. doi:10.1371/journal.pone.0014159. Cited on page 30.

[294] Hüseyin Vural, Mehmet Kaya and Reda Alhajj. A model based on random walk with restart to predict circRNA-disease associations on heterogeneous network. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '19, page 929–932, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450368681. doi:10.1145/3341161.3343514. Cited on page 130.

[295] Filip Železný, Karel Jalovec and Jakub Tolar. Learning meets sequencing: a generality framework for read-sets. In *ILP 2014: 24th Internation Conference on Inductive Logic Programming, Late-Breaking Papers*, 2014. Cited on page 31.

[296] Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *Journal of the Association for Computing Machinery*, 21(1):168–173, January 1974. doi:10.1145/321796.321811. Cited on pages 21, 24, 36, 40, 41, 59, 60, 81, 87, and 137.

[297] Jason T. L. Wamg, Steve Rozen, Bruce A. Shapiro, Dennis Shasha, Zhiyuan Wang and Maisheng Yin. New techniques for DNA sequence classification. *Journal of Computational Biology*, 6(2):209–218, 1999. doi:10.1089/cmb.1999.6.209. PMID: 10421523. Cited on page 112.

[298] Lin Wan, Gesine Reinert, Fengzhu Sun and Michael S. Waterman. Alignment-free sequence comparison (II): theoretical power of comparison statistics. *Journal of Computational Biology*, 17(11):1467–1490, 2010. doi:10.1089/cmb.2010.0056. Cited on page 29.

[299] Lei Wang, Zhu-Hong You, De-Shuang Huang and Jian-Qiang Li. MGRCDA: Metagraph recommendation method for predicting circRNA-disease association. *IEEE Transactions on Cybernetics*, pages 1–9, 2021. doi:10.1109/TCYB.2021.3090756. Cited on page 131.

[300] Zhong Wang, Mark Gerstein and Michael Snyder. RNA-seq: a revolutionary tool for transcriptomics. *Nature Reviews Genetics*, 10(1):57–63, Jan 2009. doi:10.1038/nrg2484. Cited on page 4.

[301] René 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, 2007. doi:10.1093/bioinformatics/btl629. Cited on pages 32 and 91.

[302] James D. Watson and Francis H. C. Crick. A structure for deoxyribose nucleic acid. *Nature*, 171(4356):737–738, 1953. doi:10.1038/171737a0. Cited on page 1.

[303] Emanuel Weitschek, Daniele Santoni, Giulia Fiscon, Maria Cristina De Cola, Paola Bertolazzi and Giovanni Felici. Next generation sequencing reads comparison with an alignment-free distance. *BMC Research Notes*, 7(869), 2014. Cited on page 24.

[304] D. J. A. Welsh and M. B. Powell. An upper bound for the chromatic number of a graph and its application to timetabling problems. *The Computer Journal*, 10(1):85–86, 01 1967. doi:10.1093/comjnl/10.1.85. Cited on page 63.

[305] Mike West, Geoffrey S. Ginsburg, Andrew T. Huang and Joseph R. Nevins. Embracing the complexity of genomic data for personalized medicine. *Genome research*, 16(5): 559–566, 2006. doi:10.1101/gr.3851306. Cited on page 7.

[306] David L. Wheeler, Tanya Barrett, Dennis A. Benson, Stephen H. Bryant, Kathi Canese et al. Database resources of the National Center for Biotechnology Information. *Nucleic Acids Research*, 36(suppl_1):D13–D21, 2008. doi:10.1093/nar/gkm1000. Cited on page 6.

[307] Alden H. Wright. Approximate string matching using withinword parallelism. *Software: Practice and Experience*, 24(4):337–362, 1994. doi:10.1002/spe.4380240402. Cited on page 23.

[308] Zhengpeng Wu, Xi Wang and Xuegong Zhang. Using non-uniform read distribution models to improve isoform expression inference in RNA-Seq. *Bioinformatics*, 27(4): 502–508, 12 2010. doi:10.1093/bioinformatics/btq696. Cited on page 111.

[309] Feifei Xiao, Zhixiang Zuo, Guoshuai Cai, Shuli Kang, Xiaolian Gao and Tongbin Li. miRecords: an integrated resource for microRNA–target interactions. *Nucleic Acids Research*, 37(suppl_1):D105–D110, 11 2008. doi:10.1093/nar/gkn851. Cited on page 120.

[310] Zhengzheng Xing, Jian Pei and Eamonn Keogh. A brief survey on sequence classification. *SIGKDD Explor. Newsl.*, 12(1):40–48, nov 2010. doi:10.1145/1882471.1882478. Cited on page 112.

[311] Chang Xu. A review of somatic single nucleotide variant calling algorithms for next-generation sequencing data. *Computational and Structural Biotechnology Journal*, 16:15–24, 2018. doi:10.1016/j.csbj.2018.01.003. Cited on page 6.

[312] Mingsheng Xu, Robert G. Endres and Yasuhiko Arakawa. The electronic properties of DNA bases. *Small*, 3(9):1539–1543, 2007. doi:10.1002/smll.200600732. Cited on page 4.

[313] Zhao Xu and Bailin Hao. CVTree update: a newly designed phylogenetic study platform using composition vectors and whole genomes. *Nucleic Acids Research*, 37(suppl_2): W174–W178, 2009. doi:10.1093/nar/gkp278. Cited on page 29.

[314] Dongxia Yao, Lei Zhang, Mengyue Zheng, Xiwei Sun, Yan Lu and Pengyuan Liu. Circ2Disease: a manually curated database of experimentally validated circRNAs in human disease. *Scientific Reports*, 8(1):11018, Jul 2018. doi:10.1038/s41598-018-29360-3. Cited on page 131.

[315] Huiguang Yi and Li Jin. Co-phylog: an assembly-free phylogenomic approach for closely related organisms. *Nucleic Acids Research*, 41(7):e75, 2013. doi:10.1093/nar/gkt003. Cited on pages 28 and 30.

[316] Daniel R. Zerbino and Ewan Birney. Velvet: Algorithms for de novo short read assembly using de Bruijn graphs. *Genome Research*, 18(5):821–829, 2008. doi:10.1101/gr.074492.107. Cited on pages 12, 33, and 91.

[317] Han-Yuan Zhang, Lei Wang, Zhu-Hong You, Lun Hu, Bo-Wei Zhao et al. iGRLCDA: identifying circRNA–disease association based on graph representation learning. *Briefings in Bioinformatics*, 23(3), 03 2022. doi:10.1093/bib/bbac083. bbac083. Cited on page 131.

[318] Haoyu Zhang and Qin Zhang. EmbedJoin: Efficient edit similarity joins via embeddings. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 585–594, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4887-4. doi:10.1145/3097983.3098003. Cited on page 23.

[319] Meng-Long Zhang, Bo-Wei Zhao, Lun Hu, Zhu-Hong You and Zhan-Heng Chen. Predicting drug-disease associations via meta-path representation learning based on heterogeneous information net works. In De-Shuang Huang, Kang-Hyun Jo, Junfeng Jing, Prashan Premaratne, Vitoantonio Bevilacqua and Abir Hussain, editors, *Intelligent Computing Theories and Application*, pages 220–232, Cham, 2022. Springer International Publishing. ISBN 978-3-031-13829-4. doi:10.1007/978-3-031-13829-4_18. Cited on page 130.

[320] Yuchen Zhang, Xiujuan Lei, Zengqiang Fang and Yi Pan. CircRNA-disease associations prediction based on metapath2vec++ and matrix factorization. *Big Data Mining and Analytics*, 3(4):280–291, 2020. doi:10.26599/BDMA.2020.9020025. Cited on page 131.

[321] Zhongrong Zhang, Tingting Yang and Junjie Xiao. Circular RNAs: Promising biomarkers for human diseases. *EBioMedicine*, 34:267–274, Aug 2018. doi:10.1016/j.ebiom.2018.07.036. Cited on page 115.

[322] Bo-Wei Zhao, Lun Hu, Zhu-Hong You, Lei Wang and Xiao-Rui Su. HINGRL: predicting drug–disease associations with graph representation learning on heterogeneous information networks. *Briefings in Bioinformatics*, 23(1), 12 2021. doi:10.1093/bib/bbab515. bbab515. Cited on page 131.

[323] Bo-Wei Zhao, Lun Hu, Peng-Wei Hu, Zhu-Hong You, Xiao-Rui Su et al. MRLDTI: A meta-path-based representation learning model for drug-target interaction prediction. In De-Shuang Huang, Kang-Hyun Jo, Junfeng Jing, Prashan Premaratne, Vitoantonio Bevilacqua and Abir Hussain, editors, *Intelligent Computing Theories and Application*, pages 451–459, Cham, 2022. Springer International Publishing. ISBN 978-3-031-13829-4. doi:10.1007/978-3-031-13829-4_39. Cited on page 130.

[324] Qi Zhao, Yingjuan Yang, Guofei Ren, Erxia Ge and Chunlong Fan. Integrating bipartite network projection and KATZ measure to identify novel circRNA-disease associations. *IEEE Transactions on NanoBioscience*, 18(4):578–584, 2019. doi:10.1109/TNB.2019.2922214. Cited on page 131.

[325] Zhongming Zhao, Haipeng Li, Xiaozhuang Wu, Yixi Zhong, Keqin Zhang et al. Moderate mutation rate in the sars coronavirus genome and its implications. *BMC Evolutionary Biology*, 4(1):21, Jun 2004. doi:10.1186/1471-2148-4-21. Cited on page 111.

[326] Kai Zheng, Zhu-Hong You, Jian-Qiang Li, Lei Wang, Zhen-Hao Guo and Yu-An Huang. iCDA-CGR: Identification of circRNA-disease associations based on chaos game representation. *PLOS Computational Biology*, 16(5):1–22, 05 2020. doi:10.1371/journal.pcbi.1007872. Cited on page 131.

[327] Andrzej Zielezinski, Susana Vinga, Jonas Almeida and Wojciech M. Karlowski. Alignment-free sequence comparison: benefits, applications, and tools. *Genome Biology*, 18(1):186, Oct 2017. doi:10.1186/s13059-017-1319-7. Cited on pages 29 and 31.

[328] Andrzej Zielezinski, Hani Z. Girgis, Guillaume Bernard, Chris-Andre Leimeister, Kujin Tang et al. Benchmarking of alignment-free sequence comparison methods. *Genome Biology*, 20(1):144, Jul 2019. doi:10.1186/s13059-019-1755-7. Cited on page 31.

[329] Emile Zuckerkandl and Linus Pauling. Molecular disease, evolution, and genetic heterogeneity. In M. Kasha and B Pullman, editors, *Horizons in biochemistry*, pages 189–225. Academic Press, 1962. Cited on pages 10 and 25.

Publications of the Author

Publications Related to the Thesis

Journal Papers

[239] Petr Ryšavý and Filip Železný. Estimating sequence similarity from read sets for clustering next-generation sequencing data. *Data Mining and Knowledge Discovery*, 33 (1):1-23, Jan 2019. doi:10.1007/s10618-018-0584-8
Web Of Science: 0, Scopus: 2, Google Scholar: 3
Participation: 70 %

[242] Petr Ryšavý, Jiří Kléma and Michaela Dostálová Merkerová. circGPA: circRNA functional annotation based on probability-generating functions. *BMC Bioinformatics*, 23 (1):392, Sep 2022. doi:10.1186/s12859-022-04957-8
Web Of Science: 0, Scopus: 0, Google Scholar: 0
Participation: 60 %

Conference Papers

[237] Petr Ryšavý and Filip Železný. Estimating sequence similarity from read sets for clustering sequencing data. In *Advances in Intelligent Data Analysis XV: 15th International Symposium, IDA 2016, Stockholm, Sweden, October 13-15, 2016, Proceedings*, 2016. doi:10.1007/978-3-319-46349-0_18
Web Of Science: 3, Scopus: 3, Google Scholar: 4
Best paper award
Participation: 70 %

[238] Petr Ryšavý and Filip Železný. Estimating Sequence Similarity from Contig Sets. In *Advances in Intelligent Data Analysis XVI: 16th International Symposium, IDA 2017, London, UK, October 26–28, 2017, Proceedings*, 2017. doi:10.1007/978-3-319-68765-0_23
Web Of Science: 0, Scopus: 0, Google Scholar: 1
Participation: 70 %

Other Papers

[243] Petr Ryšavý. Using Tries for Evaluating Monge-Elkan Distance on Genomic Sequences. In *Proceedings of the International Student Scientific Conference Poster – 21/2017*, 2017.
Web Of Science: -, Scopus: -, Google Scholar: -
Participation: 100 %

[244] Petr Ryšavý. On Sequence Overlaps Minimizing Post-normalized Edit Distance. In *Proceedings of the International Student Scientific Conference Poster – 22/2018*,

2018.
Web Of Science: -, Scopus: -, Google Scholar: -
Participation: 100 %

[245] Petr Ryšavý. Approximate search in genomic data. In *Proceedings of the International Student Scientific Conference Poster – 23/2019*, 2019.
Web Of Science: -, Scopus: -, Google Scholar: -
Best paper award
Participation: 100 %

Articles under Review or in Press

[240] Petr Ryšavý and Filip Železný. Reference-Free Phylogeny from Sequencing Data..
Web Of Science: -, Scopus: -, Google Scholar: -
Accepted to the BMC BioData Mining journal, in press.
Participation: 70 %

[241] Petr Ryšavý and Filip Železný. An Algorithm to Calculate the p-value of the Monge-Elkan Distance.
Web Of Science: -, Scopus: -, Google Scholar: -
Accepted to the ISBRA 2022 conference, a special issue publication in a BMC journal is expected
Participation: 70 %

Publications not Related to the Thesis

Conference Papers

[246] Petr Ryšavý and Greg Hamerly. Geometric methods to accelerate k-means algorithms. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, 324-332, May 2016. doi:10.1137/1.9781611974348.37
Web Of Science: -, Scopus: 7, Google Scholar: 12
Participation: 50 %