Novel computational techniques for mapping and classifying Next-Generation Sequencing data

Karel Brinda

UNIVERSITÉ PARIS-EST

Thèse en vue de l'obtention du titre de
Docteur de l'Université Paris-Est

Spécialité : Informatique
École doctorale : MSTIC

Novel computational techniques for mapping and classifying Next-Generation Sequencing data

Karel Břinda

Soutenue le 28 novembre 2016

Jury:

Directeur de thèse	Gregory Kucherov, DR CNRS	LIGM Université Paris-Est, France
Co-encadrant de thèse	Valentina Boeva, CR INSERM	Institut Cochin, France
Rapporteur	Veli Mäkinen, Professeur	University of Helsinki, Finland
Rapporteur	Sven Rahmann, Professeur	Universität Duisburg-Essen, Germany
Président du jury	Dominique Lavenier, DR CNRS	IRISA/INRIA Rennes, France
Examinateur	Paola Bonizzoni, Professeure	Università Degli Studi di Milano-Bicocca, Italy
Examinateur	Guillaume Blin, Professeur	LaBRI Université Bordeaux, France
Examinateur	Denis Mestivier, Professeur	Université Paris-Est, France

K. Břinda. **Novel computational techniques for mapping and classifying Next-Generation Sequencing data.** PhD Thesis, Université Paris-Est, 2016.
DOI: 10.5281/zenodo.1045317

Acknowledgements

I would like to thank my PhD supervisors, Gregory Kucherov and Valentina Boeva, for bringing me into the fascinating world of bioinformatics, and for their guidance, support, and all the exciting discussions during last three years.

I am very thankful to the members of my committee, namely Veli Mäkinen, Sven Rahmann, Dominique Lavenier, Paola Bonizzoni, Guillaume Blin, and Denis Mestivier for their time and extensive feedback. I am grateful to all people who helped me with corrections of this thesis, mainly Gregory Kucherov, Valentina Boeva, Sven Rahman, Pavel Heller, Kamil Salikhov, and Brian Arnold.

I express a great deal of thanks to my family: my wife Markéta, Mom Ivana, Dad Karel and sister Romana. This thesis could not exist without their never-ending support, love and patience. Thank you!

I thank my research collaborators, namely Kamil Salikhov, Maciek Sykulski, Simone Pignotti and Mikhail Dubov for great teamwork. I am greatly thankful to Philippe Gambette and Paul Morel for their help with French administration at the beginning of my PhD.

During my PhD I had the possibility to work at three different research institutes: LIGM Université Paris-Est, Institut Curie, and Institut Cochin, where I met people from various countries, coming from diverse cultural backgrounds, speaking different languages and working on a wide spectrum of scientific subjects. I am very grateful for this opportunity and I really appreciate all the discussions with Manar Quamhieh, Fadhela Kerdjoudj, Safa Hamdoun, Zakaria Chemli, Pavel Heller, Sonja Hiltunen, Francesco Dolce, and Anthony Labarre at LIGM; Andrei Zinovyev, Maria Kondratova, Emmanuel Barillot, Inna Kuppertstein, Tatiana Popova, Frédéric Jarlier, and Elsa Bernard at Institut Curie; and Gwenneg Kerdivel and Marie-Ange Calmejane at Institut Cochin.

Finally, I would like to acknowledge a support by the ABS4NGS grant and by Labex Bézout of the French gouvernment (program *Investissement d'Avenir*).

Abstract

Novel computational techniques for mapping and classifying Next-Generation Sequencing data

Since their emergence around 2006, Next-Generation Sequencing technologies have been revolutionizing biological and medical research. Quickly obtaining an extensive amount of short or long reads of DNA sequence from almost any biological sample enables detecting genomic variants, revealing the composition of species in a metagenome, deciphering cancer biology, decoding the evolution of living or extinct species, or understanding human migration patterns and human history in general. The pace at which the throughput of sequencing technologies is increasing surpasses the growth of storage and computer capacities, which creates new computational challenges in NGS data processing.

In this thesis, we present novel computational techniques for read mapping and taxonomic classification. With more than a hundred of published mappers, read mapping might be considered fully solved. However, the vast majority of mappers follow the same paradigm and only little attention has been paid to non-standard mapping approaches. Here, we propound the so-called dynamic mapping that we show to significantly improve the resulting alignments compared to traditional mapping approaches. Dynamic mapping is based on exploiting the information from previously computed alignments, helping to improve the mapping of subsequent reads. We provide the first comprehensive overview of this method and demonstrate its qualities using Dynamic Mapping Simulator, a pipeline that compares various dynamic mapping scenarios to static mapping and iterative referencing.

An important component of a dynamic mapper is an online consensus caller, i.e., a program collecting alignment statistics and guiding updates of the reference in the online fashion. We provide Ococo, the first online consensus caller that implements a smart statistics for individual genomic positions using compact bit counters. Beyond its application to dynamic mapping, Ococo can be employed as an online SNP caller in various analysis pipelines, enabling SNP calling from a stream without saving the alignments on disk.

Metagenomic classification of NGS reads is another major topic studied in the thesis. Having a database with thousands of reference genomes placed on a taxonomic tree, the task is to rapidly assign a huge amount of NGS reads to tree nodes, and possibly estimate the relative abundance of involved species. In this thesis, we propose improved computational techniques for this task. In a series of experiments, we show that spaced seeds consistently improve the classification accuracy. We provide Seed-Kraken, a spaced seed extension of Kraken, the most popular classifier at present. Furthermore, we suggest ProPhyle, a new indexing strategy based on a BWT-index, obtaining a much smaller and more informative index compared to Kraken. We provide a modified version of BWA that improves the BWT-index for a quick k-mer look-up.

Résumé

Nouvelles techniques informatiques pour la localisation et la classification de données de séquençage haut débit

Depuis leur émergence autour de 2006, les technologies de séquençage haut débit ont révolutionné la recherche biologique et médicale. Obtenir instantanément une grande quantité de courtes ou longues lectures de presque tout échantillon biologique permet de détecter des variantes génomiques, révéler la composition en espèces d'un métagénome, déchiffrer la biologie du cancer, décoder l'évolution d'espèces vivantes ou disparues, ou mieux comprendre les schémas de la migration humaine et l'histoire humaine en général. La vitesse à laquelle augmente le débit des technologies de séquençage dépasse la croissance des capacités de calcul et de stockage, ce qui crée de nouveaux défis informatiques dans le traitement de données de séquençage haut débit.

Dans cette thèse, nous présentons de nouvelles techniques informatiques pour la localisation (mapping) de lectures dans un génome de référence et pour la classification taxonomique. Avec plus d'une centaine d'outils de localisation publiés, ce problème peut être considéré comme entièrement résolu. Cependant, une grande majorité de programmes suivent le même paradigme et trop peu d'attention a été accordée à des approches non-standards. Ici, nous introduisons la localisation dynamique dont nous montrons qu'elle améliore significativement les alignements obtenus, par comparaison avec les approches traditionnelles. La localisation dynamique se fonde sur l'exploitation de l'information fournie par les alignements calculés précédemment, afin d'améliorer les alignements des lectures suivantes. Nous faisons une première étude systématique de cette approche et démontrons ses qualités à l'aide de Dynamic Mapping Simulator, une pipeline pour comparer les différents scénarios de la localisation dynamique avec la localisation statique et le «référencement itératif».

Une composante importante de la localisation dynamique est un calculateur online de consensus, c'est-à-dire un programme qui collecte des statistiques des alignements pour guider, à la volée, les mises à jour de la référence. Nous présentons Ococo, calculateur du consensus online qui maintient des statistiques des positions génomiques individuelles à l'aide de compteurs de bits compacts. Au-delà de son application à la localisation dynamique, Ococo peut être utilisé comme un calculateur online de SNP dans divers pipelines d'analyse, ce qui permet de prédire des SNP à partir d'un flux sans avoir à enregistrer les alignements sur disque.

La classification métagénomique de lectures d'ADN est un autre problème majeur étudié dans la thèse. Étant donné des milliers de génomes de référence placés sur un arbre taxonomique, le problème consiste à affecter rapidement aux nœuds de l'arbre une énorme quantité de lectures NGS, et éventuellement estimer l'abondance relative des espèces concernées. Dans cette thèse, nous proposons des techniques améliorées pour cette tâche. Dans une série d'expériences, nous montrons que les graines espacées améliorent la précision de la classification. Nous présentons Seed-Kraken, extension du logiciel pop-

ulaire Kraken utilisant les graines espacées. En outre, nous introduisons ProPhyle, une nouvelle stratégie d'indexation basée sur la transformée de Burrows-Wheeler (BWT), qui donne lieu à un indice beaucoup plus compact et plus informatif par rapport à Kraken. Nous présentons une version modifiée du logiciel BWA qui améliore l'index BWT pour la localisation rapide de k-mers.

Papers, posters and presentations

Papers

i) K. Břinda, V. Boeva, and G. Kucherov. "Dynamic read mapping and online consensus calling for better variant detection."
URL:http://arxiv.org/abs/1605.09070

ii) K. Břinda, M. Sykulski, and G. Kucherov. "Spaced seeds improve k-mer-based metagenomic classification." In: *Bioinformatics* 31.22 (2015), pp. 3584–92.
DOI:10.1093/bioinformatics/btv419

iii) K. Břinda, V. Boeva, and G. Kucherov. "RNF: a general framework to evaluate NGS read mappers." In: *Bioinformatics* 32.1 (2015).
DOI:10.1093/bioinformatics/btv524

iv) K. Břinda. "Languages of lossless seeds". In: *Electronic Proceedings in Theoretical Computer Science* 151 (2014), pp. 139–150.
DOI:10.4204/EPTCS.151.9

Posters

i) K. Břinda, V. Boeva, and G. Kucherov. "RNF: a general framework to evaluate NGS read mapper." Conference "HitSeQ 2015", Dublin (Ireland), July 10–11, 2015.

ii) K. Břinda, V. Boeva, and G. Kucherov. "RNF: a method and tools to evaluate NGS read mappers." Conference "RECOMB 2015", Warsaw (Poland), April 12–15, 2015.

iii) M. Sykulski, K. Břinda, and G. Kucherov. "Spaced seeds improve metagenomic classification." Conference "RECOMB 2015", Warsaw (Poland), April 12–15, 2015.

Presentations

i) K. Břinda, K. Salikhov, S. Pignoti, and G. Kucherov. "ProPhyle: a memory efficient BWT-based metagenomic classifier using k-mer propagation." Workshop "SeqBio 2016", Nantes (France), November 18, 2016.

ii) K. Břinda, K. Salikhov, S. Pignoti, M. Sykulski, and G. Kucherov. "BWT-based indexing structure for metagenomic classification." Seminar at INRIA/Irisa Rennes, Rennes (France), July 7, 2016.

iii) K. Břinda, K. Salikhov, M. Sykulski, and G. Kucherov. "BWT-based indexing structure for metagenomic classification." Student Conference "Quantitative Genomics 2016", London (UK), June 6, 2016.

iv) K. Břinda, K. Salikhov, M. Sykulski, and G. Kucherov. "Indexing structures for metagenomic classification." International workshop "Data Structures in Bioinformatics", Bielefeld (Germany), February 23–24, 2016.

v) K. Břinda, V. Boeva, and G. Kucherov. "Dynamic read mappers." International Workshop "Algorithmics, Bioinformatics and Statistics for NGS data analysis 2015", Paris (France), June 22–23, 2015.

vi) K. Břinda, V. Boeva, and G. Kucherov. "Dynamic read mappers." Workshop "SeqBio 2014", Montpellier (France), November 4–5, 2014.

vii) K. Břinda. "Languages of lossless seeds." International conference "Automata and Formal Languages 2014", Szeged (Hungary), May 27–29, 2014.

Other works

i) P. Červenka, K. Břinda, M. Hanousková, P. Hofman, and R. Seifert. "Blind Friendly Maps: Tactile Maps for the Blind as a Part of the Public Map Portal (Mapy.cz)." In: *Computers Helping People with Special Needs: 15th International Conference, ICCHP 2016, Linz, Austria, July 13-15, 2016, Proceedings, Part II,* pp. 131–138.
DOI:10.1007/978-3-319-41267-2_18

ii) K. Břinda, P. Červenka, R. Seifert, and P. Hofman. "Blind Friendly Maps." Poster at international conference "Sensory issues and Disability: Touch to learn, touch to communicate", Paris (France), March 17–19, 2016.
URL:https://hal.archives-ouvertes.fr/hal-01289174

Contents

Part I

Introduction

Contents - Part I

Chapter 1

Context, motivation and contributions

1.1 Motivation of our work

Since their emergence around 2006, Next-Generation Sequencing (NGS) technologies have been revolutionizing biological and medical research. Rapidly obtaining an extensive amount of so-called reads, i.e., sequenced fragments of DNA, from almost any biological sample enables studies has never been possible before. Let us start with several motivating examples of scientific areas, where NGS methods provided a deep insight.

Detecting genomic variants associated to diseases. With the state-of-the-art sequencing technologies, it is possible to quickly and cheaply detect mutational variants present in a population, in an organism, or sometimes even in a single cell. Variant detection is usually a prior step to genome-wide association studies connecting genetic variants to traits [6, 7, 8]. Then catalogs of variants related to specific genetic diseases can be created. Similarly, we can also associate genomic variants in bacteria to antibiotic resistance and rapidly detect the resistance directly from sequencing data [9, 10].

When a high-quality reference sequence is available, detecting genomic variants usually proceeds by read mapping, a procedure based on fast alignment of NGS reads against the reference using similarity search. Genomic variants then can be deduced from differences between the computed alignments and the reference.

Studying metagenomes. Using classification methods on sequencing data, we can reveal the composition of species in a metagenome, i.e., an environmental sample containing genomes of many individual organisms. The human microbiome project [11] and the TARA ocean project [12, 13, 14] are the most famous examples, but many other metagenomes have been studied, e.g., metagenomes of cities [15] or households [16]. In addition to the composition, we can also study functions of metagenomes and their properties. For instance, it has been shown that vast majority of unique genes in the human body are microbial [17].

Deciphering cancer biology. DNA sequencing has enhanced our understanding of specific biological processes in cancer, which was nearly impossible before. Cancer cells mutate very quickly and form various subclones, such that cancer behaves more like a set

of diseases than a single disease alone. One of the main scientific goals is to mathematically describe individual subclones of cancer, distinguish "driver" mutations from "passenger" ones, and predict future behaviour and reaction of the tumor to drugs [18].

Many methods for studying cancer rely on high quality read mapping as mutations important in the context of a specific cancer type can be often very infrequent in reads covering a given position and present in certain subclones only. Quality of alignments reported by a mapper strongly affects results of the analysis pipelines.

Clarifying human history and migration patterns. Since ancient biological samples are sequencable with state-of-the-art methods [19, 20], DNA sequencing can provide deep insights to human history. Richard III. [21], Albert I. [22], Louis XVII [23], or Tutankhamun [24] are examples of famous historical figures already studied using sequencing. Note that the techniques used in such studies are very relevant also for criminology [25].

Sequencing allows to better study ancient microbiomes, for example human pathogens [26, 27]. It helped to better characterize plague epidemics [28, 29, 30, 31] or revealed interesting information about eating habits in various historical epochs from calcified dental plaque [32].

Demographic history of human and its close relatives belongs to other widely studied topics [33, 34]. Sequencing has been used, for instance, to study Denisovans [35], Neandertals [36, 37, 38, 39, 40, 41], Aboriginal Australians [42], Native Americans [43], Romans [44] or British [45], to name at least several examples.

Ancient samples are very sensitive to contamination. Thus, it is important to correctly identify sequences which do not come from the ancient sample itself, but from the environment around.

1.2 Our contributions

In this thesis, we present several computational contributions in domains of read mapping and metagenomic classification. These methods are relevant in all the examples presented above and can be useful in different phases of analysis pipelines.

Read mapping. To improve alignments of NGS reads, we suggest to use an approach called *dynamic read mapping* that is an improvement of the standard mapping approach with correction of the reference sequence according to alignments computed so far. First, we present RNFtools, a framework for comparative analyses of different alignments techniques (Chapter 6). The entire framework is based on a novel format for naming simulated NGS reads (Appendix B). Then we study the problem of online consensus calling and provide the first online consensus caller operating directly on a stream of alignments (Chapter 7). Indeed, every dynamic mapper needs to perform consensus calling in an online fashion to decide whether and how to update the reference. Finally, we show that dynamic mapping provides alignments of superior quality compared to static mapping (Chapter 8). In a pipeline called Dynamic Mapping Simulator, we proceed with simulating different dynamic mapping scenarios by iterative calling of existing static mappers, and by comparing the resulting alignments using RNFtools. In the supplement, we also provide new theoretical results about lossless spaced seeds for read mapping (Appendix A).

Metagenomic classification. Metagenomic classification is classification of reads from an environmental sample. We improve existing methods for assignment of reads to a

taxonomic tree, which can be a prior step to abundance estimation or sequence assembly.

First we show that metagenomic classification can be strongly improved using spaced seeds and we provide Seed-Kraken, a spaced-seed modification of the most popular classifier (Chapter 11). Then we introduce out own k-mer based classifier implementing a novel BWT-indexing structure specifically designed for k-mers placed in a tree (Chapter 12).

Chapter 2

DNA sequencing

DNA sequencing is the process of determining the nucleotide sequence in DNA molecules. During the last 40 years, various sequencing techniques were developed ranging from the original Sanger sequencing to modern Next-Generation Sequencing methods. In this chapter, we provide a short overview of the most popular sequencing technologies and tools for their simulation. For a detailed review, see [46].

2.1 General approach

The general approach for sequencing is similar for all technologies. Until today, it remains hard to sequence the entire DNA molecule as a whole so the methods usually rely on sequencing fragments of copies of the original molecule. The sequencing process can be understood as "reading" these fragments and encoding the obtained information to data files, usually textual.

Generally speaking, this "reading" consists of obtaining a technology-specific signal characterizing the DNA fragment and recoding this signal to a sequence of letters of the DNA alphabet (so-called base calling), possibly providing also information about reliability of individual letters (Figure 2.1). Exact steps of the entire process vary in individual technologies. For instance, the resulting signal can have various forms such as a function of electrical current or a series of photographies. The resulting DNA strings, commonly called reads, may not fully correspond to the original DNA molecule since individual steps of the process introduce *sequencing errors*.

To design algorithms for read processing, we need to well understand properties and particularities of the technologies. The main parameters are statistical distribution of read length, statistical properties of sequencing errors (probabilities of individual types of errors, their common patterns, etc.), sequencing biases (e.g., coverage bias), amount of produced data within a single sequencing experiment, or reliability of provided base qualities. Moreover, some technologies are capable to provide reads in pairs proximal in the original DNA (so-called paired-end and mate-pair sequencing).

Also different data are obtained based on the type of sequencing such as whole genome sequencing (WGS), whole exome sequencing (WES), target sequencing (TS), whole transcriptome shotgun sequencing (WTSS, RNA-seq), methylation sequencing (MeS, BS-seq), and others [48].

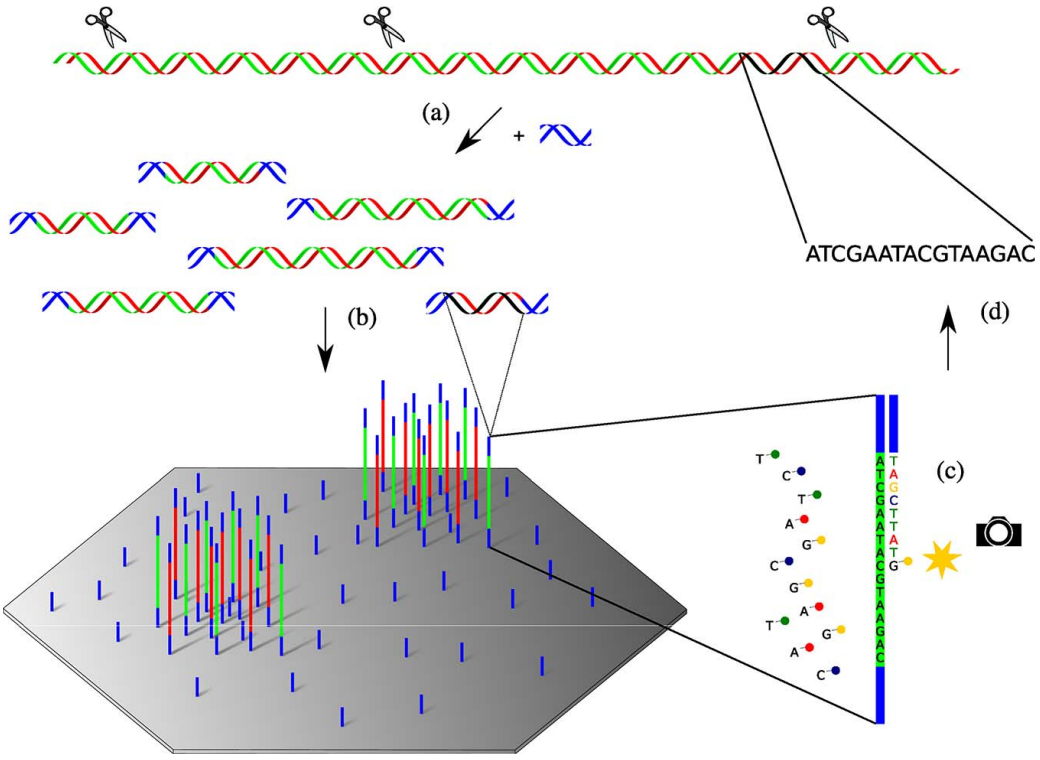


Figure 2.1: **Illustration of the sequencing process on example of the Illumina platform.** *©2015 IEEE. Reprinted, with permission, from [47].*

2.2 First methods for DNA sequencing

The first method for DNA sequencing was developed in 1977 by Frederick Sanger. Based on the chain-termination method, his technique enabled to obtain a list of occurrences of each of four nucleotides in the DNA molecule. The original approach was later strongly improved and commercialized by Applied Biosystems, which introduced in 1987 the first automatic sequencing machine. Sanger sequencing remained to be the main method for the next 25 years after its development. Even today, with Next-Generation sequencing technologies available, it still finds its use, especially for validation of results.

Sanger sequencing can provide reads of length up to approximately $1,000$ bp [51] with an extremely low error rate ranging from 0.00001 to 0.0001 [52]. Highly limited speed and the overall cost per basepair remain to be the main disadvantages of Sanger-like approaches.

2.3 Next-Generation Sequencing methods

Research of sequencing technologies was strongly accelerated in the 1990s when sequencing the human genome was set as a priority scientific goal and Human Genome Project (HGP) launched. Even though the entire human genome was sequenced using traditional methods [53, 54], it became obvious that these methods were too slow and costly for whole genome sequencing, thus unsuitable for massive sequencing of many living organisms. Indeed,

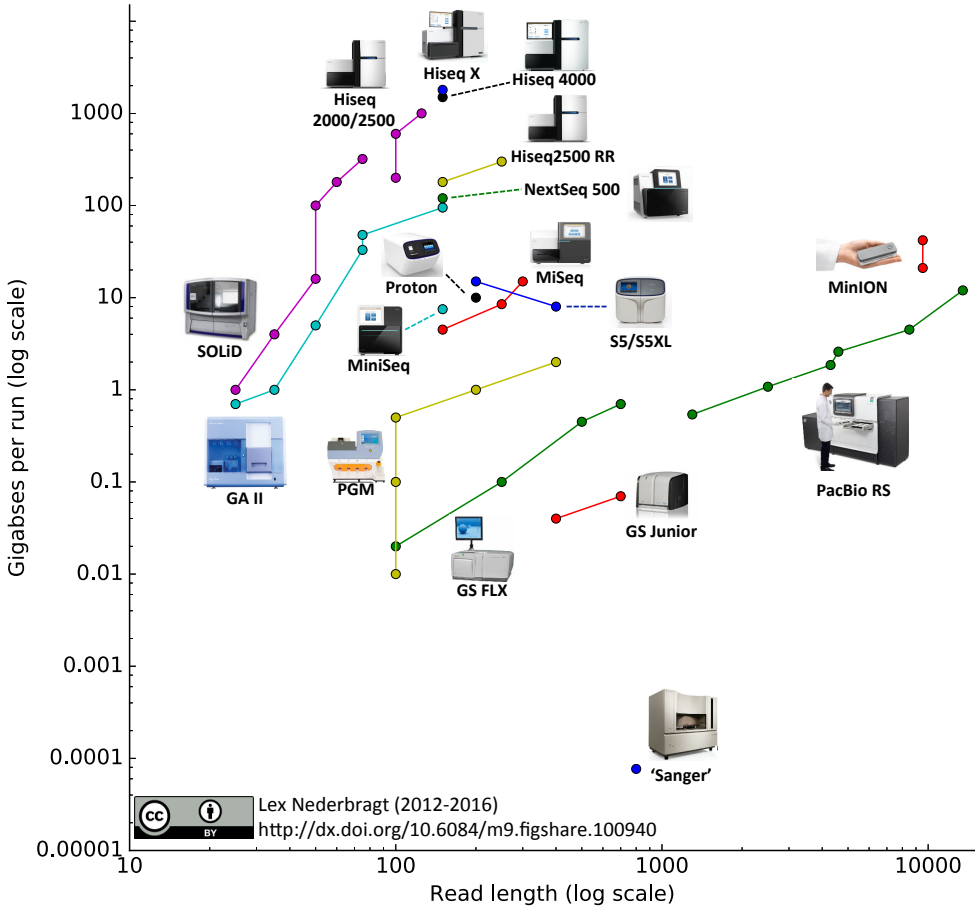


Figure 2.2: **Development in sequencing technologies.** Overview of existing sequencing platforms and their throughput per run and typical length of resulting reads, namely: *ABI Sanger*, *Illumina* (GA, HiSeq, MiSeq, MiniSeq, and NextSeq), *Ion Torrent* (PGM, Proton, and S5/S5XL), *SOLiD*, *454* (GS FLX and GS Junior), *Oxford Nanopore* (Minion), and *PacBio* (PacBio RS). *©2016 Lex Nederbragt. Reprinted from [49].*

the overall expenses of HGP are estimated to $2.7 billion. As a result, a huge effort was invested into research of completely new technologies, which could provide rapid highly parallel sequencing.

First such novel technologies were introduced in 2005 [51], and they are commonly referred as Next-Generation Sequencing (NGS), second generation sequencing or massive parallel sequencing. They immediately revolutionized genomic research as they enabled to instantly obtain millions of reads. The first Solexa sequencers provided reads of length 25 bp only [55], but even such reads permitted successful single-nucleotide polymorphism (SNP) calling. In course of time, the short-read technologies (Illumina, SOLiD, 454, or Ion Torrent) became mature enough and they nowadays provide reads of length <400 bp (with exception of 454 with slightly longer reads) with a relatively low error rate.

The introduction of long-read technologies in 2011, providing reads of length $>10,000$ bp, is surely another important milestone. These technologies, mainly represented by PacBio and Oxford Nanopore, are sometimes referred separately as third generation sequencing. The associated error rate >0.10 (sometimes even >0.25) makes the data analysis even

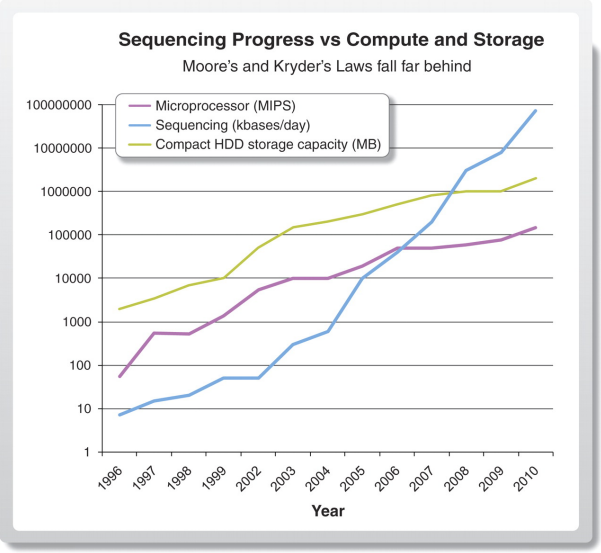


Figure 2.3: **Sequencing progress vs. compute and storage.** A doubling of sequencing output every 9 months has outpaced and overtaken performance improvements within the disk storage and high-performance computation fields. *From [50]. Reprinted with permission from AAAS.*

more challenging and also computationally highly intensive.

2.3.1 Sequencing technologies producing short reads

Illumina. Illumina (originally named Solexa) was released in 2006 [51] and has ultimately become a technology dominating the market [46]. Its state-of-the-art sequencers can produce reads of length 100 bp to 300 bp and paired-end reads are supported. Its overall error rate is very low; the most common errors are substitutions with a typical rate 0.005 and 0.010 for first and second end of a pair, respectively [56, 57]. The error rate increases towards the ends of the reads, but the errors can be relatively easily corrected [58, 59, 60, 61].

Ion Torrent. The Ion Torrent sequencing platform, first released in 2010 [51], can produce reads of length up to 400 bp. The most frequent errors are indels, which appear with rate 0.03 [62], whereas substitution errors are by order of magnitude less frequent. As the error rate can be improved by quality clipping, some publications (e.g, [46, 63]) mention the improved error rate 0.01.

SOLiD. SOLiD sequencers were introduced in 2006 [51]. They can produce reads up to 100 bp with very low error rate <0.001, possibly paired-end. A particularity of SOLiD sequencers is the used alphabet. Opposed to the other technologies, reads are encoded in color (di-nucleotide) space [64], i.e., transitions between adjacent nucleotides are stored instead of the nucleotides itself. A major advantage of this encoding is the fact that sequencing errors can be distinguished from single-nucleotide variants. While the former is observed as a single mismatch, the latter causes two adjacent mismatches. On the other

hand, read mappers without explicit support for SOLiD are not applicable, which strongly limits its usage.

454. 454 sequencers were first introduced in 2005 [51]. According to produced data, 454 lies on the border between short and long read technologies. They provide reads of length up to 1,000 bp (depending on exact sequencer type) with error rate about 0.01, of which the majority are indels [46, 63]. Paired-end reads are supported.

2.3.2 Sequencing technologies producing long reads

Pacific Bioscience. The PacBio sequencing technology provides reads of length up to 20,000 bp with error rate ranging from 0.11 to 0.15 [65]. A major advantage of PacBio is the fact that errors are distributed randomly, therefore, they are easier to be distinguished from genomic variants. Note that short reads can be used for their correction (see, e.g., [66]).

Oxford Nanopore. Oxford Nanopore produces very special sequencers, distinct from the other technologies in many aspects. First of all, Nanopore sequencers have a size of a smart phone, which makes them the most mobile sequencers on the market.

The technology itself is based on decoding electrical signals from protein pores, which are embedded in an electrically resistant polymer membrane. Voltage created across this membrane causes pass of DNA molecules through the membrane in a single direction. The associated changes in electrical current on the pores are recorded and exact sequence of nucleotides decoded from them.

Properties of Oxford Nanopore data strongly depend on the specific choice of the used chemistry with many possible combinations. Obtained reads can be up to 200,000 bp long [46] with a typical error rate about 0.12, mainly represented by indels [46]. The most error-prone step of the sequencing process is decoding the electrical signal, mainly because it is hardly possible to maintain a constant speed of DNA molecule passage during the sequencing. Therefore, especially homopolymeric regions are hard to be sequenced (thus decoded [67]) correctly.

Oxford Nanopore sequencing is currently a rapidly developing technology, providing data of constantly increasing quality. The associated high error rate remains to be the major disadvantage in practical applications. Nevertheless, the high mobility and comparatively low prices compensate for this drawback and make Oxford Nanopore a very perspective technology, highly suitable for "point-of-care" disease detection, field pathogen detection, civil and army protection, water quality surveillance [68], real-time disease surveillance (e.g, of Ebola [69]), or for sequencing in the space [70, 71]. Other particularities of the technology are a so-called selective sequencing (ReadUntil) [72], i.e., fast skipping molecules out-of-interest in order to accelerate sequencing, and direct methylation sequencing [73].

2.4 Read simulation

To evaluate methods for NGS data analysis, we need to realistically simulate reads. First of all, it is the only approach ensuring that we know the ground truth (reads' true origin in the genome, positions of sequencing errors, variants present in the genome, etc.). Second, real sequencing experiments are usually time demanding and costly.

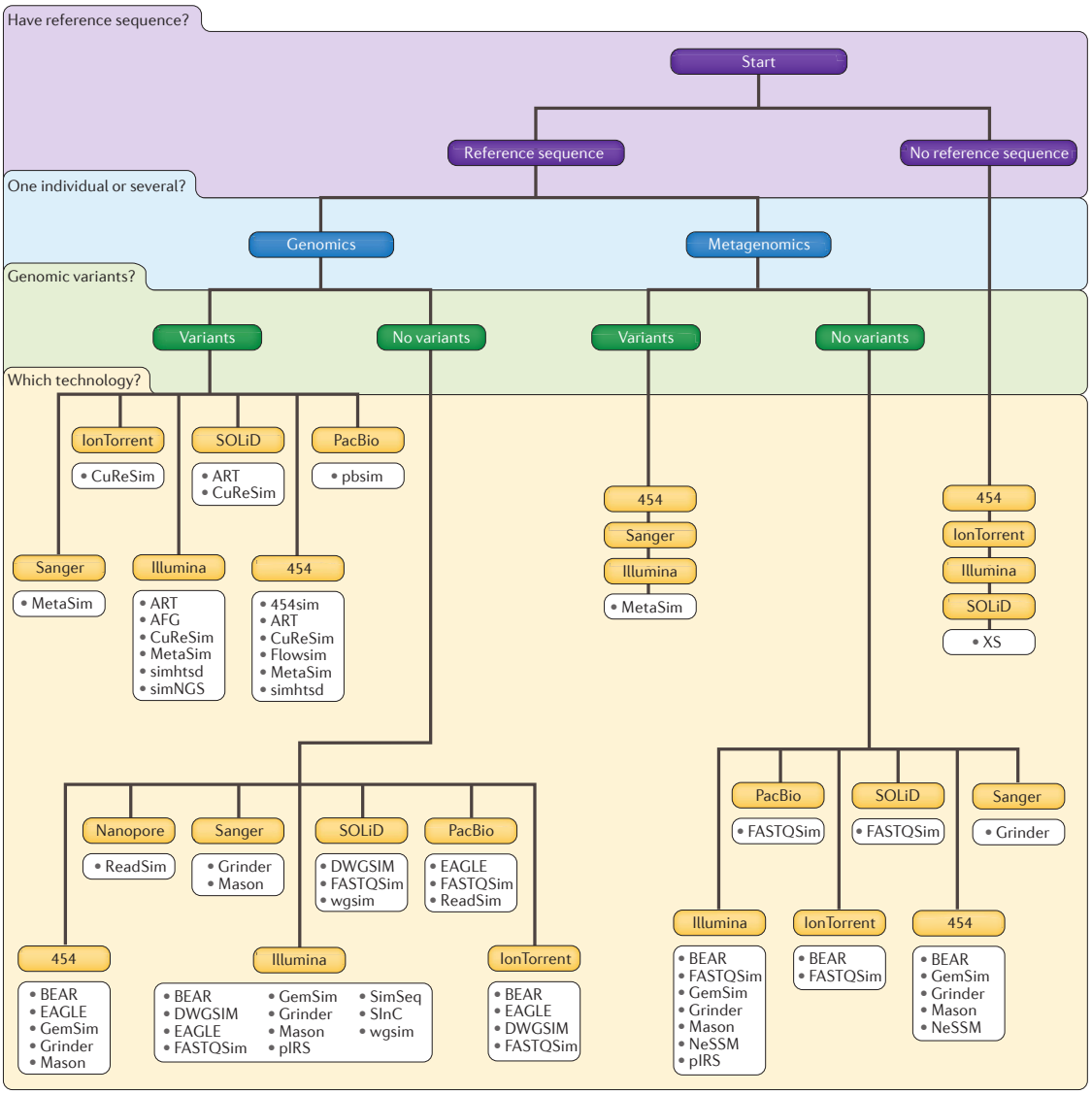


Figure 2.4: **Decision tree for the selection of an appropriate read simulator.** This tree contains most of the state-of-the-art simulators from this chapter. It enables to quickly exclude unsuitable simulators and take a list of candidates supporting the specified sequencing technology and type of the simulation. A final simulator should be selected after comparing simulators' properties described in [74, 75] (such as categories of simulated variants). *Reprinted by permission from Macmillan Publishers Ltd: Nature Reviews Genetics [75], ©2016.*

Many sequencing technologies are available nowadays and every technology has proper characteristics, usually also depending on experimental conditions. As a result, many simulators have been developed and it is a non-trivial task to select an appropriate one. To do so, we should consider the following aspects. A simulator can be either multipurpose (e.g., WGsim), or specifically tuned for a particular technology (e.g., PBsim [76]). It can support different types of sequencing, e.g., whole-genome sequencing, metagenomic sequencing, RNA-seq, or BS-seq. Simulators can be further categorized based on types

of variants that they introduce, from those supporting only SNPs and small indels (e.g., WGsim) to simulators introducing chromosomal duplications or large scale variations (e.g., pIRS [77]). Alternatively, a dedicated tool outside the read simulator itself can be used for this task (e.g., Mason Variator [78], RSVSim [79], SCNVSim [80], SNP Mutator[1], or VarSim [81]). A good simulator should also support diploid simulations, i.e., introduce two sets of variants to the reference sequence.

Fidelity of simulation is strongly determined by the employed statistical model; e.g., for sequencing errors, sequencing biases (e.g., coverage bias), or base qualities. Models can range from uniform distributions of mismatches (e.g., WGsim) to advanced models learning large set of parameters from alignments of real reads (e.g., ART [82]).

Besides the simulated sequences themselves, various information about simulation should be stored. Unfortunately, almost every read simulator uses own proper way to store sequencing errors, variants, reads' genomic coordinates, etc. To solve this issue, we have developed the RNF format. For more information, see Chapter 6.

Here we provide a comprehensive list of existing read simulators. For more detailed information, please see two recently published reviews [74, 75]. Decision tree in Figure 2.4 (adopted from [75]) is currently the best existing guide for selecting the appropriate read simulator.

Exhaustive list of read simulators

1. **Multipurpose simulators.** These tools are designed for a wider class of technologies.
   i) **Short reads.** Artificial FASTQ generator [83], DWGsim[2], FASTQSim [84], Mason [78], NEAT [85], SeqMaker [86], and WGsim[3].
   ii) **Long reads.** FASTQSim [84], Loresim 2[4], and Loresim[5].
2. **Simulators for a particular technology.** They are specifically tuned for a single technology.
   i) **10x Genomics.** LRSIM [87].
   ii) **Illumina** ART-Illumina [82], EAGLE[6], GemSim [88], pIRS [77], SInC [89], SimNGS[7], SimHTDS[8], and Wessim [90].
   iii) **IonTorrent.** CuReSim [91].
   iv) **Oxford Nanopore.** NanoSim [92], NanoSimH[9], ReadSim [93], and SiLiCO [94].
   v) **PacBio.** LongISLND [95], PBSim [76], PBLibSim[10], ReadSim [93], Random-reads (a part of the BBtools package [96]), SiLiCO [94], and SimLoRD [97].
   vi) **Roche 454.** 454sim [98], ART-454 [82], FLOWSIM [99], simhtsd[11], and Wessim

[1] https://github.com/CFSAN-Biostatistics/snp-mutator
[2] http://github.com/nh13/dwgsim
[3] http://github.com/lh3/wgsim
[4] https://github.com/gt1/loresim2
[5] https://github.com/gt1/loresim
[6] https://github.com/sequencing/EAGLE
[7] http://www.ebi.ac.uk/goldman-srv/simNGS/
[8] https://sourceforge.net/projects/simhtsd/
[9] https://github.com/karel-brinda/NanoSimH/
[10] https://github.com/ethanagbaker/pblibsim
[11] https://sourceforge.net/projects/simhtsd/

[90].

vii) **SOLiD.** ART-SOLiD [82]; DWGsim, and WGsim.

3. **Simulators of particular data.** They are designed for other experiment types than whole-genome sequencing.

   i) **Metagenomic simulators.** BEAR [100], FASTQSim [84], Grinder [101], GemSim [88], Mason [78], MetaSim [102], and NeSSM [103] (also a GPU version exists).

   ii) **RNA-seq simulators.** BEERS [104], DWGsim, Flux [105], PSIM [106], rlsim [107], RNASeqReadSimulator[12], RSEM [108], and SimSeq [109].

   iii) **BS-seq simulators.** BSSim, DNemulator [110], Sherman[13], and WGBS Suite [111].

4. **Miscellenaous**

   Gargammel [112] simulates ancient DNA. XS [113] simulates formats of FASTQ files of various sequencing platforms (e.g., read names) and also reference sequences (instead of taking existing ones). GenFrag [114], Celsim [115], and FASIM [116] are historical predecessors of modern read simulators.

[12] https://github.com/davidliwei/RNASeqReadSimulator
[13] http://www.bioinformatics.babraham.ac.uk/projects/sherman/

Chapter 3

Main techniques of pairwise sequence comparison

> In this chapter, we summarize the main alignment-based and alignment-free techniques for pairwise sequence comparison, with a particular emphasis on those which are relevant for read mapping (studied in Part II) and metagenomic classification (studied in Part III).

Pairwise sequence comparison is in the core of many problems in computational biology and computer science. Let us mention several of them to illustrate some typical applications.

1. **Mapping of NGS reads.** To map NGS reads to a reference genome, modern mappers proceed by searching the most similar regions in the reference [47].
2. **Taxonomic classification of NGS reads.** Reads sequenced from a metagenome are classified to taxonomic clades [117]. This classification is done based on quick estimates of similarity between the reads and the reference genomes placed within a taxonomic tree.
3. **Homologous regions detection.** Homologs, i.e., regions descending from a common ancestor, are usually detected based on sequence similarity. They are widely studied within evolutionary biology.
4. **Genome assembly.** Many genome assemblers proceed by building a graph of prefix-suffix overlaps between NGS reads and enumerating paths in this graph [118].
5. **Phylogenetic inference.** Phylogenetic trees and networks illustrate evolution history of species, languages [119, 120], or even manuscripts [121] and stories [122]. Phylogenies are usually inferred from similarities between individual sequences.
6. **Detection of duplicate web pages.** Before adding a new web page into the index, search engines need to quickly verify whether a highly similar document has not been already indexed [123].

There exist two different general approaches to sequence comparison. *Alignment based methods* aim at identification of edit operations transforming one sequence into another (or their segments), while *alignment-free methods* quickly estimate the similarity of the sequences based on their composition.

3.1 Alignment-based methods

The biological interest in sequence comparison started with the study of homologs, i.e., regions descending from a common ancestor. To do so, it was necessary to handle the notion of sequence similarity mathematically and sequence alignment [124] techniques have been studied.

Sequence alignment can be viewed as a search of transformations of multiple input sequences into a single sequence called *consensus*, which should be maximally similar to all the original sequences. Such a transformation consists of a series of modifications (usually letter-by-letter) such as a *match* (keep the letter), a *mismatch* (edit the letter), an *insertion* (insert a letter), or a *deletion* (delete the letter), from a predefined list. Every operation is associated with a certain integer cost (called *score of the operation*) defined by a *scoring system*. Our goal is to minimize the sum of the costs of all performed transformations (a so-called *alignment score*). A single transformation from an input sequence to the consensus can be described using a so-called Compact Idiosyncratic Gapped Alignment Report string (CIGAR) (see, e.g, [125]).

Pairwise sequence alignment is the sequence alignment of two sequences. In this particular case, we can simply reduce it to a transformation of the first sequence into the second one. From now on, we will consider pairwise alignments only.

The approach considering a transformation of the *entire* input sequence to the entire output sequence is called *global alignment* as it captures the global similarity between the sequences. In practice, we often are more interested in local similarities corresponding to the transformations of a substring to a substring. In such a case, we talk about a so-called *local alignment*. It can be viewed as a global alignment after deleting some prefix or/and some suffix from both of the input sequences (this operation is usually called *clipping*). To sum up, when doing local alignment, we search a score-maximizing transformation of arbitrary substring of the first sequence to an arbitrary substring of the second sequence. Similarly we define also a *semiglobal alignment*, where either a prefix or a suffix is deleted from each sequence, but not both at the same time.

A wise choice of an appropriate scoring system is the basic precondition for obtaining biologically relevant results. Many different scoring systems exist, often specific to particular tasks (e.g., homology search of short sequences, or alignment of Illumina NGS reads). Two simplest systems are derived from the *Hamming distance* (match 1, mismatch -1, no insertions or deletions allowed), and from the the *edit distance* (match 1, mismatch -1, insertion -1, deletion -1). Even though these systems can be very useful in simple applications, more complex systems are usually needed for NGS data.

In more general scoring systems, the scores for substitutions and matches are usually defined using a *substitution matrix* (score matrix, scoring matrix) M, where M_{ab} is the score of the substitution $a \to b$. Several standard substitution matrices have been designed, based on different evolution and sequencing models (see, e.g., [126]). These matrices are typically symmetrical, as for instance BLOSUM62 [126] or PAM120 [127]; but not necessarily [128]. As follows from theory [127], every linear scoring system defined using a substitution matrix directly corresponds to a model with independent mutations with exact frequencies. These frequencies can be derived from the substitution matrix; and on opposite, for given frequencies of mutations, such a matrix can be constructed (see, e.g., [124, Chapters 1 and 2]). In consequence, under the widely accepted simplified models, the best scoring systems uses a substitution matrix computed from the substitution frequencies observed in good alignments.

Good scoring systems should also support insertion and deletions. Their score is usually defined using an affine function of their length so, for instance, starting a new insertion is more expensive than its extension.

3.1.1 Alignment using dynamic programming

The original methods for sequence alignment from the 1970s and 1980s fully relied on dynamic programing. These methods are also the only ones which can find the optimal solutions. *Needleman–Wunsch algorithm* [129] solves the problem of the global alignment, *Smith-Waterman algorithm* finds the optimal local alignment [130], and *Gotoh algorithm* [131] finds the optimal alignment for a scoring system with the affine gap penalties. All three algorithms are often referred jointly as *Smith-Waterman-Gotoh algorithms*.

The $O(m \cdot n)$ space and time complexity of the Smith-Waterman-Gotoh algorithms make them hardly applicable for long sequences such as mammalian reference genomes. In the following sections, we present various specialized heuristics for individual use-cases (such as alignment of NGS reads). Since these heuristics use the Smith-Waterman-Gotoh algorithms on certain small subproblems (e.g., ranking and comparing heuristically found alignments), we need their high-performance implementations and several techniques for speeding them up are widely used.

A first technique is based on exploiting properties of the modern CPUs. The implementations can be heavily cache-optimized and use the *Single Instruction - Multiple Data* (SIMD) instruction sets such as SSE2 or AVX2.

A second technique uses a restriction of the dynamic programming to a strip of width k around its diagonal in the dynamic programming table. Then the space and memory complexities decrease to $O(m \cdot k)$. The resulting algorithms, usually referred as *Striped-Smith-Waterman-Gotoh* [132], are suitable only for a limited, but in practice frequent, class of applications such as score computation in read mapping of Illumina reads, where certain upper bounds on deletions and insertions lengths can be established.

A third technique, based on using bit-parallel algorithms, is usually feasible only for simple score systems. The most prominent example is the edit distance, which can be computed in the bit-parallel way using the Myers bit-parallel algorithm [133], or using some of its modified versions [134, 135]. Another option is to quickly filter sequences with too low edit distance (using, e.g., shifted Hamming distance algorithm [136]) prior to the standard alignment. Besides Myers algorithm, many other bit-parallel algorithms with subtle differences have been designed; see, e.g., Wu-Manber algorithm [137], Hyyrö algorithm [138], or a particular bit-parallel algorithm for integer weights [139].

Many libraries provide implementation of the Striped-Smith-Waterman-Gotoh algorithms for the standard architectures, namely AAlign [140], ALP&FALP [141], DiagonalSW[1], Edlib [142], ksw from klib[2], Parasail [143], Seqan [144], SSW Library [145], and SWPS 3 [146]. There exist also several implementations for special computational infrastructure, namely for GPUs (CUDASW++ [147, 148, 149], GSWABE [150], MASA [151], NVbio[3] SW# [152], SWCuda [153]), for OpenMP (MASA [151]), and for Intel Xeon Phi (SWAPHI [154, 155]). GPU implementations exist also for Myers [156] and Wu–Manber [157, 158, 159] algorithms.

[1] https://sourceforge.net/projects/diagonalsw/
[2] https://github.com/AttractiveChaos/klib
[3] https://nvlabs.github.io/nvbio/

3.1.2 Heuristics for aligning a sequence against a database

In the 1980s, many influential large genomic and protein databases emerged (e.g., GenBank in 1982, European Nucleotide Archive in 1982, DNA Data Bank of Japan in 1986). Searching in these database was very time-consuming since the dynamic-programming algorithms scaled very badly with the amount of data.

The problem was partially alleviated by the programs FASTP [160] from 1985 (for searching protein sequences) and FASTA [161] from 1988 (for searching nucleotide sequences). Both of these programs have been eventually distributed within a single software package called FASTA. Remark that the FASTA format has its roots in this project. The FASTA project improved heuristics from [162] and finally provided easy to use package enabling relativelly fast querying even on IBM PC. Indeed, matching of several hundreds short amino-acid sequences against the entire National Biomedical Research Foundation library was a matter of minutes [160].

BLAST (Basic Local Alignment Search Tool), one of the most influential tools of computational biology ever, was published in 1990 [163] and quickly became a leading program for sequence comparison, staying extremely popular until today. At the time of writing this thesis, the original BLAST paper [163] has already more than 60,000 citations on Google Scholar whereas the main FASTA paper [161] has approximately 12,000. Compared to FASTA, BLAST provided a significant speed-up, high sensitivity, and rigorous statistical characterizations of reported alignments such as estimates on the E-value (as the first program).

Since the first release, BLAST has been continuously improved, see, e.g., Gapped BLAST [164], PSI BLAST [164], MegaBLAST [165, 166], or Magic-BLAST[4]. Also, its database is regularly updated (see its current version[5]).

3.1.3 Heuristics for genome-to-genome alignments

Much attention has been also devoted to genome-to-genome alignments. Since this problem is not directly related to this thesis, we provide only a list of several examples of programs that have been developed for this task: MUMmer [167], MUMmer 2 [168], AVID [169], BLASTZ [170], LAGAN [171], MUMmer 3 [172], LASTZ [173], Cgaln [174], FEAST [175], and LAST [176, 177].

3.1.4 Read mapping

Introduction. After the completion of the human genome in 2003 [187], a huge effort was invested into the development of new methods for sequencing and the Next-Generation Sequencing emerged. These technologies, sometimes also called massive parallel sequencing technologies, are able to massively sequence genomes and produce extremely huge amounts of so-called reads, i.e., sequenced fragments of the genome (see Chapter 2).

Even the first sequencers, producing reads of length 25 bp only, enabled detecting genomic variants. This could be done by aligning the reads to a reference sequence of a high quality, followed by looking at the differences. To do a rapid alignment of millions of short NGS reads, completely new algorithm had to be designed. Indeed, BLAST [163], already a very popular tool at this time, appeared to be simply too inefficient for this task (see Figure 3.1). The original heuristics (e.g., ELAND) were rather simple and often

[4] ftp://ftp.ncbi.nlm.nih.gov/blast/executables/magicblast
[5] ftp://ftp.ncbi.nlm.nih.gov/blast/db/

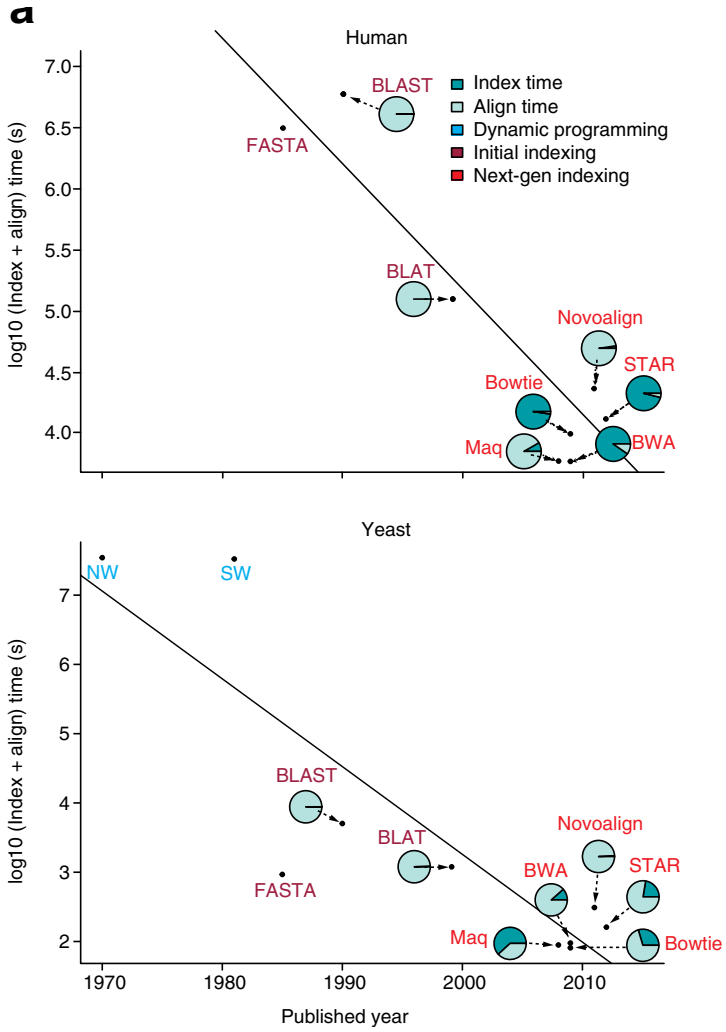


Figure 3.1: **Three generations for programs for read alignment.** Illustration of running times of three generations of programs for alignments of 75 bp long reads simulated from human and yeast genomes. The first generation contains dynamic programming-based algorithms (Smith-Waterman [130] and Needelman-Wunsch [129]), the second generation BLAST-like programs (FASTA [161], BLAST [163] and BLAT [178]), the third generation comprises read mappers based on hash tables (MAQ [179] and NovoAlign), suffix array (STAR [180]), and BWT-index (BWA-backtrack [181] and Bowtie [182]). Remark that BWA-backtrack, Bowtie and MAQ are already obsolete and have been replaced in practice by a new generation of read mappers such as BWA-MEM [183] and Bowtie 2 [184]. *©2016 Muir et al., CC BY 4.0, adapted from [185].*

relied on fast similarity search up to a fixed Hamming distance, usually one or two. The first generation of read mappers did not support insertions and deletions at all.

Almost immediately, a great boom of NGS read mappers started (see Figure 3.2). Since then, sequencing technologies are constantly improving and authors of read mappers must well react on this development. Many tools were release at a specific state of NGS technologies and became quickly obsolete. Only several tools have managed to stay modern until today, thanks to their good design and regular adaptation for modern data, e.g., the

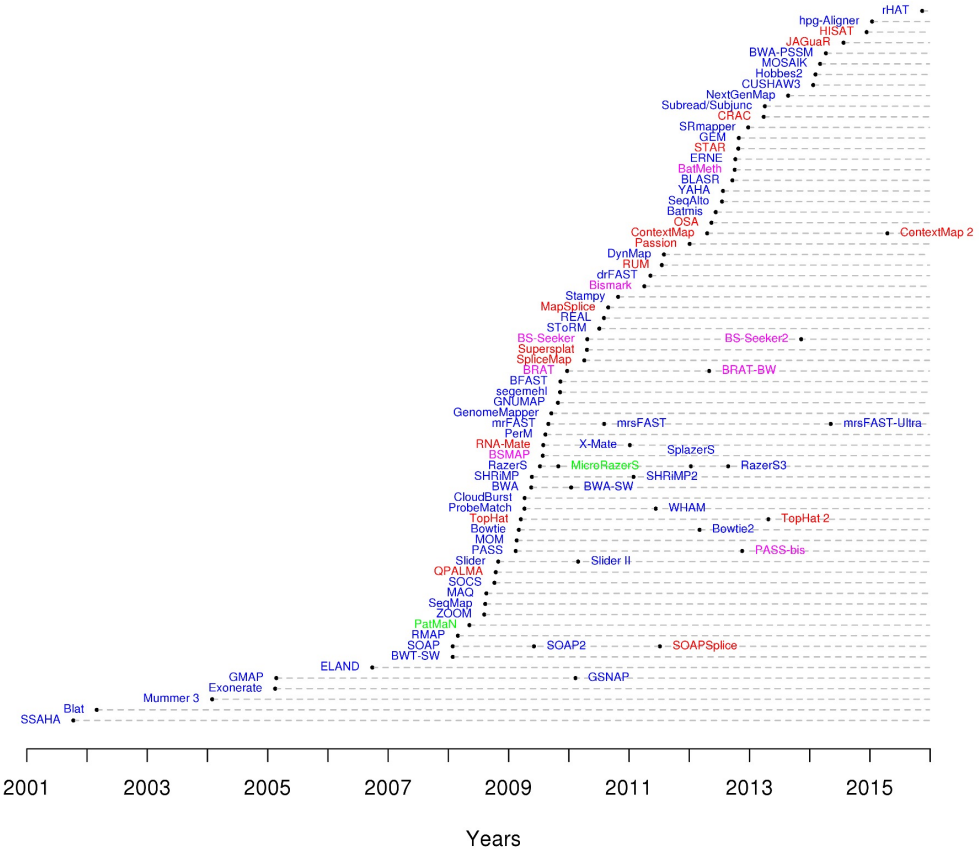


Figure 3.2: **Timeline with NGS read mappers.** Up-to-date version of a timeline with NGS read mappers from [186]. DNA mappers are displayed in blue, RNA-seq mappers in red, miRNA mappers in green, and BS-seq mappers in purple. ©*N. A. Fonseca,* ***http://www.ebi.ac.uk/~nf/hts_mappers/****.*

BWA [183, 188, 181] and Bowtie [182, 184] families of mappers.

A small revolution in read mapping has been raised recently by technologies such as Oxford Nanopore or PacBio. They produce long reads with high rates of sequencing errors, which have significantly changed the initial conditions of the read mapping problem. Mapping of long reads is now a very challenging task targeted by many research teams.

Note that release of SAM/BAM format [125] is an important milestone of read mapping. Before that, tools produced data in proper output formats, which required particular ad-hoc scripts for conversion in the analysis pipelines.

Challenges. There are several factors that make mapping a challenging task in practice. Genome sequences are highly repetitive and often contain several regions equally similar to a read [189]. Employing a sequencing technology producing long or paired-end reads can alleviate but not completely eliminate this difficulty. Reliability of mapping strongly depends on how much the sequenced genome differs from the reference. Close individuals (such as those of the same species) usually differ only weakly and highly variable regions tend to be rare. For distant individuals, evolutionary events (such as genomic

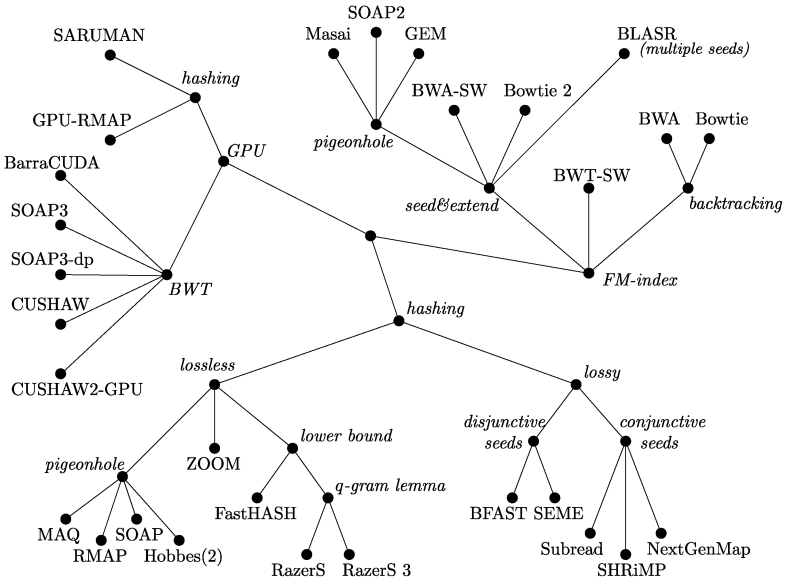


Figure 3.3: **Indexing strategies for read mapping.** *©2015 IEEE. Reprinted, with permission, from [47].*

rearrangements or gene duplications) may cause more substantial differences between the two genomes making it impossible to deduce the true origin of some reads using similarity search.

Sequencing errors constitute another major obstacle. Their rate strongly depends on the employed technology – from Illumina HiSeq producing reads of 100 bp with about 1% of errors to PacBio SMRT and Oxford Nanopore producing reads of up to several tens of thousand bp with about 15% of errors – and on the experimental conditions. Furthermore, some reads may come from regions absent in the reference sequence or may result from contamination by DNA from other genomes. Such reads should be detected and annotated by the mapper, i.e. marked as unaligned or as aligned with low quality. Finally, the sheer volume of input data makes of mapping a computationally demanding task constraining the choice of the underlying algorithm. For instance, whole genome sequencing projects of the human genome of 3 Gbp often use more than 30× coverage.

Algorithmic overview. Modern read mappers usually compute an index for the reference sequence and then map a stream of the reads against this index. They usually proceed by a so-called *seed-and-extend* paradigm. For every read, the mappers identifies regions highly similar to the reference (the *seeding* step) and then it extends them into full alignments (the *extending* step). The mapper then reports one or several best suiting alignments of the read, usually accompanied by the alignment score and the mapping quality (the probability that the reported alignment position is wrong).

Two basic indexing approaches are widely used in mapping. In the first approach (implemented, e.g, in NovoAlign[6], SHRiMP 2 [190], or BFAST [191]), the mapper stores a hash table (for the seeding step) and the reference (for the extending step, which is done usually using the (Striped-)Smith-Waterman-Gotoh algorithms). The hash-table-based

[6] http://www.novocraft.com/products/novoalign/

mappers are usually very sensitive, but occupy a lot of memory. The second approach relies on compressed full-text indexes [192], usually on the BWT-index [193] (see Section 4.3). This single structure can be used both for seeding and extending. Moreover, the associated memory footprint is small compared to hash tables, so advantages of BWT-based mappers are obvious and a majority of the most popular state-of-the-art mappers belong to this group, e.g., BWA-MEM [183], Bowtie 2 [184] or GEM [194]. For a tree of popular indexing strategies, see Figure 3.3.

Many different techniques for seeding exist [195, 196, 197]. They can be based, e.g., on shared contiguous k-mers of fixed length (e.g., MAQ [179]), shared spaced k-mers defined by a spaced seed (e.g., SHRiMP 2 [190]), shared spaced k-mers of variable lengths defined by an adaptive spaced seed (e.g., LAST [176]), or maximum exact matches (e.g., BWA-MEM [183]), to mention at least several examples. To conclude the indexing strategies, let us remark that the traditional seeding techniques work well with the short-read technologies, but become insufficient for long reads with a high rate of sequencing errors. This calls for novel seeding approaches [198, 199] or even for aligning the Nanopore signal directly to the reference without base calling[7].

Features of mappers. According to supported technologies, mappers can be classified to short read mappers (e.g., BWA-backtrack [181]), long read mappers (e.g., GraphMap [198]), and those which can deal well with both types of reads (e.g., BWA-MEM [183]). Another distinction can be done according to the type of produced alignments. Mappers can detect local alignments (e.g., Bowtie 2 [184]), or global alignments (e.g., Bowtie 2 [184]), or switch automatically between the both types of alignments (e.g., BWA-MEM [183]). Many particular types of reads may require a special treatment, e.g., RNA-seq reads, BS-seq reads, or reads in color space. Support for these types of reads is another important feature.

Two modes of mapping exist, *all-mapping* and *best-mapping*. The all-mapping mode is based on reporting all alignments having similar score to the best alignment, or on reporting all alignments with a score within a certain threshold (e.g., with edit distance ≤ 3). Mapping in the best-mapping mode aims at finding the best alignment only. The latter approach can save computational time, but many relevant alignments might be missed. To report reliable mapping qualities, the mapper has to compute at least two best alignments in any case.

Many other features would deserve a discussion (techniques of parallelization, support for pair-end and mate-pair reads, support for SNPs, ways of treatment of alternative and decoy sequences, etc.). For more information about them, please see specialized reviews (e.g., [200], [201], [202], and notably [47]).

BWA-MEM as an example of a mapper. BWA-MEM [183] is nowadays one of the most popular mappers so we use it as an example of a good mapper. Its success stems from an well-designed algorithm working well for many sequencing technologies, from its highly efficient implementation, and from a good availability and portability of the code. Moreover, BWA-MEM has a a very intuitive interface, comprehensible documentation, and perfect user support. Anecdotically, the BWA-MEM's manuscript [183] was rejected and it exists only as an arXiv preprint until today.

The underlying indexing engine (used also in BWA-backtrack [181] and BWA-SW

[7] http://simpsonlab.github.io/2015/04/08/eventalign/

[188]) is based on a high-quality implementation of BWT-index [193], that was originally adopted from BWT-SW [203] and subsequently highly optimized.

The BWA-MEM mapping algorithm proceeds by the following steps. It first identifies maximum exact matches (MEM) for every suffix of a read, which are then used as seeds. Too long seeds are shortened, close seeds are chained together, and certain shorter chains are filtered out. Subsequently, the Smith-Waterman algorithm is used to extend the seeds and to connect those seeds, which are proximal in the reference. BWA-MEM automatically chooses between global and local alignment, preferring global. If paired-end reads are used, a special pairing strategy is used to favor alignments in a correct distance – this distance is computed on-the-fly from first high-quality alignments. As the last step, the best computed alignments are reported.

List of mappers. Here we provide a list of mappers existing at the time of writing this thesis. First we mention the mappers, which use classical approaches (majority of popular tools belong to this category). Then we enlist the mappers designed for special data (e.g., for BS-seq), the mappers using special computational approaches (e.g., graph mappers), and the mappers designed for special computational infrastructures (e.g., GPU mappers).

Our list is complementary to other existing lists, e.g., to the list on Wikipedia[8], the list on OmicsTools[9], to the list maintained by N. A. Fonseca[10]), or to the list in [204]. Note that mappers are sorted in a descending order with respect to the year of the corresponding publication (or a preprint).

1) **Standard read mappers.**

i) **BWT-based.** The standard read mappers based on BWT-index (FM-index), on suffix arrays, or on suffix trees are: LAMSA [205], MUMdex Aligner [206], ERNE 2 [207], Mapper [208]; ALPHALPHA [209], GRyCAP [210], YARA [211]; ARYANA [212], BWA-PSSM [213]; CUSHAW 3 [214, 215], HPG Aligner [216], RandAl [217]; BWA-MEM [183]; Isaac Genome Alignment Software [218], Masai [219], NucBase [220]; BatMis [221], BLASR [222], Bowtie 2 [184], GEM [223, 194, 224], CUSHAW2 [225]; LAST [176]; BWA-SW [188]; Bowtie [182], BWA-backtrack [181], SOAP 2 [226], and OASIS [227].

CLC Mapper [228], DAMAPPER-BWT[11], GENALICE MAP[12], TMAP[13], and Vmatch[14] are not accompanied by any manuscript.

ii) **Hash table based.** The standard mappers based on hash tables are PEMapper [229]; FSVA [230], GraphMap [198], Hobbes 3 [231], MECAT [232], NanoBLASTer [233], NextGENe Sequence Alignment tool [234]; AMAS [235], BitMapper [236], GensearchNGS [237], JVM [238], merAligner [239], rHAT [240], SNPwise [241]; HIVE-Hexagon [242], Hobbes 2 [243], LRD [244] Mosaik [245], MrsFAST-Ultra [246]; SEME [247], Subread [248], SRmapper [249]; ERNE [250], Hobbes [251], RazerS3 [252], rRNA [253, 254], SeqAlto [255], YAHA [256], WHAM [257]; AGILE

[8] https://en.wikipedia.org/wiki/List_of_sequence_alignment_software#Short-Read_Sequence_Alignment

[9] https://omictools.com/read-alignment-category

[10] http://www.ebi.ac.uk/~nf/hts_mappers/

[11] https://github.com/gt1/damapper_bwt

[12] http://www.genalice.com/product/genalice-map/

[13] https://github.com/iontorrent/TS/tree/master/Analysis/TMAP

[14] http://www.vmatch.de/

[258], SHRiMP 2 [190], SNAP [259], Stampy [260]; FANGS [261], GSNAP [262], GASSST [263], MrsFAST [264], Q-Pick [265], REAL [266]; BFAST [191], GNUmap [267], MapNext [268], MOM [269], MPSCAN [270], MrFAST [271, 272], PASS [273], PerM [274], ProbeMatch [275], RazerS [276], SHRiMP [277]; MAQ [179], RMAP [278, 279], SeqMap [280], SOAP [281], ZOOM [282]; GMAP [283], YASS [284]; SSAHA 2 [285]; PRIMEX [286]; BLAT [178]; SSAHA [287].

Anfo[15], BitMapper 2-CPU[16], DAMAPPER[17], ELAND, ISAAC 2[18], NovoAlign[19], Ngmlr[20], and SMALT[21] do not have any manuscript.

iii) **Others.** Hybrid-index based mappers, HIA [288] and YOABS [289], combine both hash tables and a BWT-based indexes. Mappers Slider II [290] and Slider [291] use merge sorting. Tanoti[22] is built on top of BLAST [163].

2) **Mappers for special data.** RNA-seq reads are spliced, in BS-seq reads every non-methylated C is converted to T, and color reads use a completely different encoding. All these types of reads call for special mapping techniques.

i) **RNA-seq reads.** The following mappers are specifically designed for RNA-seq data: HPG Aligner [292], POMP [293], RapMap [294]; BBmap [96], ContextMap 2 [295, 296], HISAT [297]; FANSe 2 [298], JAGuaR [299]; CRAC [300], OLego [301], STAR [180], TopHat 2 [302]; ContextMap [303], FANSe [304], OSA [305], PASSion [306], RNASEQR[307]; ABMapper [308], RUM [104], SOAPsplice[309]; GSNAP [262], HMMSplicer [310], MapSplice [311], PALMapper [312], SpliceMap [313], Supersplat [314]; CASHX [315]; QPALMA [316]; SSAHA 2 [285]; SSAHA [287]; EST_GENOME [317].

Mappers HISAT 2[23] and ReadsMap[24] do not have any manuscript. For a comparison of RNA-seq read mappers, see [318] and [319].

ii) **BS-seq reads.** The following mappers are specifically designed for BS-seq data: BRAT-nova [320], WALT [321]; Bison [322], GPU-BSM [323], TAMeBS [324]; BatMeth [325] (it support also SOLiD reads), BISS [326], Merman [327] (a component of the BSmooth package), RRBSMAP [328], Segemehl [329]; Bismark [330]; PASH 3 [331]; BSMAP [332], RMAPBS [279], VerJInxer [333], and BSmapper[25]. The following standard mappers support BS-seq mapping: PEMapper [229], ERNE 2 [207], HPG-Methyl [334], GNUMAP-bs [335], PASS-bis [336], BRAT-BW [337], ERNE [250], LAST [110], GSNAP [262], legacy versions of MrsFAST [264], and NovoAlign. These pipelines use other mappers to map BS-seq reads: Bisulfighter [338], BS-Seeker 2 [339], MethylCoder [340], BS-Seeker [341], and B-Solana [342] (for SOLiD reads). There also exist several comparison of BS-seq read mappers [343, 344, 345, 346]. Especially [346] can be highly recommended. Mapping of BS-seq reads brings many problems, e.g., it is hard to choose a reference [347].

[15] https://bioinf.eva.mpg.de/anfo/
[16] http://home.ustc.edu.cn/~chhy/BitMapper2.html
[17] https://github.com/gt1/DAMAPPER
[18] https://github.com/Illumina/isaac2
[19] http://www.novocraft.com/products/novoalign/
[20] https://github.com/philres/ngmlr
[21] http://www.sanger.ac.uk/science/tools/smalt-0
[22] http://www.bioinformatics.cvr.ac.uk/tanoti.php
[23] https://github.com/infphilo/hisat2
[24] http://www.softberry.com/berry.phtml?topic=products
[25] https://sourceforge.net/projects/bsmapper/

iii) **Complete genomics reads.** We are aware only of SirFAST [348] and RTG Core[26].

iv) **MiRNA.** Tailor [349]; MicroRazerS [350]; PatMaN [351]; NovoAlign. For an evalutation of methods, see [352].

v) **Color space reads.** Many nucleotide space mappers support also color space, namely CUSHAW 3 [214, 215], Mosaik [245]; drFAST [353], SHRiMP 2 [190], X-MATE [354]; mrFAST-CO [264], SToRM [355]; BFAST [191], Bowtie [182], BWA-backtrack [181], PerM [274], SHRiMP [277]; MAQ [179], ZOOM [282]. Mappers supporting only the color space are RNA-MATE [356] and SOCS [357].

vi) **CRISPR.** We are aware only of CRISPRAlign [358].

vii) **Mapping to protein references.** While the vast majority of read mappers is designed for DNA-DNA alignment, in certain biological applications read alignment against a protein reference or a protein-coding DNA reference may be needed. This problem is algorithmically partially different from DNA-DNA read mapping because an alphabet of higher cardinality is used (note that there are 20 distinct amino acids in genetic code) and reads must be translated to 6 different frames (3 in each direction). Alphabet reduction may used to increase sensitivity or to decrease memory requirements. Reduction scheme is usually derived empirically. There exist currently three tools which can be called read mappers to a protein reference, namely DIAMOND [359], RAPSearch2 [360], and RAPSearch [361]

viii) **Aligning of unaligned reads.** There exists a class of methods for treating reads, which could not be mapped using standard techniques [362, 363, 364].

3) **Special computational approaches.** Here we provide a list of mappers, which use non-traditional algorithms for read mapping,

i) **Graph mappers.** Among researchers there is a wide consensus that the traditional concept of sequences as reference structures has become insufficient [365, 366]. As a solution, references sequences could be replaced by *reference graphs* (sometimes called *variation graphs*) since they can alleviate most of the problems, for instance, substantially more reads may get mapped perfectly [366].

Generalization of mapping and variant calling algorithms is not straightforward and belongs to the main challenges of the current bioinformatics. For instance, even simple concepts such as genomic intervals become difficult to handle [367]. Mapping to graphs is complicated to formalize mathematically [368, 369]. Also indexing graphs is highly challenging from algorithmic point of view [370, 371].

The first step from linear references to graph references are the SNP-tolerant mappers such as Mosaik [245], mrsFAST-Ultra [246], VATRAM [372, 373], SNPwise [241], GSNAP [262] and NovoAlign. They allow to incorporate a database of SNPs into the reference, e.g., dbSNP[374]. Slightly more variation can be encoded into so-called bubbles in BWBBLE [375]. GenomeMapper [376] aligns reads to a set of similar genomes representing the graph. [375] Then there exist two mappers aligning reads to de-Bruijn graphs, BGREAT [377] and deBGA [378].

Two recently introduced mappers proceed by aligning reads to reference graphs, inferring a new high-quality reference sequence, and remapping all reads to the obtained reference. Gramtools [379] uses a dedicated method for mapping reads to

[26] https://github.com/RealTimeGenomics/rtg-core

a population graph. Then it infers a new linear reference, and remaps reads to this reference. A similar approach is used in CHIC Aligner[27]. The main conceptual difference from Gramtools is a usage of a so-called kernel string, i.e., a string created from an LZ77-based index of concatenated references. The reads are first mapped to the kernel strings using an arbitrary standard mapper and the obtained alignments are used for inferring the new reference.

The most promising project, which is still under development, is a mapper supporting general graphs called VG[28]. For further information on methods for mapping to graphs, see [365].

ii) **Dynamic mapping.** Dynamic mapping, i.e., an approach when the reference sequence is continuously updated based on the already computed alignments, has been little studied so far and only two partial experimental solutions with naive mapping algorithms are available [380, 381]. See Part II for our work on this topic.

iii) **Compressive mapping.** Compressive mapping [382, 383] aims at sublinear scaling with growing genomic data sets.

iv) **Mapping using MinHash.** Another interesting direction of read mapping exploits MinHash sketches [384, 123], which are a generalization of minimizers [385]. To the best of our knowledge, there are currently only five mappers using this approach, namely MashMap[386] (a mapper for Oxford Nanopore and PacBio reads), Minimap [387] (a mapper for Oxford Nanopore reads), BALAUR [388] (a privacy preserving read mapper), VATRAM [372, 373] (an experimental variant tolerant mapper), and Minialign[29] (a mapper derived from Minimap).

v) **Numerical mapping.** Read mapping could proceed by Fourier transform and compute the alignment in the frequency domain [389], similarly to what is widely used, e.g., for image registration [390] (image registration can be viewed as a certain analogy of read mapping). Only few works [391, 392, 393] considered such an approach and no real mapper using this approach exists so far.

vi) **Real-time mapping.** Real-time mapping aims at mapping reads during the process of their sequencing. To the best of our knowledge, HiLive [394] is currently the only mapper supporting such a mode.

4) **Mappers for special computational infrastructures**

i) **GPU (Graphic Processing Unit).** The mappers using Graphical Processing Units are GPU BWA-MEM [395]; CUDAlign 4.0 [396], MAPSEQ-G [397]; Arioc [398], GRyCAP [210]; BWA-meth [399], CUDAlign 3.0 [400], CUSHAW2-GPU [401], GPU-BSM [323] (a mapper for BS-seq data), MaxSSmap [363], PEANUT [402], RAMICS [403]; NextGenMap [404], SOAP3-dp [405], G-DNA [406], Masher [407]; BarraCUDA [408, 409], CUSHAW [410], G-Aligner [411], GPU-BWT [412], SOAP3 [413]; SARUMAN [414]; CUDAlign [415], GPU-RMAP [416], and MUMmerGPU [417]. BitMapper 2[30] has not been published, yet. For a review of GPU mappers, see [418]

[27] https://www.cs.helsinki.fi/en/gsa/chica/
[28] https://github.com/ekg/vg
[29] https://github.com/ocxtal/minialign
[30] http://home.ustc.edu.cn/~chhy/BitMapper2.html

ii) **MapReduce.** The following mappers use MapReduce: SparkBWA [419]; BigBWA [420]; DistMap [421] (a toolkit supporting 9 different mappers); SEAL [422], CloudAligner [423]; CloudBurst [424]; and BlastReduce [425].

iii) **MPI (Message Passing Interface).** mpiBWA[31] [426], pBWA [427] and pFANGS [428] parallelize BWA-MEM, BWA-backtrack [181] and FANGS [261], using MPI respectively. pMap[32] can parallelize Bowtie [182], BWA-backtrack [181], GSNAP [262], MAQ [179], RMAP [278], and SOAP [281]. G-DNA [406] is a GPU-based mapper with a support for parallelization using MPI.

iv) **FGPA (Field-Programmable Gate Array).** The FGPA mappers are FHAST [429]; FGPA-SRA [430]; Shepard [431]; and VelociMapper[33].

v) **Others.** Read mapping with PIM (Processing in Memory) has been studied in [432] and [433]. The B-MIC [434] and MICA [435] mappers exploit MICA (Many Integrated Core Architecture). RAMPS [436] and Convey BWA [437] were developed for the Convey HC-2 hybrid core computing system. Also particular accelerated variant BWA-MEM for this system has been published [438].

Note that several toy mappers, mostly unsuitable for practical applications, have been developed within Topcoder Marathon Match DNAS1[34].

3.2 Alignment-free methods

Alignment-free methods estimate the similarity of sequences from their composition in words or patterns, usually k-mers. These approaches, studied from 1980's [439], have become very popular for NGS data processing for several reasons. First, they are much more computationally efficient than the alignment-based techniques. Second, they can compare unknown sequences provided that we know their composition. For instance, we can compare two genomes based on their k-mer compositions computed directly from NGS reads without their assembling. Similarly, a composition of an evolution ancestor can be derived from the composition of its descendants, which can be useful, e.g., in taxonomic classification of NGS reads.

Most of alignment-free methods work with so-called frequency dictionaries, i.e., compare two sequences using distances between vectors of abundancies of all their subwords, or between vectors of abundancies of the subwords of a fixed length k called k-mers. These vectors are usually referred to *frequency vectors*. Many different distances exist, see, e.g., [440, 441, 442] for more information. Some of them even take into account the qualities of individual bases [443, 444]. Note that for a large enough k (but still much shorter than the original sequence), the frequency vectors contain sufficient information to reconstruct the original sequence (see, e.g, [445]). Alignment-free methods are related to machine learning, especially kernel methods (see, e.g., [446]). Remark that stringology techniques can be applied to compute machine learning kernels [447].

In our work, the traditional alignment-free methods are hardly applicable for two practical reasons. First, even though many efficient methods for computing frequency vectors through k-mer counting have been recently published (e.g., [448, 449, 450, 451,

[31] https://github.com/fredjarlier/mpiBWA
[32] http://bmi.osu.edu/hpc/software/pmap/pmap.html
[33] http://www.timelogic.com/catalog/799/velocimapper
[34] http://goo.gl/7W7Nn2

452, 453, 454, 455, 456, 457]), obtained dictionaries would still occupy too much memory for k's of practical interest (usually ≥ 15). Second, associated massive operations with floating point numbers are very expensive. Both problems become more critical when we need to compare a sequence to many sequences at the same time (e.g., in metagenomic classification of NGS reads against a large database of thousands of reference genomes and their taxonomic ancestors). Therefore, we have to give up on frequency vectors and to rely only on the information whether a given k-mer is present in a sequence, or not. The resulting indexes, either simple hash tables (such as in Kraken [458]) or advanced full-text indexes (such as in ProPhyle [Chapter 12]), are extremely fast and provide reasonable memory footprints. We elaborate more on this in Part III.

Up to now, we considered methods for arbitrary biological sequences. Now let us have a look at the relation of read mapping to alignment-free techniques. This has been recently formalized [459, 460] through a concept of *pseudoalignments*. The central idea is that in many real scenarios, we only need to know in which sequence – a genome, a chromosome, or a transcript – the read originates from but we can relax on computing its precise position in that sequence. Pseudoalignments can be then computed very efficiently, much faster than alignments. Indeed, we can, for instance, match very quickly reads' k-mers against a colored de-Bruijn graph [459].

Part III is related to the concept of pseudoalignments in the following way. Prophyle, the metagenomic classifier which we present in Chapter 12, classifies reads to a taxonomic tree by matching them against a collection of k-mer sets, assigned to individual nodes of the tree. This collection-based approach is partially similar to transcript de-Bruijn graphs [459]. In addition to computing the pseudoalignments, we also study how to estimate a score of the corresponding alignment (which we do not compute) using the hit count or the coverage criterion in combination with spaced seeds (see Chapter 11).

3.3 Spaced seeds

Many k-mer-based methods for sequence comparison, both alignment-based and alignment-free, can be improved using so-called *spaced k-mers*. While "traditional" *contiguous k-mers* are taken as substrings of length k from the original string, spaced k-mers are extracted using a bit-mask specifying positions of letters to be taken. For instance, let us consider a string `ACGCC`$\cdots$ and a bit-mask 1101. Then, instead of k-mers `ACG`, `CGC`, `GCC`; we can work with spaced k-mers `AC-C`, `CG-C`, ... The bit-masks used for this purpose are usually referred to *spaced seeds* and they are traditionally encoded using '`#`' or '1' for matching positions, and '`-`' or '0' for "joker" positions. The number of `#`'s is called the *weight* and the seed's length is called the *span*. So the seed from the example can be also written as `##-#` and it has span 4 and weight 3. Within this framework, contiguous k-mers are generated using spaced seeds consisting of `#`'s only so, for instance, k-mers of length 3 correspond to spaced seed `###` with span and weight both equal to 3.

Spaced seeds first appeared in 2000s as a concept improving seed-and-extend sequence comparison in lossless [461, 462] and lossy [463] settings (for a general information about seeding techniques, see, e.g., [196] and and the references therein). Later they were shown to improve alignment-free methods as well, e.g., estimation of phylogenetic distances [464, 465], metagenomic classification [3], or string kernels for machine learning [466, 467, 468]. Many extensions of spaced seeds were suggested, e.g., vector seeds [469], indel seeds [470], subset seeds [471], neighbor seeds [472, 473], or adaptive seeds [176]. For readers interested in this domain, we recommend Laurent Noé's exhaustive list of seed literature [474]

(currently containing 150 papers).

Whereas the processing of contiguous k-mers is usually straightforward, as the only parameter is k and many indexing strategies are available (e.g., BWT-index), spaced seeds bring two challenges: designing good seeds and indexing spaced k-mers. We elaborate on this below.

Many bioinformatics programs use variants of spaced seeds, e.g., PatternHunter [463, 475, 476] and YASS [284] for similarity search; Zoom [282], ELAND, BFast [191], Perm [274], PLAST [477], SToRM [355], LAST [176], SHRiMP 2 [190], Seed BLAST [478], SASS [479], Diamond [359], GraphMap [198] and WALT [321] for similarity search and read alignment, or seed-Kraken [3] and CLARK-S [480] for metagenomic classification. From the perspective of the final program, a spaced seed is usually a parameter, which is either specified by the user or is hardcoded in the program. The seed should be carefully designed using some method described below or using some specialized program for seed design, such as Iedera [481] and RasBhari [482].

We now briefly overview main classes of spaced seeds and associated computational problems.

Seeds for lossless text filtration. Spaced seeds for lossless text filtration are formalized using so-called (m, k)-problem. For a given m and k, a seed Q solves the (m, k)-problem if it can detect strings in Hamming distance $\leq k$ among strings of length m. For a precise mathematical definition of the (m, k)-problem, see Appendix A. The efficiency of lossless filtration is measured by the selectivity, which corresponds to number the weight of the seeds. Roughly speaking, our aim is to maximize this value in order to reduce the false positive rate of the resulting filter.

Lossless seeds were first introduced in [461, 462]. It was shown [483, 484] that deciding whether a seed solves an (m, k)-problem is an NP-complete task. Asymptotic properties of optimal lossless seeds, i.e., those with maximal possible weight, were described in [485, 486] and [487, 488]. Various mathematical objects were shown to be related to lossless spaced seeds, namely perfect rulers [489, 490], quadratic residues [491] (for their usage within the lossy framework, see [492]), or Steiner systems [493].

From the practical viewpoint, lossless seeds are of less importance than lossy seeds, but they have remarkable mathematical properties. It has been observed [485, 486, 494, 4, 493] that good seeds tend to be very regular, often being a repetition of a short pattern. As we show in Appendix A, this regular structure is a consequence of the fact that lossless seeds form regular languages generated by certain de-Bruijn graphs. Periodic seeds are useful in practice as they simplify their indexing. This fact is exploited, e.g., in mappers PerM [274], LAST [176] and WALT [321].

Seeds for lossy text filtration. *Lossy seeds* were first introduced in PatternHunter [476, 475, 463]. While lossless seeds guarantee that all strings up to a specified Hamming distance will be found by filtration, lossy seeds may miss a fraction of them, but the associated false positive rate can be kept under control. This property is desirable in practice, therefore, most of the seed-based mappers mentioned at the beginning of this section use lossy seeds. A seed extension of BLAST, called Seed BLAST [478], uses this approach as well.

The efficiency of lossy filtration is measured by two parameters called *sensitivity* and *selectivity*, where the former measures the rate of false negatives and the latter the rate of false positives. While the sensitivity of a seed can be roughly estimated from its weight,

the computation of the sensitivity is a complicated task (see, e.g., [495, 496] and the references therein). Moreover, this number depends on the used probabilistic model (e.g., Bernoulli). Various methods for its estimation for different probabilistic models have been studied see, e.g., [497, 498, 499].

It has been shown that sensitivity correlates with the so-called overlapping complexity [500], which can be easily computed. Programs SpEED [500] and RasBhari [482] exploit this property in combination with the hill climbing algorithm.

In addition to SpEED and RasBhari, there exist two more tools, Iedera [481] and Mandala [501]. In particular, Iedera can be highly recommended as it implements various probabilistic models and provides a good performance in designing spaced seeds of different types.

Seeds for classification. Spaced seeds have a potential to improve k-mer-based classification methods, notably kernel-based methods [468] and alignment-free metagenomic classification. In Chapter 11, [3] we show that spaced seeds combined with the hit number and the coverage improve the accuracy of metagenomic classification.

Design of seeds for classification has not been much studied, yet. It can be based on maximization of the Pearson or Spearman correlation between alignment score, and the number of hits or the coverage. Such a design is supported, e.g., by Iedera [481].

Methods for alignment-free phylogeny estimation using spaced k-mers [464, 465] are related to the classification problem because they also implicitly rely on estimates of alignment score from the hit count. Note that metagenomic classification has also been studied from the viewpoint of discriminative spaced k-mers [502].

Clustering. Spaced seeds have also been used for clustering of NGS reads [503] in order to reduce time and memory requirements of genome assembly. The authors of [503] provided a method to design so-called *block spaced seeds* for the purpose of clustering.

Chapter 4

Data structures for NGS data analysis

In this chapter, we shortly present main data structures relevant to our work. In general, selecting an appropriate data structure and its implementation is a non-trivial task. The choice of data structure may strongly influence the performance of the program as well as obtained results. This choice has to be done carefully and take into account properties of the underlying problem and possible size of the data (e.g., many programs are intentionally limited to run on model genomes such as the human genome).

4.1 Hash tables

A hash table is a data structure to represent associative arrays storing collections of $(key, value)$ pairs. The value of the hash function applied to the key is used as the index in the table. Hash tables have been widely studied in computer science, and various hashing schemes are known in the literature. Several mainstream libraries for hash tables are widely used in bioinformatics, such as khash (part of klib[1]), `std::unordered_map` (part of the C++ standard library), `boost::unordered_map` (part of Boost[2]), or Google SparseHash[3].

4.2 Classical full-text indexes

In this section, we present two classical text indexes: suffix trees and suffix arrays. Even though we do not use them directly in our software, they are predecessors of the BWT-index which has probably been the most widely used index in bioinformatics for the last years. Both of them constitute an index for a text string T supporting string matching queries: retrieve all occurrences of a given pattern P in T. The number of occurrences of P in T is usually denoted by occ.

Suffix tree. *Suffix tree* [504] is one of the most studied data structures in stringology. The suffix tree is defined as the compacted *trie* of all suffixes of the input string. Suffix

[1] https://github.com/attractivechaos/klib
[2] http://www.boost.org/
[3] https://github.com/sparsehash/sparsehash

trees can be constructed in linear time and space such that searching for a pattern P can then be done in time $O(|P|)$, and reporting all occurrences of P in time $O(|P| + occ)$.

As a pointer-based structure, the suffix trees need $O(n)$ computer words but $O(n \log(n))$ bits of memory. Unfortunately, the resulting memory requirements are so high that suffix trees are now rarely used in real programs. One of a few exceptions is, e.g., MUMmer [172]. To illustrate the memory problem on a practical example, MUMmer would need more than 45 GB for the human genome (in practice, MUMmer fails to run on the human genome due to limits on the maximal string length). For more information and references on suffix trees, we refer e.g. to [504],[505, Chapter 8]. Other variants of suffix trees have been studied such as *compressed suffix tree*, *truncated suffix tree*, or *affix trees*.

Suffix array. *Suffix array* [506] is another very popular data structure. Unlike suffix trees, suffix arrays are widely used in practice but still suffer from high memory consumption. It has been shown in [507] that every algorithm that uses the suffix tree can systematically be replaced by an algorithm that uses a particular variant of suffix arrays so that it solves the same problem with the same time complexity. Therefore, suffix arrays are capable to fully replace suffix trees in practical applications.

The suffix array is a permutation of the positions of T taken in the lexicographical order of corresponding suffixes. Searching for a pattern P corresponds to identifying those suffixes of T which have P as a prefix. Since the suffixes are sorted lexicographically, the suffixes prefixed by P are consecutive in the suffix array and form an interval. Its borders L_P and R_P can be found by binary search, with time complexity $O(|P| \log(|T|))$. If we are provided the longest common prefix array (LCP) (i.e., the array of lengths of two consecutive prefixes after their sorting), the complexity of search can be reduced to $O(|P| + \log |T|)$ [506].

Several variants of suffix arrays have been introduced such as *dynamic suffix arrays* [508] or *spaced suffix arrays* [509]. For an overview of state-of-the-art methods of suffix array construction see [509]. Implementations suitable for practical purposes are, e.g., SDSL-Lite [510], libdivsufsort[4], or QSufSort [511, 512].

4.3 BWT-index

Burrows-Wheeler transform. The *Burrows-Wheeler transform* [513] (commonly abbreviated as BWT) is a textual transform widely used for compression (e.g., in bzip2[5]) and for indexing, in particular in the so-called BWT-index (e.g., in BWA [181]). These applications of the Burrows-Wheeler transform (BWT) are possible thanks to its reversibility and properties of the resulting strings such as grouping identical letters together.

To transform a string, a special character '$' is appended to its end and all its cyclic shifts are sorted lexicographically. The last column of the matrix having these shifts in its rows (the so-called Burrows-Wheeler matrix) is the resulting BWT. The BWT can be easily extracted from the suffix array constructed with any of the algorithms from the previous section.

Many aspects of BWT have been studied including its relations to combinatorics on words [514] (e.g., relations to Sturmian words [515], to palindromic richness [516], to balance properties of words [517], or to the Gessel-Reutenauer transformation [518]), its

[4] https://github.com/y-256/libdivsufsort
[5] http://www.bzip.org/

statistical properties (e.g., entropy [519]), or its combinatorial relationship with suffix arrays [520].

Specific variants and generalizations have been considered such as bijective BWT [521, 522, 523]; dynamic BWT [524], or bi-directional BWT [525]. For BWT construction algorithms specifically designed for NGS data, we refer to [526, 527, 528]. For additional general information about BWT, see [529] and references therein.

BWT-index. The *BWT-index* [193] (sometimes also called *FM-index*) is a full-text index combining BWT and suffix arrays. It has been shown [193] that with the BWT string B, a table $O(a, i)$ storing the number of occurrences of each letter a in the prefix of B of length i, and a table $C(a)$ storing multiplicities of letter a in B, it is possible to simulate the pattern search in the suffix array. Starting with the entire interval of the suffix array corresponding to the empty pattern, we obtain the resulting interval in $|P|$ steps by processing P right-to-left. The intervals of the suffix array are updated according the the following formula:

$$L_{aX} = C(a) + O(a, L_X - 1) + 1$$
$$R_{aX} = C(a) + O(a, R_X)$$

However, the positions of pattern occurrences have still to be retrieved from the resulting interval $L_P, \ldots, R_P$ of the BWT. This can be done using the suffix array. The main trick of the BWT-index is sampling the O and suffix arrays, leading to a smaller memory footprint compared to normal suffix arrays.

Many libraries implement the BWT-index, e.g., Seqan [144], SDSL-lite [510], or nvbio[6] (GPU implementation). A number of NGS read mappers are based on BWT-indexes (see Section 3.1.4) providing some efficient implementations. The implementations from BWA [181] (partially based on [203]) and Bowtie [182] are particularly popular choices.

Note that many variants of the BWT-index have been suggested including BWT-index for graphs [370, 371], BWT-index of alignments [530, 531], bi-directional BWT-index [532, 533, 534], relative BWT-index [535], or dynamic BWT-index [536].

[6] https://nvlabs.github.io/nvbio/fmindex_page.html

Part II

Dynamic read mapping

Contents - Part II

Chapter 5

Context and motivation

5.1 Introduction

High-throughput sequencing technologies (Next-Generation Sequencing, commonly abbreviated NGS) made available terabases of DNA sequence data coming from numerous de-novo and resequencing experiments.

A central computational problem in NGS data analysis is inferring the original DNA sequence of an individual from a large set of NGS reads, i.e., short sequence fragments generated by a sequencing machine. This task is usually solved by mapping reads to a high-quality reference sequence or by a computationally expensive genome assembly algorithm.

Here we study genotyping by sequencing when a reference sequence is available. A typical pipeline for inferring the genotype of a sequenced individual consists of the following steps (see, e.g, [537, 538]). Reads obtained from a sequencer are mapped to the reference sequence and low quality alignments are discarded. Aligned reads are further sorted and possibly locally re-assembled or re-aligned around indels in order to better distinguish SNPs from indels. Systematic biases in base qualities can be further removed from the reads in a base quality recalibration step. Then, variants are called (either for each individual separately or jointly for a group of them) and genotype likelihoods inferred. At the final step, variant qualities are recalibrated, and variants filtered and annotated.

In such a pipeline, read mapping is a crucial step affecting overall results. The ultimate goal of a read mapper [202, 201] is to align NGS reads or their parts to their "true origin" in the reference genome, inferring their positions and occurred variants. To achieve this goal, each read is aligned to the most similar region(s) in the reference determined by the alignment scores.

There are several factors that make mapping a challenging task in practice. Genome sequences are highly repetitive and often contain several regions equally similar to a read. Employing a sequencing technology producing long or paired-end reads can alleviate but not completely eliminate this difficulty. Reliability of mapping strongly depends on how much the sequenced genome differs from the reference. Close individuals (such as those of the same species) usually differ only weakly and highly variable regions tend to be rare. For distant individuals, evolutionary events (such as genomic rearrangements or gene duplications) may cause more substantial differences between the two genomes making it impossible to deduce the true origin of some reads using similarity search. Sequencing errors constitute another major obstacle. Their rate strongly depends on the employed technology – from Illumina HiSeq producing reads of 100 bp with about 1% of errors to

PacBio SMRT and Oxford Nanopore producing reads of up to several tens of thousand bp with about 15% of errors – and on the experimental conditions. Furthermore, some reads may come from regions absent in the reference sequence or may result from contamination by DNA from other genomes. Such reads should be detected and annotated by the mapper, i.e. marked as unaligned or as aligned with low quality. Finally, the sheer volume of input data makes of mapping a computationally demanding task constraining the choice of the underlying algorithm. For instance, whole genome sequencing projects of the human genome of 3 Gbp often use more than 30× coverage.

As a consequence of the above, practical read mappers must represent a fine-tuned trade-off between memory requirements, speed and sensitivity, and incorporate specific characteristics of target sequencing technologies. The software must be highly optimized, parallelized, and well-debugged. Today, most common practical mappers include BWA-MEM [183], Bowtie 2 [184], GEM [194], LAST [176], and NovoAlign. These mappers align reads coming in a stream, in the online fashion, against a fixed reference sequence. Since the reference is unchanging, we call such an approach *static mapping*.

Despite a great deal of work on mapping techniques and a large number of existing mappers (see Chapter 3), the mapping process remains error prone and often produces wrong placement of reads, partial mappings, or incorrectly considers reads as non-mappable. Furthermore, another inherent drawback of static mapping is the bias introduced by alleles which differ between the reference and sequenced genomes as alleles present in the reference tend to be favored in the alignment of reads. This may seriously affect the results of the analysis [539].

We propose a *dynamic read mapping* approach that significantly improves read alignment accuracy. The general idea of dynamic mapping is to continuously update the reference sequence on the basis of previously computed read alignments. Even though this concept already appeared in the literature, we believe that our work provides the first comprehensive analysis of this approach

5.2 Our contributions

In Chapter 6, we provide a generic format Read Naming Format (RNF) for assigning read names with encoded information about original positions an associated software package RNFtools. This toolkit is particularly designed for comparison of different mapping approaches and we use it later for evaluation of dynamic mapping.

In Chapter 7, we present Ococo, the first consensus caller capable to process input reads in the online fashion from an unsorted SAM/BAM stream. The resulting algorithm, supporting substitution updates, uses a compact statistics scheme based on small bit counters. An online consensus caller is a required componed for dynamic mapping.

In Chapter 8, we introduce DyMaS, a dynamic mapping simulator that mimics different dynamic mapping scenarios and enables to evaluate their benefits. We show that in all alternatives, dynamic mapping results in a much better accuracy than static mapping.

Chapter 6

RNF: a general framework to evaluate NGS read mappers

We present a generic format Read Naming Format (RNF) for assigning read names with encoded information about original positions. Furthermore, we present an associated software package RNFtools containing two principal components. MIShmash applies one of popular read simulating tools (among DWGsim, ART, Mason, CuReSim, etc.) and transforms the generated reads into RNF format. LAVEnder evaluates then a given read mapper using simulated reads in RNF format. A special attention is payed to mapping qualities that serve for parametrization of ROC curves, and to evaluation of the effect of read sample contamination. RNFtools is available from http://github.com/karel-brinda/rnftools and RNF specification from http://github.com/karel-brinda/rnf-spec.

6.1 Introduction

The number of Next-Generation Sequencing (NGS) read mappers has been rapidly growing during the last years. Then, there is an increasing demand of methods for evaluation and comparison of mappers in order to select the most appropriate one for a specific task. The basic approach to compare mappers is based on simulating NGS reads, aligning them to the reference genome and assessing read mapping accuracy using a tool evaluating if each individual read has been aligned correctly.

As we have seen in Chapter 2.4, there exist many read simulators (e.g., ART [82], CuReSim [91], DNemulator [110], DWGsim[1], FASTQSim [84], FLOWSIM [99], GemSIM [88], Mason [78], PBSIM [76], pIRS [77], SInC [89], WGsim[2], XS [113]) as well as many evaluation tools (e.g., CuReSimEval, DWGsim_eval, MAF[3], RABEMA [540], Seqsuite[4], Teaser [541], WGsim_eval, etc.). However, each read simulator encodes information about the origin of reads in its own manner. This makes combining tools complicated and makes writing ad-hoc conversion scripts inevitable.

Here we propose a standard for naming simulated NGS reads, called Read Naming Format (RNF), that makes evaluation tools for read mappers independent of the tool

[1] http://github.com/nh13/dwgsim
[2] http://github.com/lh3/wgsim
[3] http://github.com/vsbuffalo/maf
[4] http://cbrc3.cbrc.jp/~martin/seq-suite/

used for read simulation. Furthermore, we introduce RNFtools, an easily configurable software, to obtain simulated reads in RNF format using a wide class of existing read simulators, and also to evaluate NGS mappers.

Simulation of reads. A typical read simulator introduces mutations into a given reference genome (provided usually as a FASTA file) and generates reads as genomic substrings with randomly added sequencing errors. Different statistical models can be employed to simulate sequencing errors and artefacts observed in experimental reads. The models usually take into account CG-content, distributions of coverage, of sequencing errors in reads, and of genomic mutations. Simulators can often learn their parameters from an experimental alignment file.

At the end, information about origin of every read is encoded in some way and the reads are saved into a FASTQ file.

Evaluation of mappers. When simulated reads are mapped back to the reference sequence and possibly processed by an independent post-processing tool (remapping around indels, etc.), an evaluation tool inputs the final alignments of all reads, extracts information about their origin and assesses if every single read has been aligned to a correct location (and possibly with correct edit operations). The whole procedure is finalized by creating a summarizing report.

Various evaluation strategies can be employed (see, e.g., introduction of [91]). Final statistics usually strongly depend on the definition of a correctly mapped read, mapper's approach to deal with multi-mapped reads and with mapping qualities.

Existing read naming approaches. Depending on the read simulator, information about the read's origin is either encoded in its name, or stored in a separate file, possibly augmented with information about the expected read alignment. While WGsim encodes the first nucleotide of each end of the read in the read name, DWGsim and CuReSim encode the leftmost nucleotide of each end. Unfortunately, these read naming schemes were specifically designed for particular sequencing technologies and single evaluation strategies, therefore they are not suitable as generic formats. ART produces SAM and ALN alignment files, MASON creates SAM files, and PIRS makes text files in its own format.

6.2 Methods

We have created RNF, a standard for naming simulated reads. It is designed to be robust, easy to adopt by existing tools, extendable, and to provide human-readable read names. It respects a wide range of existing sequencing technologies as well as their possible future evolution (e.g., technologies producing several "subreads"). We then developed a utility for generating RNF-compliant reads using existing simulators, and an associated mapper evaluation tool.

6.2.1 Read Naming Format (RNF)

Read tuples. *Read tuple* is a tuple of sequences (possibly overlapping) obtained from a sequencing machine from a single fragment of DNA. Elements of these tuples are called *reads*. For example, every "paired-end read" is a *read tuple* and both of its "ends" are individual *reads* in our notation.

```
Coor        12345678901-23456789012345678901234567890

Source 1 - reference genome
chr 1       ATGTTAGATAA-GATAGCTGTGCTAGTAGGCAGTCAGCCC
chr 2       ttcttctggaa-gaccttctcctcctgcaaataaa

Source 2 - generator of random sequences

READS:
r001        ATG-TAGATA ->
r002/1         TTAGATAACGA ->
r002/2                                  <- TCAG-CGGG
r003/1                                tgcaaataa ->
r003/2              gaa-gacc-t ->
r004                       ATAGCT............TCAG ->
r005                                  GTAGG ->
             <- agacctt
                        <- TCGACACG
r006       ATATCACATCATTAGACACTA
```

(a) Simulated reads

r. tuple	LRN	SRN
r001	sim__1__(1,1,F,01,10)__[single-end]	#1
r002	sim__2__(1,1,F,04,13),(1,1,R,31,39)__[paired-end]	#2
r003	sim__3__(1,2,F,09,17),(1,2,F,25,33)__[mate-pair]	#3
r004	sim__4__(1,1,F,15,36)__[RNA-seq-spliced],C:[6=12N4=]	#4
r005	sim__5__(1,1,R,15,22),(1,1,F,25,29),(1,2,R,05,11)__[chimeric]	#5
r006	rnd__6__(2,0,N,00,00)__[rand-contamin]	#6

(b) Long and short read names

Figure 6.1: **Examples of RNF reads.** Examples of simulated reads (in our definition *read tuples*) and their corresponding RNF names, which can be used as read names in the final FASTQ file: a single-end read (r001); a paired-end read (r002); a mate-pair read (r003); a spliced RNA-seq read (r004); a chimeric read (r005); and a random contaminating read with unspecified coordinates (r006).

To every *read tuple*, two strings are assigned: a short read name (SRN) and a long read name (LRN). SRN contains a hexadecimal unique ID of the *tuple* prefixed by '#'. LRN consists of four parts delimited by double-underscore: i) a prefix (possibly containing expressive information for a user or a particular string for sorting or randomization of order of *read tuples*), ii) a unique ID, iii) information about origins of all segments (see below) that constitute *reads* of the *tuple*, iv) a suffix containing arbitrary comments or extensions (for holding additional information). Preferred final read names are LRNs. If an LRN exceeds 255 (maximum allowed read length in SAM), SRNs are used instead and a SRN–LRN correspondence file must be created.

Segments. *Segments* are substrings of a *read* which are spatially distinct in the reference

and correspond to individual lines in a SAM file[5] Thus, each *read* has an associated chain of *segments* and we associate a *read tuple* with *segments* of all its *reads*.

Within our definition, a "single-end read" (Fig. 6.1, r001) consists of a single *read* with a single *segment* unless it comes from a region with genomic rearrangement. A "paired-end read" or a "mate-pair read" (Fig. 6.1, r002 and r003) consists of two *reads*, each with one *segment* (under the same condition). A "strobe read" consists of several *reads*. Chimeric *reads* (i.e., reads corresponding to a genomic fusion, a long deletion, or a translocation; Fig. 6.1, r005) have at least two *segments*.

For each *segment*, the following information is encoded: leftmost and rightmost 1-based coordinates in its reference, ID of its reference genome, ID of the chromosome and the direction ('F' or 'R'). The format is:
`(genome_id,chromosome_id,direction,L_coor,R_coor)`.

Segments in LRN are recommended to be sorted with the following keys: `source`, `chromosome`, `L_coor`, `R_coor`, `direction`. When some information is not available (e.g., the rightmost coordinate), zero is used ('N' in case of direction; Fig. 6.1, r006).

Extensions. The basic standard can be extended for specific purposes by extensions. They are part of the suffix and encode supplementary information (e.g., information about CIGAR strings, sequencing errors, or mutations).

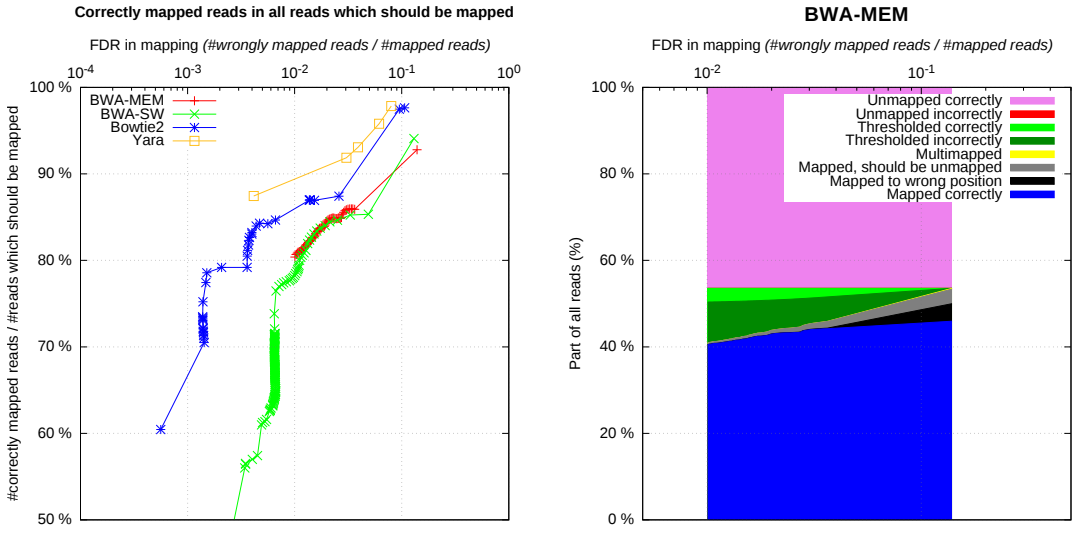
6.2.2 RNFtools

We also developed RNFtools, a software package associated with RNF. It has two principal components: MIShmash for read simulation and LAVEnder for evaluation of NGS read mappers. RNFtools has been created using Snakemake [542], a Python-based Make-like build system. All employed external programs are installed automatically when needed. The package also contains a lightweight console tool `rnftools` which can, in addition, be used for conversion of existing data and transformation of RNF coordinates using a LiftOver chain file.

MIShmash. MIShmash is a pipeline for simulating reads using existing simulators and combining obtained sets of reads together (e.g., in order to simulate contamination or metagenomic samples). Its output files respect RNF format, therefore, any RNF-compatible evaluation tool can be used for evaluation.

LAVEnder. LAVEnder is a program for evaluating mappers. For a given set of BAM files, it creates an interactive HTML report with several graphs. In practice, mapping qualities assigned by different mappers to a given read are not equal (although mappers tend to unify this). Moreover, even for a single mapper, mapping qualities are very data-specific. Therefore, results of mappers after the same thresholding on mapping quality are not comparable. To cope with this, we designed LAVEnder to use mapping qualities as parameterization of curves in 'sensitivity-precision' graphs (like it has been done in [183]).

[5] Since spliced RNA-seq *reads* (Fig. 6.1, r004) are usually reported in single lines in SAM, we recommend to keep them in single RNF segments without splitting even though they might be considered spatially distinct.



(a) ROC curves for all mappers.

(b) Categories of reads for a single mapper.

Figure 6.2: **Contamination experiment.** All LAVEnder graphs have false discovery rate on x-axis and use mapping quality as the varying parameter for plotted curves.

6.3 Results

To test the entire framework and verify its functionality, we simulated 200,000 single-end reads from human and mouse genomes (100,000 from HG38, 100,000 from MM10) by DWGsim using MIShmash and mapped them using Bowtie 2 [184], BWA-MEM [183], BWA-SW [188], and YARA [211] to HG38. Then we evaluated the obtained alignments using LAVEnder (Figure 6.2). This experiment revealed that YARA copes with contamination better than Bowtie 2 [184], BWA-MEM [183], and BWA-SW [188]. In particular, the ability to distinguish contamination was demonstrated to be very low in case of the BWA family.

6.4 Conclusion

We designed the RNF format and proposed it as a general standard for naming simulated NGS reads. We developed RNFtools consisting of MIShmash, a pipeline for read simulation, and LAVEnder, an evaluation tool for mappers, both following the RNF convention (thus inter-compatible). Currently, MIShmash has a built-in interface with the following existing read simulators: ART, CuReSim, DWGsim, Mason, and WGsim.

We expect that authors of existing read simulators will adopt RNF naming convention as it is technically simple and would allow them to extend the usability of their software. We also expect authors of evaluation tools to use RNF to make their tools independent of the used read simulator.

Chapter 7

Ococo: the first online consensus caller

We present Ococo, the first online consensus caller. Ococo reads alignments in an unsorted SAM/BAM stream and, employing compact statistics using small bit counters, decides about single-nucleotide updates of the reference sequence. The updates are reported in the online fashion using unsorted VCF. Ococo is available from http://github.com/karel-brinda/ococo, BioConda and Homebrew Science.

7.1 Introduction

A dynamic mapper has to collect statistics about previously mapped reads. When this statistics accumulates a sufficient evidence of a difference at a certain locus between sequenced and reference genomes, an update of the reference is reported that triggers a corresponding update of the index. This process is the same as the usual consensus calling except that it has to be performed in the online (streamed) fashion. In Chapter 9 we provide further remarks on the biological relevance of the consensus and on the effect of ploidy.

Here, we consider consensus calling as SNP calling using reads coming from a single individual. SNP calling is a widely studied problem (see, e.g., [543, 179, 544, 545, 290, 537, 546, 547, 548, 549, 550]) but to the best of our knowledge, all published solutions are offline, i.e, all reads have to be aligned and sorted prior to the calling step. Variants are inferred from differences between aligned reads and the reference and are usually stored in VCF format [551] together with confidence scores. They are then filtered in order to keep only those which are reliable enough, in particular those which do not correspond to sequencing errors. After that, they are incorporated into the original reference to obtain the consensus.

7.2 Methods

7.2.1 Consensus calling algorithm

Regular offline variant callers usually operate by sliding a small window through the genome and keeping in memory information (read alignments, mapping qualities, base

Incoming base	**A-counter** bin	dec	**C-counter** bin	dec	**G-counter** bin	dec	**T-counter** bin	dec	**Sum**	
	010	2	110	6	001	1	010	2	11	Initial state
C	010	2	111	7	001	1	010	2	12	C-counter incr.
T	010	2	111	7	001	1	011	3	13	T-counter incr.
C	001	1	011	3	000	0	001	1	5	All counters bit-shifted to right
	001	1	100	4	000	0	001	1	6	C-counter incr.

Figure 7.1: **Internal statistics of an online consensus caller.** This example shows how 3-bit counters keep reduced truncated pileup information for a single position. The counters with initial values $(2, 6, 1, 2)$ are incremented after nucleotides C, T, C have been received for this position. After receiving C and T, the corresponding counters have been simply incremented. Then, after receiving C, all counters are bit-shifted before the C-counter is incremented.

qualities) collected within individual windows. After variants are detected, their significance scores are computed using the underlying statistical model.

In online consensus calling, reads come from a stream and can map to random positions, therefore our "window" must be the entire genome. As a consequence, we can afford storing only a limited information about processed read alignments. Ococo only deals with single-nucleotide replacement updates, other types of updates will be considered in Discussion. Ococo supports two distinct working modes of an online consensus caller. In the *real-time mode*, suitable for dynamic mapping, updates are computed and incorporated into the reference after processing every read. In the *batch mode*, reads are processed in batches and updates are made after processing all reads of a batch, which can be an appropriate solution, e.g, for the hybrid approach (see Discussion).

7.2.2 Compact representation of variant statistics

As opposed to traditional offline consensus callers, an online caller can keep only a very reduced information about alignments. As a result, a space associated with a single position must be limited to a few bytes if we want to keep the statistics in RAM.

For example, in the case of human genome and 6 GB of dedicated RAM (note that the caller is likely to run jointly with other programs such as a read mapper), only up to two bytes per position can be used by the consensus caller. If indels are not considered, the full statistics should contain four integer counters per position, one per nucleotide. Our solution is to keep only most significant bits of each counter. If a counter is saturated but should be incremented, we first bit-shift all the four counters to the right losing the rightmost (least significant) bit, and then increment the counter. An example is given in Fig. 7.1. This mechanism makes it possible to compute nucleotide frequencies in a limited space and, as a side effect, to filter out sequencing errors that are expected to be randomly distributed. At the beginning of the calling procedure, counters are initialized according to the reference sequence (if it is provided), then counters are updated according to the read nucleotides aligned at the corresponding position.

7.2.3 Update strategies

When a counter is incremented, we have to decide whether this triggers an update. We propose two different strategies for updates, a *majority strategy* and a *stochastic strategy*. For the sake of simplicity and speed, both strategies consider genomic positions independently and take the decision on the basis of the counters associated with the position. This approach can be potentially improved, at the cost of an additional time, by considering a small window and using Bayesian inference (see, e.g., [543, 549]) to decide on updates.

Let $C_\alpha^{(j)}$ be the value of the α-counter at position j for a nucleotide $\alpha \in \{A, C, G, T\}$ and let $C^{(j)} = \sum_{\alpha \in \{A,C,G,T\}} C_\alpha^{(j)}$.

Majority strategy. The reference sequence at position j is updated to α once the condition $C_\alpha^{(j)} \geq \frac{1}{2} C^{(j)}$ holds, i.e., α reaches the majority at this position. This strategy results in a moderate number of updates which tend to vanish when the updating process reaches saturation. On the other hand, this strategy may not be flexible enough to avoid the alignment bias.

Stochastic strategy. When a counter is incremented, a new base at the corresponding position is drawn randomly from current counter values: the base is set to α with probability $\frac{C_\alpha^{(j)}}{C^{(j)}}$. Note that each sequencing error present in a read causes an increment of the corresponding counter and can, with a small probability, flip the corresponding nucleotide in the reference. Therefore, small fluctuations of the reference sequence can occur even in the hypothetical case of the reference being equal to the sequenced genome. However, a major advantage of this strategy is the reduction of the alignment bias. For diploid or polyploid genomes, the reference will be constantly oscillating between all variants. On the downside, this strategy may generate a large number of updates resulting in a significant overhead.

Note that neither of these two strategies depends on a particular counting mechanism, therefore they can be potentially used with a more complex and expressive statistics.

7.3 Implementation and availability

Ococo[1] supports single-nucleotide updates, and implements both stochastic and majority update strategies as well as both real-time and batch modes. The software is implemented in C++ and can be integrated into a genomic pipeline or linked directly to a mapper as a library. It can be installed also using the Homebrew Science[2] and BioConda[3] package systems.

Ococo currently supports three counters configurations: 16, 32, and 64 bits per position corresponding to 3, 7, and 15 bits per a single nucleotide counter, respectively. Updates in the reference sequence can be either recorded using "unsorted VCF" or can be directly incorporated in a FASTA file (possibly memory-mapped).

[1] http://github.com/karel-brinda/ococo
[2] http://brew.sh/homebrew-science/
[3] http://bioconda.github.io

Chapter 8

DyMaS: a dynamic mapping simulator

To evaluate the benefits of dynamic mapping, we developed DyMaS (Dynamic Mapping Simulator), a software pipeline that mimics different dynamic mapping scenarios. We applied the pipeline to compare dynamic mapping with the conventional static mapping and, on the other hand, with the so-called iterative referencing – a computationally expensive procedure computing an optimal modification of the reference that maximizes the overall quality of all alignments. We conclude that in all cases, dynamic mapping results in a much better accuracy than static mapping, approaching the accuracy of iterative referencing. The DyMaS pipeline and the all other scripts and files from this chapter are available from http://github.com/karel-brinda/dymas.

8.1 Introduction

In the course of mapping, read alignments computed so far provide a useful information about allelic differences between sequenced and reference sequences, which can be a helpful guide for adjusting future mappings, or can even call for remapping previously mapped reads. In this chapter, we study the *dynamic mapping* approach when the reference sequence is continuously updated based on already computed alignments, and show that this can significantly improve the quality of mapping. Fig. 8.1 provides an illustration to this idea. The general goal of our work is to analyse the pros and cons of dynamic mapping and to provide accurate quantitative estimates supporting our analysis.

Design and implementation of dynamic mapping constitute a difficult task. One major obstacle is that the underlying index data structure for the reference sequence must support dynamic updates. There are two main types of indexing structures used by modern mappers. The first one relies on *full-text indexes*, in particular on BWT-index [552], and is used, among others, by the BWA family [181, 188, 183], GEM [194], or Bowtie 2 [184]. Another group is based on *hash tables* and includes such mappers as SHRiMP2 [190], PerM [274], NovoAlign, BFast [191], StamPy [260], and others. Generally speaking, hash table-based structures require a much larger memory space and, therefore, projects dealing with large data (such as whole eukaryotic genomes) usually use mappers of the first group. On the other hand, BWT-index hardly lends itself to dynamic updates. One attempt to implement dynamic updates of BWT-index was made in [536, 524, 508], another approach

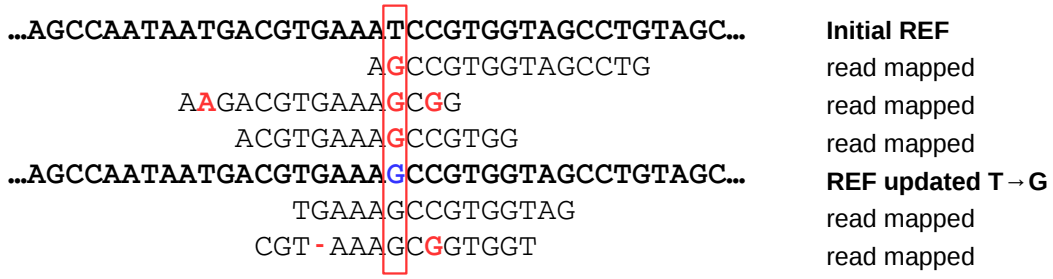


Figure 8.1: **Demonstration of idea of dynamic mapping.** In dynamic mapping, reference sequence is updated according to already mapped reads. In this example, aligned reads suggest that the base at the highlighted position is G rather than T, and the reference is corrected as soon as G is supported by sufficiently many reads. To guide the updates, a dynamic mapper must be equipped with an online consensus caller distinguishing SNPs and sequencing errors. Disagreements between aligned reads and the reference (sequencing errors, SNPs and indels) are highlighted in red and updates of the reference in blue.

was studied in [535]. In both cases, provided implementations are only experimental and cannot be considered as ready-to-use tools for large-scale dynamic mapping. A dynamic k-mer-based index has very recently been proposed in [553].

Another important component of dynamic mapping is online consensus calling which guides the updates of the reference. An online consensus caller must support quick updates of the reference after new read alignments have been computed. This requires keeping, in a compact form, a genome-wide statistics of possible variants, such as a truncated pileup information. Finally, a dynamic mapper may have to support remapping of already mapped reads.

Dynamic mapping has been little studied so far and only two partial experimental solutions with naive mapping algorithms are available [380, 381]. Program DynMap [380] implements a data structure specifically designed for mapping reads to a dynamically updated sequence. Unfortunately, space complexity of DynMap is linear in the number of reads which makes it hardly applicable to contemporary sequencing technologies. Manuscript [381] (apparently unpublished) is the first attempt to systematically analyse the benefit and the overhead of dynamic mapping using a simple dedicated tool (`https://github.com/jpritt/fm-update`). With an index based on dynamic suffix array [508], the accuracy of dynamic mapping has been compared to iterative referencing, depending on the mutation rate, read length, number of reads and sequencing error rate. The author of [381] also considered the problem of statistics reduction and suggested to use coverage vectors in order to take only first k reads for each base of the reference.

In this chapter, we present a software pipeline for simulating various alternatives of dynamic mapping. We then provide a comparative analysis of different dynamic mapping scenarios on several datasets simulated from bacterial genomes, using the previously developed RNFtools framework [2]. Obtained results show that dynamic mapping can provide a significant improvement of the accuracy of read alignment compared to traditional mapping approaches. The pipeline software and the results of our experiments are available from `http://github.com/karel-brinda/dymas`.

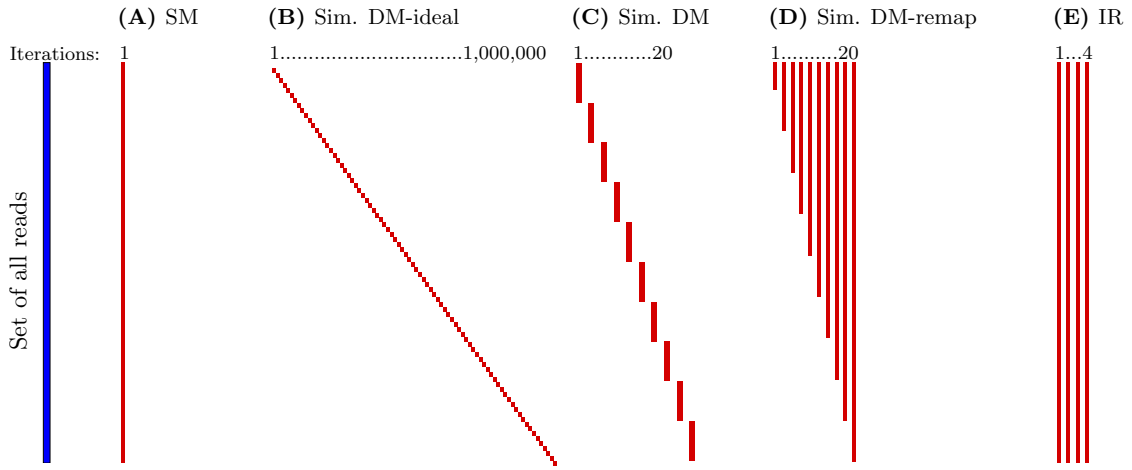


Figure 8.2: **Simulation of dynamic mapping.** This schematic view shows how dynamic mapping and iterative referencing are simulated in Dynamic Mapping Simulator using a static mapper alone. Subsets of reads to be mapped in individual iterations are shown by vertical lines. In static mapping (SM) (**A**), all reads are mapped in a single iteration. Figure **B** illustrates an ideal, but not feasible, simulation when consensus is called after each new read is mapped. In practice, reads are processed by batches (Figure **C**,**D**). Dynamic mapping (DM) with (**D**) and without (**B**,**C**) remapping is simulated using nesting or disjoint subsets of reads, respectively. Figure **E** illustrates iterative referencing (IR). In all scenarios **B** – **E**, each iteration is followed by consensus calling and index rebuilding.

8.2 Methods

8.2.1 Simulation algorithm

Suppose that we have a reference genome G and a set of reads $R = \{r_1, \dots, r_p\}$ that have to be mapped to G. A dynamic mapping can be simulated using a static mapper and an off-line consensus caller by iterating p times the following algorithm. Let G_i be the reference genome obtained after iteration i, where $i \in \{1, \dots, p\}$, and let $G_0 = G$. At iteration i, we build an index for the reference G_{i-1}, map read r_i to G_{i-1}, call consensus from alignments of $r_1, \dots, r_i$, and build a new reference G_i (see Fig. 8.2B). The difference with the truly dynamic mapping is that instead of modifying the reference index after mapping a new read, we rebuild it from scratch.

One technical point here is that if indel updates are supported by the procedure, the coordinates of downstream loci can be shifted, which would require a transformation of alignments of $r_1, \dots, r_i$ from the coordinate system of G_{i-1} to that of G_i. We will focus on this point in Discussion.

The above approach faithfully simulates dynamic mapping without remapping (i.e. without reconsidering previously computed read alignments), however it cannot be used in practice since the computationally demanding index rebuilding has to be performed too many times. To overcome this obstacle, we observe that at every iteration i, new updates can be reported only at positions covered by the newly mapped read r_i. We can then partition reads into t batches $Q_1, \dots, Q_t$, for some $t \ll p$, such that the alignments of any two reads from the same batch do not overlap. This can be achieved, e.g., by assigning

only dissimilar reads to the same batch. Now the algorithm works in t iterations and at iteration $i \in \{1, \ldots, t\}$, all the reads from batch Q_i are mapped, which makes the whole procedure fast enough.

In practice, however, computing such batches is a complex task in itself, and instead of fully avoiding overlaps, we minimize them. We choose t large enough and partition reads into t equal-size batches $Q_1, \ldots, Q_t$ (Fig. 8.2C). If the average coverage per iteration C_i satisfies

$$C_i = \frac{\sum_{r \in Q_i} \text{length(r)}}{\text{length(G)}} \ll 1$$

for every $i \in \{1, \ldots, t\}$, then overlaps are rare.

Until now, we considered dynamic mapping without remapping. To measure the potential effect of remapping, we slightly modify the previous algorithm. At iteration i, instead of Q_i, we map the whole set $Q_1 \cup \ldots \cup Q_i$ (see Fig. 8.2D). Note that such an algorithm simulates the ideal situation when a dynamic mapper remaps all reads which can be better aligned after an update of the reference.

To summarize, dynamic mapping is simulated as follows. Given a set of reads and a reference genome G, we define C_i, compute corresponding number of iterations t and randomly split reads into batches $Q_1, \ldots, Q_t$ of nearly equal size. Then, every iteration consists of three steps: re-building the index according to the current version of the reference, mapping reads, and calling consensus. Depending on whether we choose to perform remapping or not, mapping step i involves $Q_1 \cup \ldots \cup Q_i$ or Q_i respectively.

8.2.2 Iterative referencing

To analyze the contribution of dynamic mapping, we compare it to static mapping and, on the other hand, to the so-called *iterative referencing* [554]. Iterative referencing (sometimes called *iterative read mapping* or *iterative remapping*) consists in repeatedly mapping all reads followed by consensus calling, see Fig. 8.2E. Thus, each update of the reference is inferred from *all* read alignments spanning this locus, and the whole mapping procedure is iterated until the convergence of the reference is reached. Clearly, iterative indexing is a costly method, unfeasible for large datasets, as the index building and the mapping of the whole read set has to be performed several times. On the other hand, iterative indexing tends to produce an optimal variant of the reference that maximizes the overall quality of all alignments. Therefore, in our work we use iterative referencing as a reference for evaluating the quality of dynamic mapping. Various forms of iterative referencing have previously been used in [555, 556, 557, 558, 559, 560, 561, 562, 563, 564].

8.3 Implementation

We developed Dynamic Mapping Simulator[1], a pipeline for comparative evaluation of three mapping methods: static mapping, dynamic mapping (with or without remapping), and iterative referencing (Fig. 8.3).

For a given reference genome, reads with mutations and sequencing errors are simulated by RNFtools [2] using DWGsim[2]. Obtained reads are then distributed into FASTQ files for individual iterations (see Fig. 8.2). Each iteration starts with a mapping step (using

[1] http://github.com/karel-brinda/dymas
[2] https://github.com/nh13/DWGSIM

BWA-MEM [183] or Bowtie 2 [184]) producing a BAM file. At the consensus calling step, a VCF file with updates is created. Two ways of calling consensus have been implemented. One way is calling consensus using Ococo (see the previous section) directly from unsorted BAM files coming from the mapper. An alternative way is sorting the alignments and producing a pileup file using SAMtools [125], followed by consensus calling via an ad hoc Python script. Finally, reported updates are incorporated into the reference using BCFtools [565] and the index is rebuilt.

Note that consensus calling using Ococo is much faster since it avoids the time-demanding sorting step, however it supports only substition updates. In calling from pileup, deletions and insertions are additionally supported, but the simulation may be slightly less accurate as alignments are processed in the increasing order of starting positions which differs from the order of the original stream. Furthermore, before the evaluation step, read coordinates encoded in RNF names (see below) must be recoded into the current coordinate system in accordance with obtained liftOver chain files.

The entire pipeline is implemented using Snakemake [542] with SMBL (Snakemake Bioinformatics Library[3]). Each component (read mapping, pileup computation, consensus calling, etc.) has a standalone easily modifiable Python class. In several auxiliary scripts, GNU Parallel [566] has been used.

8.4 Evaluation

To compare the three mapping methods (static mapping, dynamic mapping, iterative referencing), we use the RNFtools framework [2]. Within this approach, names of simulated reads store information about their position in the reference. Comparing these coordinates with those reported by mapper allows us to evaluate the correctness of the mapping.

Given a set of BAM files corresponding to iterations of dynamic mapping and iterative referencing, RNFtools creates:

- Data files for all iterations containing the amount of reads of different categories as a function of the threshold on mapping quality.
- Graphs visualizing progressive updates of the ROC curve through individual iterations (see Fig. 8.4A).
- Graphs displaying fractions of different categories of reads processed in individual iterations (see Fig. 8.4B).
- Interactive HTML report with all previous data and figures.

All graphs have false discovery rate (FDR) on their x-axes. A lower FDR corresponds to a higher threshold on mapping quality, hence a larger number of reads that do not pass quality filtering.

Correctness of an alignment is estimated with respect to its leftmost coordinate. Depending on the tolerance provided to RNFtools, fully-aligned or partially aligned reads can be considered correctly aligned. This allows us to distinguish different consequences of dynamic mapping on the reported alignments.

[3] https://github.com/karel-brinda/smbl

Experiment	Reference genome	Contaminating genome
Exp. 1	*Borrelia garinii* (NC_017717.1) length 905,534 bp coverage 10× (91,856 reads)	(no contamination)
Exp. 2	*Mycobacterium tuberculosis* (NC_018143.2) length 4,411,709 bp coverage 10× (447,481 reads)	*Borrelia garinii* (NC_017717.1) length 905,534 bp coverage 5× (45,928 reads)
Exp. 3	*Neisseria meningitidis* (NC_017513.1) length 2,184,862 bp coverage 10× (221,616 reads)	*Borrelia garinii* (NC_017717.1) length 905,534 bp coverage 5× (45,928 reads)
Exp. 4	*Solibacter usitatus* (NC_008536.1) length 9,965,640 bp coverage 10× (1,010,810 reads)	*Borrelia garinii* (NC_017717.1) length 905,534 bp coverage 5× (45,928 reads)

Table 8.1: **Experiments conducted with Dynamic Mapping Simulator.**

8.5 Results

Experimental setup and representation of results. To evaluate the effect of dynamic mapping, we applied our Dynamic Mapping Simulator in a series of four experiments (Table 8.1). Each experiment focused on a specific reference genome. In all experiments except one, we considered, in addition, "contamination" reads coming from another genome. This allowed us to analyze the consequences of sample contamination by other genomes as well as presence of reads issued from regions absent in the reference sequence.

Experiment 1 evaluates different modes of dynamic mapping on a short genome without any contamination. Experiment 2 highlights the impact of contamination. Experiments 3 and 4 focus on two particular types of bacterial genomes, namely a highly repetitive genome (*Neisseria meningitidis*) and a long genome (*Solibacter usitatus*, 10.0 Mbp), respectively.

Within each experiment, we measured the mapping performance for different modes: static mapping, dynamic mapping and iterative referencing with or without remapping, with or without calling deletions, with or without calling both insertions and deletions (indels). The performance of each mapping strategy is represented by a ROC curve on a chart relating the fraction of incorrectly mapped reads (false discovery rate) to the fraction of correctly mapped reads among all reads that should be mapped (Fig. 8.5 and Fig. 8.4A). This 'sensitivity-precision' curve uniformly represents the mapping quality on the whole range of parameters and independently of the specific scoring system, score threshold and other involved specific parameters [2]. We also partition reads in eight categories based on whether they are mapped or unmapped correctly (Fig. 8.4B).

• *Reads that should be mapped* are subdivided into categories *Mapped correctly* (mapped

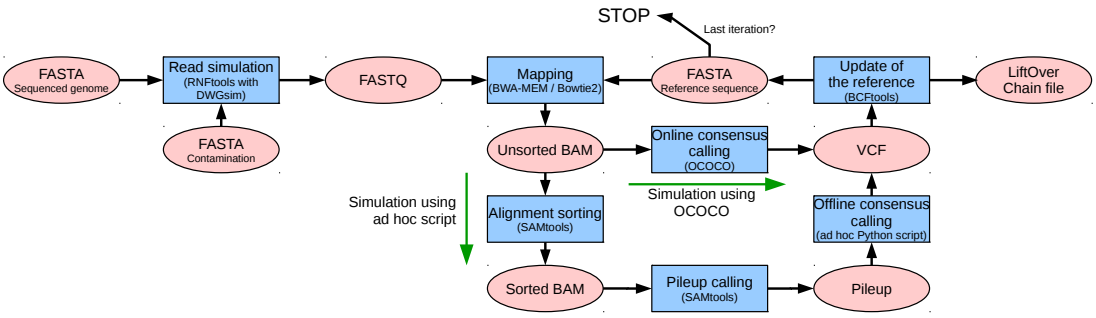


Figure 8.3: **Dynamic Mapping Simulator: overview of the pipeline.** Reads are selected for mapping as illustrated in Fig. 8.2. Reference sequence is repeatedly corrected according to the consensus called from already mapped reads. If insertion or deletion updates are allowed, LiftOver Chain file is used to update RNF coordinates in read names.

to correct position, passed quality filter), *Mapped to wrong position* (mapped to incorrect position, passed quality filter), *Multimapped* (multiple reported alignments passed quality filter including the correct alignment), *Thresholded incorrectly* (should be mapped but was discarded by quality filter), *Unmapped incorrectly* (incorrect assignment of flag 'unmapped').

• *Reads that should not be mapped* (i.e. coming from contamination) are split into *Unmapped correctly* (correct assignment of flag 'unmapped'), *Mapped, should be unmapped* (mapped and passed quality filter), *Thresholded correctly* (mapped but filtered out).

Technically, in the case of dynamic mapping and iterative referencing, a run of the pipeline produces a set of BAM files, each file storing results of an individual iteration of the mapping algorithm. The final result is considered the one of the last iteration.

Analysis of the results. We provide a comprehensive comparative view of different scenarios of dynamic mapping and iterative referencing, contrasted with the regular static mapping (Fig. 8.5). The first immediate observation is that for each experiment, all curves have a similar shape and can be ordered to have a strictly increasing performance. This supports the observation that their difference is only due to the quality of the reference sequence.

As expected, both dynamic mapping and iterative referencing are significantly better than static mapping. In the first approximation, iterative referencing and dynamic mapping with remapping provide comparable results, with iterative referencing being slightly superior. The latter is natural, as iterative referencing uses all read alignments to make decisions about updates, while dynamic mapping uses only a part of them. On the other hand, results of dynamic mapping without remapping are significantly worse than those of dynamic mapping with remapping. This means that many reads processed in the beginning of the mapping process are inaccurately aligned, which significantly affects the overall results. Thus, correction of those alignments appears an essential step for improving the mapping procedure.

We observe from that indel updates bring a consistent improvement (Fig. 8.5). Moreover, even supporting deletions alone leads to a more accurate mapping, and accounting for both deletions and insertions of events improves the accuracy even further. Note however that supporting indels by a truly dynamic mapper runs into a hard algorithmic problem

of dynamic updates of the underlying data structure and online consensus calling (see Introduction and Discussion).

We found that contamination of read data by sequences from another genome did not have any qualitative consequences on the results. When contamination was introduced, qualities of all runs had been decreased comparably.

Technical observations. We also experimented with using Bowtie 2 as a mapping subroutine. A particularity of Bowtie 2 is that it has both global and local alignment modes, whereas BWA-MEM performs only local alignment. This allowed us to compare the impact of these modes on the results of the pipeline. Our conclusion is that dynamic mapping should be used in combination with a local alignment algorithm. In the case of global alignment, starting and ending bases of a read tend to be aligned incorrectly in highly mutated regions, which causes statistics corruption and, as a consequence, wrong updates resulting in a damaged reference.

Usage of per-base alignment qualities which has been shown to decrease the amount of false positives in updates [567] has not provided any improvement in our experiments. However, this might be because we do not simulate structural variants for which such a recalibration can improve the accuracy of calling.

We also compared the effect of different amount of statistics stored by a consensus caller (see section above on consensus caller). We found that the default Ococo option of 16 bits per position did not produce a loss of accuracy compared to the option of 32 bits per position (data not shown).

Detailed reports. Full data, HTML reports, and detailed information about performed experiments can be found in directories *Experiments* and *Reports* at `http://github.com/karel-brinda/dymas`.

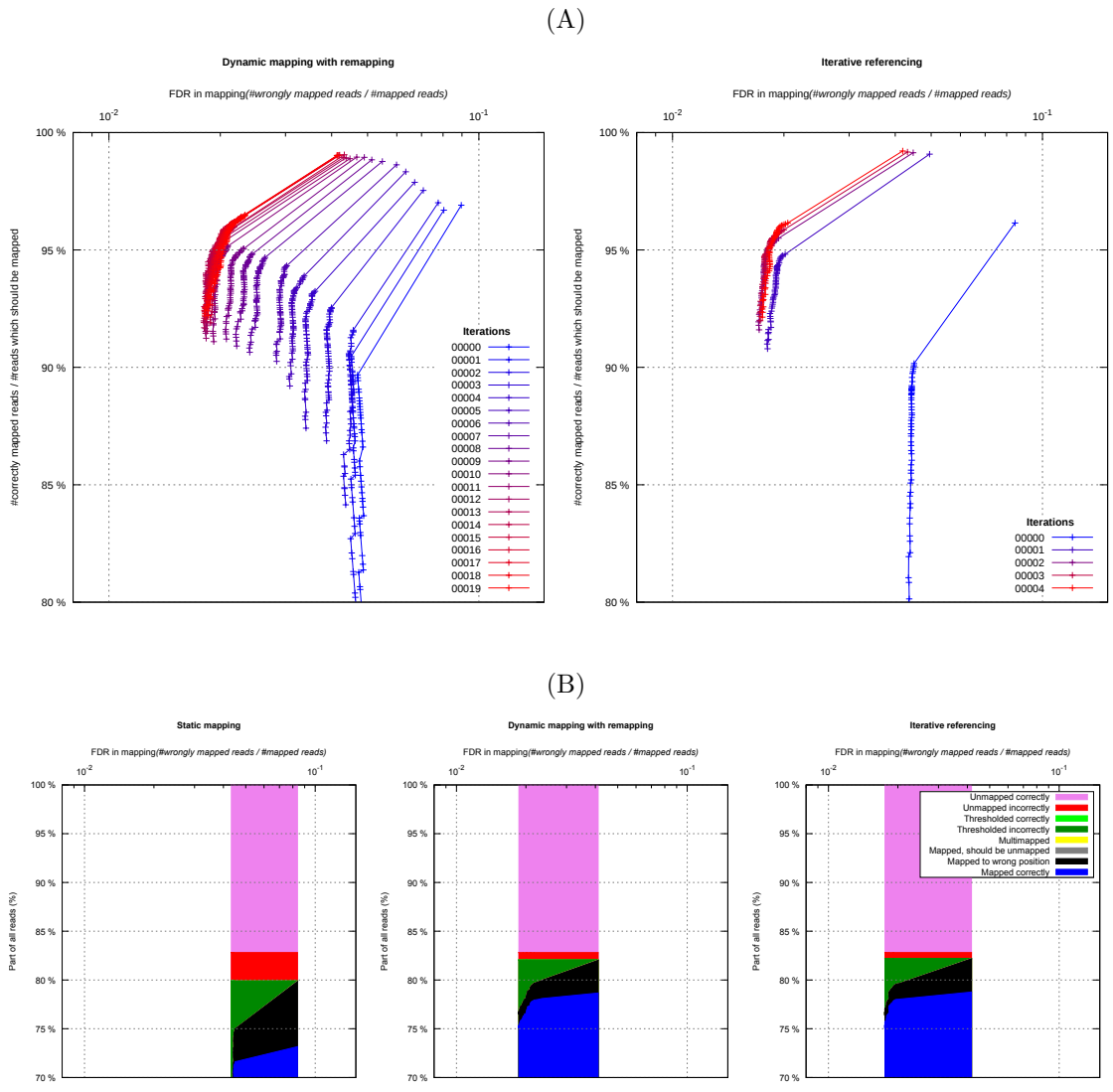


Figure 8.4: **Detailed report of a selected run of Experiment 3.** Pipelines have been run using BWA-MEM mapping algorithm, with the option of indel calling. Reads were simulated with sequencing error rate 0.02 from mutated *Neisseria meningitidis* genome (mutation rate 0.07) with coverage 10× and contaminated by reads from *Borrelia garinii* with coverage 5×. **(A)** Comparison of ROC (Receiver Operating Characteristic) curves of all iterations of simulated dynamic mapping with remapping (left) and of iterative referencing (right). **(B)** Categories of reads after static mapping, dynamic mapping with remapping, and iterative referencing. Fractions of different categories of reads depending on the level of precision (false discovery rate) for static mapping (left), dynamic mapping with remapping (center) and iterative referencing (right).

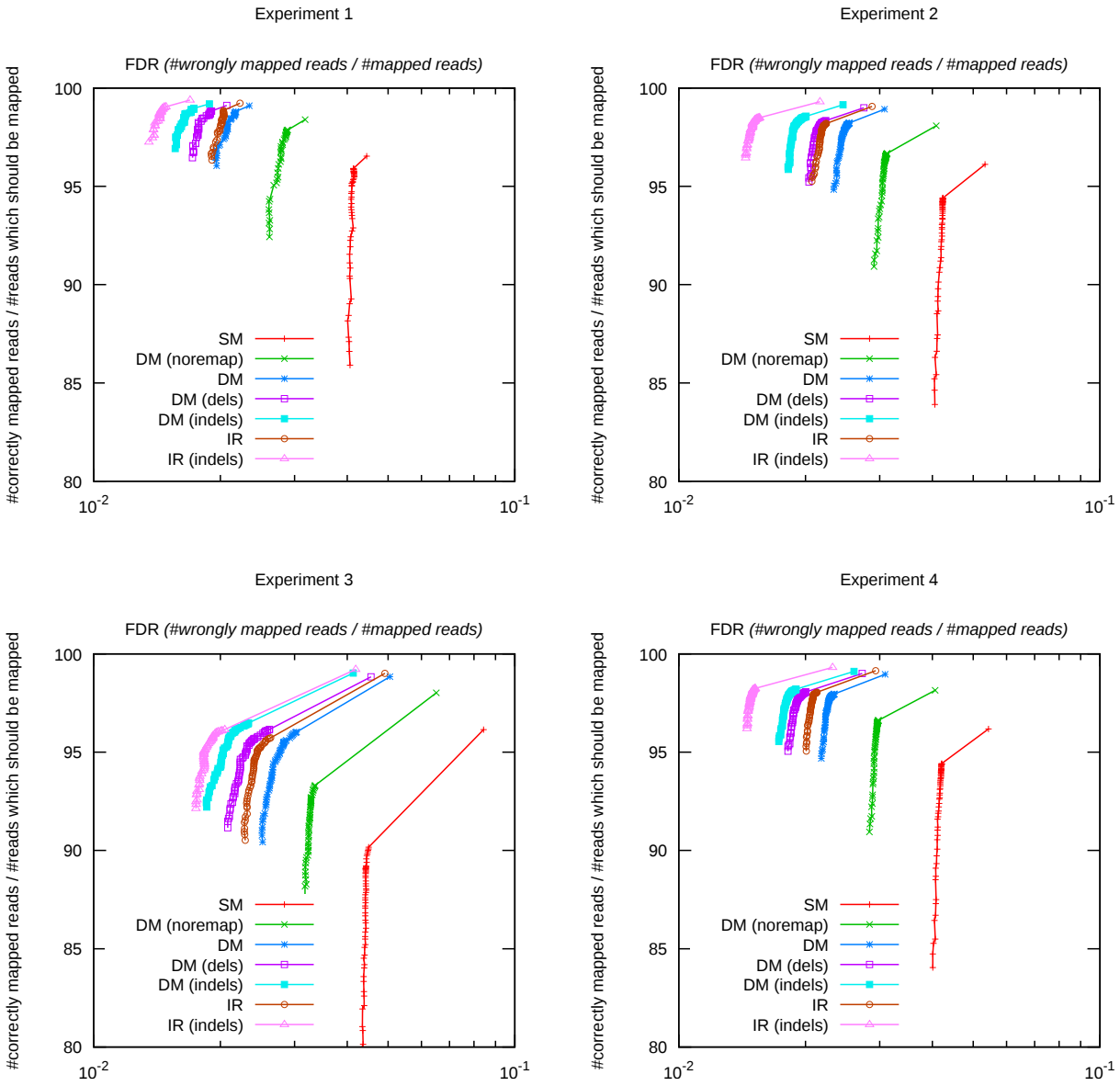


Figure 8.5: **Comparison of mapping strategies.** Comparison of static mapping (SM), dynamic mapping without calling indels (DM), with calling deletions (DM (dels)), with calling indels (DM (indels)), without remapping (DM (noremap)), and iterative referencing without calling indels (IR), with calling deletions (IR (dels)), with calling indels (IR (indels)) for every experiment.

Chapter 9

Discussion

Consensus sequence and ploidy Even though consensus is primarily used as a technical mean to obtain better alignments, the consensus sequence has a biological significance. If the sequenced genome is haploid, the final consensus tends to correspond to this genome. If it is diploid, the consensus will only contain one variant nucleotide of each SNP, or can even contain none of them if both variants are supported by approximately equal numbers of reads. However, an online consensus calling algorithm can be extended to support diploid genomes as well, provided that ambiguous bases are supported by the mapping algorithm and the underlying index (similarly, e.g., to [262] or [246]). The consensus would then be updated to an ambiguous nucleotide if two distinct variants have been observed at this position.

Non-substitution updates Updates other than substitutions (insertions, deletions, segment reversals, etc.) are difficult to support in dynamic mapping for two main reasons. Firstly, they entail changes in genomic coordinates, and therefore either all reported alignments have to be translated into the coordinate system of the final consensus sequence (which may require local realignment of reads), or differences between the original reference sequence and the current consensus have to be continuously recorded in a dedicated data structure allowing the retrieval of original coordinates. Secondly, appropriate counters and calling algorithms have to be designed for each type of updates.

If we are limited to deletions only, these problems can be solved by using a padded reference sequence (see, e.g., [568]) and introducing specific counters for deletions. Then, deletions are treated in consensus calling in the same way as mismatches, after extending the nucleotide alphabet by an additional letter '∗'. If both insertions and deletions have to be supported, the algorithm becomes much more difficult to implement. Using a padded reference allows one to only support insertions at positions from a pre-specified list (by introducing '∗' at these positions), or at positions where a deletion has previously been made. A full support of insertions of arbitrary length and occurring at arbitrary positions remains an open problem that is still awaiting a practical algorithmic solution.

Databases of SNPs SNP databases such as dbSNP [374] can be incorporated into dynamic mapping in two ways. We can either use an SNP database to set initial values of online consensus caller counters, or restrict the updates only to SNPs recorded in the database. Restricting possible updates would reduce the computational complexity of the whole algorithm and would improve the accuracy of processing those updates by keeping

a richer variant statistics and improving indel calling. Note that this approach can be viewed as a particular case of *path projection*, see section Reference graphs.

Pileup information As a side product of dynamic mapping, a simplified pileup information can be obtained (for more information about the pileup format, see SAMtools documentation [125]). Therefore, NGS methods supporting pileup as input format (such as those of [569, 570]) can be combined with dynamic mapping in order to avoid expensive steps of alignment sorting and pileup calling. Note that while dynamic mapping causes a constant-factor slow-down of the mapping step – from time t to time kt, where k is a deceleration constant specific to each dynamic mapping algorithm – the sorting step has $n \log(n)$ time complexity, where n is the number of reads, which often makes this step the most time demanding of the pipeline.

In the Ococo online consensus calling algorithm (see Methods), several pieces of information are lost compared to the "standard pileup" (as produced by SAMtools): information about base qualities and alignment qualities, order of bases (only counts are kept), distinction of strands (forward or reverse), deletions and insertions. Moreover, the pileup is truncated when the counter capacity is lower than the maximal coverage in the experiment.

At the price of an additional memory consumption, the counting mechanism can be modified to recover the distinction of strands (via doubling all counters) and the count of deletions (via adding a specific counter for deletions, see section Non-substitution updates). Truncating can be avoided in Ococo already now by setting the size of counters large enough.

Hybrid approach Many computational pipelines are deployed on clusters of many nodes, which enables to combine online consensus calling with static mapping in a similar way as dynamic mapping without remapping has been simulated. More specifically, one particular node would be assigned to collect alignments from mapping nodes and to call consensus online in the batch mode. Another node, in charge of maintaining the index, could iteratively retrieve the current consensus, rebuild the index and distribute it to mapping nodes. Such a hybrid approach (combining dynamic mapping and iterative referencing) remains to be verified in practice.

Sequencing in streaming mode Most of existing sequencers are technically possible to be run in a streaming mode, and several studies [571, 572, 573, 574, 575, 576] already considered the streaming mode in various tasks. Moreover, practical software solutions for streaming sequencing have already been proposed [577].

In the context of dynamic mapping, streaming sequencing can be particularly beneficial, as produced reads can be directly piped to a dynamic mapper. Then the sequencing can be kept on until the internal statistics gets saturated and the consensus sequence becomes stable enough. After these combined sequencing and mapping steps, only a certain amount of last reported alignments should be used for the subsequent analysis. The reason is that, as follows from the comparison of dynamic mapping with and without remapping (Fig. 8.5), reads mapped to a mature consensus are aligned more accurately.

Detection of somatic cancer mutations Somatic cancer mutations are often very underrepresented in the reads as tumor samples are contaminated by normal cells and, on the other hand, somatic mutations are often specific to particular subclones of the tumor.

Therefore, frequencies of somatic cancer mutations can be even lower than the sequencing error rate, which makes their accurate detection very hard (see, e.g., [578]). To deal with this issue, normal (control) samples need to be sequenced in addition to tumor samples to allow for a comparative analysis – see, e.g., papers [579, 580, 581, 582, 583, 569, 578, 584] on detecting single-nucleotide variants or papers [585, 586, 569, 570] on copy-number abberations.

To deal with low mutation frequencies, methods have to rely on high-quality alignments. As we showed in our work, dynamic mapping can produce significantly more accurate alignments, which results in a better identification of somatic mutations proximal to SNPs. Note that cancer SNV databases such as COSMIC [587] can be incorporated into dynamic mapping, as it was with SNP databases (see section Databases of SNPs).

Low somatic mutation frequencies make it necessary to employ a high sequencing coverage which in turn results in a longer processing time because of the slow alignment sorting step. However, this step can be avoided by working directly with the simplified pileup (see section on Pileup information).

Long reads Sequencing technologies producing long reads, such as *Pacific Biosciences single-molecule real-time sequencing* or *Oxford Nanopore sequencing*, have recently emerged. Compared to traditional NGS platforms such as *Illumina MiSeq* or *HiSeq*, they produce considerably longer reads with higher rates of errors, especially indels. This situation calls for new algorithms specifically designed for long reads, and dynamic mapping can be helpful in several ways here.

Some methods of long read correction (see, e.g., [588]) proceed by aligning short reads to long reads, possibly in a way similar to iterative referencing (see, e.g., [562]). Within these alignment-based methods, dynamic mappers have a great potential to improve the mapping step. However, considering the type of errors in long reads, indel updates have to be supported by the mapper (see section Non-substitution updates).

Another perspective is applying dynamic mapping of long reads. Unfortunately, consensus calling from long read alignments appears to be a hard problem even in the offline setting, due to observed error profiles. For instance, genome assembly algorithms apply sophisticated techniques based on partial order graphs to obtain a good consensus (see, e.g., [589]). In order to adapt dynamic mapping to long reads, update strategies in online consensus calling have to be significantly improved, in particular a single update decision has to take into account counter information at multiple positions. This approach requires a better statistical modeling and deserves further research.

Guided genome assembly If a high quality reference genome is not available, the approach based on read mapping and variant calling cannot be applied directly. Instead, a computationally expensive genome assembly (see, e.g., [118]) becomes unavoidable. Nevertheless, when a reference genome for a related species is available, it can be exploited in order to speed up the assembly process or to improve the quality of produced assemblies. This approach is known as *reference-guided assembly* or *assisted assembly*. Either a single [590, 591, 592] or multiple [593] reference genomes are used as references for auxiliary read alignment to create contigs, or a reference helps to combine assembled contigs [594, 595, 596, 597]. In those approaches where mapping *of* or *to* contigs is employed, dynamic mapping can help in a similar way to what was discussed in section Long reads (contigs can be viewed as a special case of long reads).

Phylogenetic inference A class of methods of phylogenetic inference relies on read mapping. In those methods, reads are mapped to one or multiple references, SNP positions are extracted and a phylogenetic tree is reconstructed, typically using a maximum likelihood technique. It has been shown [598] that even in the case of relatively low degree of divergence between queried and reference sequences, the mapping bias can cause inaccurate estimations of the distance which results in incorrect tree topologies. The correction capacity of dynamic mapping could reduce this bias and produce better phylogenetic distances, which constitutes a promising application of dynamic mapping.

Reference graphs As the traditional concept of sequences as reference structures has become insufficient [365], *reference graphs* (sometimes also called *variation graphs*) appear to be a more appropriate model than *reference sequences*. Therefore, new techniques of mapping to reference graphs [376, 375, 368, 370, 369, 377, 297, 526], as well as implementations of reference graphs [599, 600, 601] are now a subject of intensive research.

Since state-of-the-art graph mappers (such as VG[1] with GCSA2[526] as indexing component) are still experimental and no dynamic mapper has been developed so far, a hypothetical *dynamic graph mapper* can now be studied only from a theoretical viewpoint. Two ways of combining dynamic mapping and mapping to reference graphs can be considered: *path projection* and *dynamic graphs*.

In the *path projection* approach, a fixed graph reference is projected onto a linear sequence. According to observed variants, the reference sequence is updated in such a way that it still corresponds to some path in the graph reference. Therefore, this can be considered as a form of dynamic mapping with a restricted set of allowed updates specified by the graph reference. Decisions about updates can be done stochastically, similar to stochastic consensus calling described in the Methods section. In such an approach, the probability of a path to be selected should reflect the fraction of reads already mapped to this path and their mapping qualities.

Implementing path projection requires a dynamic mapper with a particular online consensus caller working with a graph reference. Instead of collecting statistics for individual bases, it should collect statistics for paths in the graph. The main advantage of this approach is that it can be built on top of a mapping algorithm working with linear references (such as BWA or Bowtie2). Note that very dissimilar parallel paths in the graph could be decomposed into several smaller graphs to prevent extensive updates of the linearized sequence.

In the *dynamic graphs* approach, the reference graph itself is updated according to reads mapped so far. The graph is either extended by incorporating all observed variants as new paths, or modified within the same topology, or both. In the first case, online consensus calling can be completely avoided if *all* observed differences are incorporated to the graph. However, all sequencing errors would then get included, which might result in a massive growth of the graph. In the second case, an online consensus caller could be similar to Ococo but with a support for reference graphs. In both cases, the underlying graph structure and the corresponding index must support updates.

Future work Implementing an efficient dynamic mapper remains an open problem, however most critical ingredients of such an implementation have already been developed. An online consensus calling algorithm has been implemented in the Ococo program within our

[1] https://github.com/vgteam/vg

work, and a promising dynamic k-mer index called SkipPatch has been recently reported in [553]. Altogether, implementing a dynamic mapper limited to substitution updates and "restricted" indel updates (see paragraph Non-substitution updates) appears nowadays as a challenging but mainly engineering task.

Several other algorithmic problems remain open when more general updates are considered. The main open problem is to design a more complex framework to maintain variation statistics in compact space through the mapping procedure, including information about arbitrary-length insertions and large-scale variants (inversions, transpositions, etc.). A related problem is to define an optimal strategy for reference updates, based on the collected statistics. This may involve building an appropriate probabilistic model, generalizing the stochastic strategy we considered in this work.

With a fully dynamic mapper available, the improvement of alignment quality will have to be evaluated for different tasks, such as variant calling, somatic mutation analysis, reference guided assembly, phylogenetic inference, or processing long reads. Several new perspectives of NGS data processing will open up, such as real-time mapping or real-time reference-based variant calling.

It would be also interesting to explore applications of online consensus calling other than dynamic mapping, e.g. to a distributed implementation of mapping with specific processor nodes dedicated to correcting the reference and rebuilding the index (see Hybrid approach).

Part III

k-mer-based metagenomic classification

Contents - Part III

Chapter 10

Overview

Metagenomics is a powerful approach to the study of environmental samples. In this chapter, we provide an overview of state-of-the-art metagenomic methods.

10.1 Introduction

The advent of high-throughput sequencing technologies (*Next-Generation Sequencing*, commonly abbreviated to NGS) revolutionized metagenomics, by avoiding the need of cloning the DNA and thus greatly facilitating the obtention of metagenomic samples, at the same time drastically decreasing its price. Early examples of metagenomic projects include the analysis of samples of seawater [602, 603], human gut [17], or soil [604]. Present-day metagenomic studies focus on various bacterial, fungal or viral populations, exemplified by the Human Microbiome project [605] that investigates microbial communities at different sites of human body.

Modern metagenomics deals with vast sequence datasets. On the one hand, metagenomic samples (*metagenomes*) obtained through NGS are commonly measured by tens or even hundreds of billions of bp [606, 607]. These sequences generally come from a number of different species, some of which either have a previously sequenced reference genome, or have a related sequenced species sufficiently close phylogenetically to determine this relationship by sequence comparison. Other sequences, however, may come from organisms that have no sufficiently close relatives with sequenced genomes, or from DNA fragments that show no significant similarity with any available genomic sequence. The *metagenomic classification* problem is to assign each sequence of the metagenome to a corresponding taxonomic unit, or to classify it as 'novel' (Figure 10.1).

A way to improve the accuracy of metagenomic classification is to match the metagenome against as large set of known genomic sequences as possible. With many thousands of completed microbial genomes available today, modern metagenomic projects match their samples against genomic databases of tens of billions of bp [458].

Alignment-based classifiers [117] proceed by aligning metagenome sequences to each of the known genomes from the reference database, in order to use the best alignment score as an estimator of the phylogenetic "closeness" between the sequence and the genome. This could be done with generic alignment program, such as BLAST [164], BLASTZ [170], or BLAT [178]. While this approach can be envisaged for small datasets (both metagenome and database) and is actually used in such software tools as Megan [608] or PhymmBL

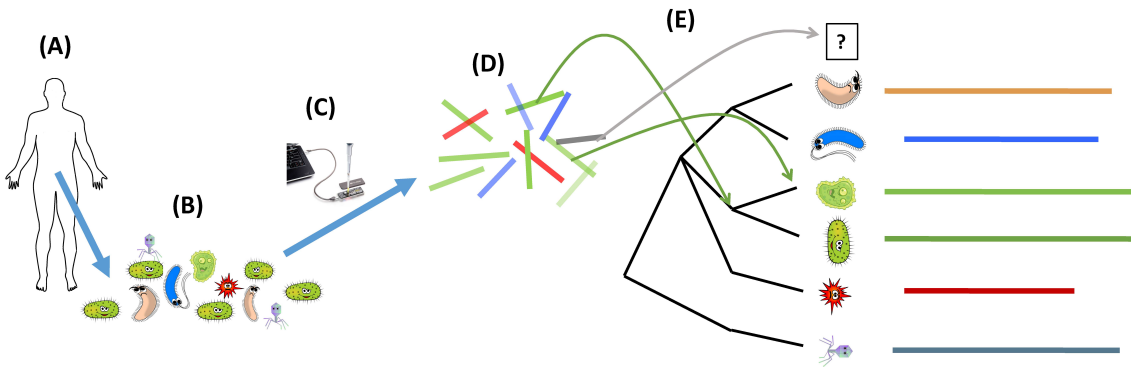


Figure 10.1: **Metagenomic classification.** From a given environment (**A**), a biological sample containing genetic material is extracted (**B**) and sequenced (**C**). The obtained NGS reads (**D**) are then assigned to a taxonomic tree or classified as 'novel' (**E**). The assignments can be performed either by aligning reads to the reference genomes (*alignment-based methods*), or based on the composition of k-mers in the reads and in the reference genomes (*alignment-free methods*).

[609] (see [117] for more information), it is unfeasible on the scale of modern metagenomic projects. On the other hand, there exists a multitude of specialized tools for aligning NGS reads – BWA [181], Novoalign[1], GEM [223], Bowtie [182] and many others (see their list in Chapter 3) – which perform alignment at a higher speed and are adjusted to specificities of NGS-produced sequences. Still, aligning multimillion read sets against thousands of genomes remains computationally difficult even with optimized tools. Furthermore, read alignment algorithms are usually designed to compute high-scoring alignments only, and are often unable to report low-quality alignments. As a result, a large fraction of reads may remain unmapped [610].

Several techniques exist to reduce the computational complexity of this approach. One direction is to pre-process the metagenomic sample in order to assemble reads into longer contigs, potentially improving the accuracy of assignment. Assembly of metagenomic reads has been a subject of many works (see [611]) and remains a fragile approach, due to its error-proneness and high computational complexity. Overall, it appears feasible mostly for small-size projects with relatively high read coverage.

To cope with increasingly large metagenomic projects, *alignment-free methods* have recently come into use. These methods do not compute read alignments, thus do not come with benefits of them, such as gene identification. Alignment-free sequence comparison is in itself an established research area, reviewed in a recent dedicated special issue [612]. Most of alignment-free comparison methods are based on the analysis of words, usually of fixed size (k-mers), occurring in input sequences. A popular approach is to compute the distance between *frequency vectors* of all k-mers in each of the sequences. In the context of metagenomics, however, when one of the sequences is short (NGS read), the analysis is based on the shared k-mers, without taking into account their multiplicities in reference genomes. This is also dictated by the prohibitive computational cost of computing and storing k-mer multiplicities for metagenomics-size data.

Two recently released tools – LMAT [614] and Kraken [458] – perform metagenomic

[1] http://www.novocraft.com/

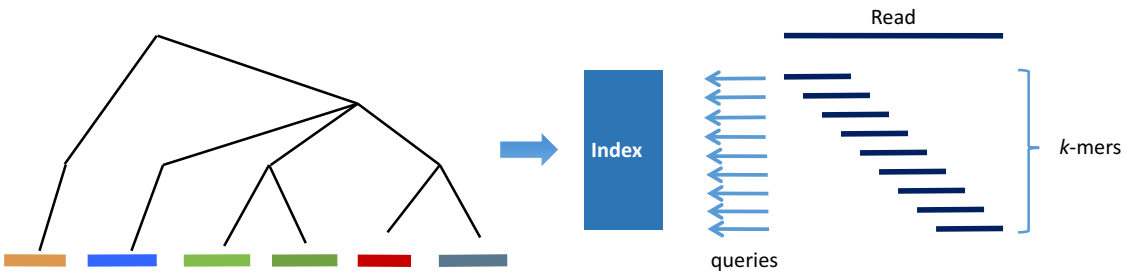


Figure 10.2: ***k*-mer index for alignment-free metagenomic classification.** An index is built from a user-specified taxonomic tree and the set of reference genomes placed in the leaves of the tree. When a k-mer is queried, the index retrieves either a single node in the tree (e.g., Kraken [458]), or a list of nodes (e.g., ProPhyle [Chapter 12]), or a list of nodes with weights (e.g., Vowpal Wabbit [613]).

classification of NGS reads based on the analysis of shared k-mers between an input read and each genome from a pre-compiled database. Given a taxonomic tree involving the species of the database, these tools "map" each read to a node of the tree, thus reporting the most specific taxon or clade the read is associated with. Mapping is done by sliding through all k-mers occurring in the read and determining, for each of them, the genomes of the database containing the k-mer (Figure 10.2). Based on obtained counts and tree topology, algorithms [614, 458] assign the read to the tree node "best explaining" the counts.

10.2 State-of-the-art

Several other similar tools were published shortly after LMAT [614] and Kraken [458]. Here, we provide a short description of programs related to our work. We categorize them into three categories: *read classifiers* contains programs assigning individual reads; *abundance estimators* includes tools computing abundancies at some taxonomic level (e.g., at the level of species); *metagenome distance estimators* comprises utilities estimating distances between metagenomes from NGS reads directly, without any classification.

More information about these tools and useful comparative analyses can be found in dedicated studies [610, 615, 616, 617, 618, 619] from which especially [610] can be highly recommended. An in-depth overview of state-of-the-art has been published recently [607]. As an exhaustive list of papers, we also recommend the online list of papers and programs maintained by Jonathan Jacob [620].

10.2.1 Read classifiers

i) **Alignment-free read classifiers**

- **Kraken [458].** Kraken was the first tool enabling ultra-fast classification of large read sets against a taxonomic tree and it became a standard reference program for evaluation of metagenomic classification methods. Therefore, we describe it in more details than the other methods.
  Using a large hash table, Kraken stores LCA (lowest common ancestor) for each k-mer from the source database. The highly cache-optimized index enables to

quickly retrieve the LCA node for any queried k-mer. Classification then proceeds in two steps. Positions of all read's k-mers are identified in the tree, the heaviest root-to-node path is detected and the read is then assigned to this node. When multiple nodes with equally heavy paths exist, the read is assigned to their LCA. The default Kraken index for 2,787 bacterial genomes from the RefSeq database with k-mers of length 31 occupies 75 GB RAM, but approximately 120 GB is required for its construction. The performance reported on its website is higher than a million of assigned reads per second on a single core.

Compared to other methods, Kraken approach has two main advantages: it is extremely fast, and it enables index sampling. The superior performance of Kraken is mainly due its simplicity and perfect cache optimization (e.g., k-mers sharing the same minimizer are placed at proximal positions in memory). Since the main data structure is a hash table of k-mers, its sampling is easy, based on keeping only a fraction of the original entries. This can reduce Kraken memory footprint several folds. For instance, the Mini-Kraken database [458], containing every 19th k-mer, requires 4 GB RAM only.

- **LMAT [614].** LMAT implements an approach similar to Kraken. It also exploits LCAs and classifies reads based on shared k-mers. In addition to Kraken, it incorporates a more precise statistical model with score normalization using node-specific probabilities of random assignments. Moreover, LMAT is capable to create an index of the most informative k-mers. Nevertheless, its huge memory requirements do not allow its usage on vast majority of computers or even clusters. Indeed, the original version required 619 GB for the basic database of prokaryotic genomes. Recently, a new version called LMAT-ML has been presented [621]. It builds a small marker library such that LMAT can be run on computers with 24 GB RAM.
- **ProPhyle [Chapter 12].** ProPhyle is our metagenomic classifier, algorithmically similar to Kraken. It uses an index based on Burrows-Wheeler transform, which is smaller and more expressive at the same time.
- **Seed-Kraken [3, Chapter 11].** Seed-Kraken is our extension of the Kraken classifier. It improves the original approach using spaced seeds. The entire algorithm works similar to Kraken, with an exception of a minor modification in the assignment algorithm required by non-existence of canonical spaced k-mers for asymmetrical seeds. Seed-Kraken provides a better 'selectivity-sensitivity' trade-off than Kraken, but at a price of lower speed.
- **Centrifuge [622].** Centrifuge is a classifier based on BWT-index (implementation from Bowtie2 [184]). Using the NucMer algorithm [172] for detecting similar sequences, Centrifuge binarizes the taxonomic tree and propagates shared sequences up. Then it creates a highly compressed BWT-index. For instance, for a database of 4,300 prokaryotic genome, the reported index size is 4 GB only. However, the small size of the index is achieved by a lossy compression in the step of merging similar sequences.

  To classify reads, Centrifuge first finds sufficiently long exact matches between the read and the index, and computes species-level scores as squared lengths of obtained matches. From this information, assignments at other taxonomic levels can be obtained. Using the EM-algorithm, Centrifuge can also estimate abundancies at any taxonomic rank.

- **Kaiju [623].** Kaiju classifies reads at the protein level using BWT-index. It proceeds by translating reads into six possible reading frames, which are then split at stop codons. For the obtained fragments, Kaiju detects maximum exact matches with the database and reports the LCA of taxons with the longest matches. As protein coding genes occupy only small part of genomes, the resulting index can be extremely compact. Kaiju the memory footprint is only 6 GB for the default database of bacterial, archaeal, and viral protein sequences in RefSeq.
- **CLARK [624].** CLARK is a method based on sequence classification into predefined groups using discriminative k-mers and it does not make use of taxonomic trees. After building the database that is basically a hash table, CLARK matches k-mers from a read against this database. Memory usage of the standard version is fully comparable to Kraken, since approximately 70 GB RAM is used for a basic database of 2,752 bacterial genomes for genus level and 164 GB RAM is needed for index construction.

  A light version of CLARK called CLARK-l consumes 3 GB during classification and 4 GB during index construction. Recently, another modification of CLARK called CLARK-S, has been introduced [480]. The major conceptual difference is the usage of spaced seeds (hence, discriminative spaced k-mers). Memory requirements of CLARK-S are comparable to CLARK.
- **One Codex [625].** One Codex is a web platform for read classification using an algorithm similar to Kraken, also based on matching k-mers of length 31. The main difference from Kraken is a larger database containing 40,000 microbial genomes.
- **CoMeta [626].** CoMeta assigns reads to taxonomic trees based on the coverage criterion (see the next chapter for more details). The underlying indexing structure relies on Bloom filters. In the paper, experiments are performed with distinct k-mer lengths (from 15 to 30) and, for instance, bacterial database for $k = 30$ results in an index of size 40 GB.
- **Vowpal Wabbit for metagenomics [613].** The authors implemented a large scale machine learning-based linear classifier. They showed that, with k-mers of length ≤ 12, such a method can be competitive in speed and accuracy to other state-of-the-art approaches presented here.
- **SMART [627].** SMART aims at high scalability using a MapReduce searching strategy. It builds a database of k-mers of length 30, which is then stored using 256 hash-tables (k-mers are distributed according to first four bases). Read classification proceeds with k-mer matching in 256 parallel instances, merging results, and the final read assignment. Note that the program is capable to account for one mismatch in k-mer matching. Integrating the entire NCBI Genbank, SMART searches in the largest database from all classifiers presented in this section. Due to high memory and CPU requirements, the program can be set up on big clusters only.

ii) **Alignment-based read classifiers**

- **MetaPhlAn [628, 629].** MetaPhlAn is a read classifier based on Bowtie2 [184]. It builds a database of strongly conserved clade-specific sequences from 3,000 bacterial reference genomes, where clades can be of different taxonomic levels.

MetaPhlAn 2 extends the database to 17,000 different genomes, including viral and eukaryotic ones.

- **MEGAN [630, 608].** MEGAN4 is an integrative platform, which supports also metagenomic classification. Every read is assigned to the LCA of genomes it has been previously mapped to using BLAST [163].
- **MetaPhyler [631].** MetaPhyler builds a database from 31 phylogenetic marker genes. Then it uses BLASTX [163] to map reads to these sequence. Based on the obtained alignments, particular scores are to used to decide about the final taxonomy assignments.

iii) **Assembly-based read classifiers**

- **MetaCluster-TA [632].** MetaCluster-TA exploits the idea of assembling contigs in order to increase the annotation performance. It first assembles so-called virtual contigs, which are composed of reads coming from the same genome, but not necessarily from proximal regions. Using BLASTN [163], the obtained contigs are then clustered, merged and annotated.

iv) **Read classifiers combining multiple other classifiers**

- **WeVote [633].** WeVote is a method for metagenomic classification based on weighted voting. The program calls other classifiers implementing k-mer-based, marker-based and naive-similarity based approaches, and decides on the final read's classification based on the assignments obtained from these tools.
- **CSSS [634, 635].** Exploiting the nearest neighbor algorithm, CSSS classifies sequences using scores obtained from other alignment-based and alignment-free tools. The final score is computing using adaptive weighting.
- **PhymmBL [636, 609].** PhymmBL is a hybrid classifier combining interpolated Markov models [637] with BLAST [163].

10.2.2 Abundance estimators

i) **Alignment-free abundance estimators**

- **GOTTCHA [638].** GOTTCHA first builds a database of k-mers of length 24 specific for a fixed taxonomic rank. Then, by breaking into non-overlapping k-mers and with use of BWA [181], it matches reads against this database and computes abundancies for individual taxonomic levels.
- **Kallisto [459].** Though originally developed for RNA-seq quantification [460], Kallisto can be also used for abundance estimation for metagenomic datasets at various taxonomic levels [459]. The tool is easy to use and highly efficient, with memory footprint of approximately 60 GB for a database of 2,300 bacterial genomes. Kallisto does not annotate individual reads nor takes taxonomic trees into account.
- **Bracken [639].** Using Bayesian approach, Bracken estimates species abundancies from Kraken read assignments. The tool has been developed as a reaction to [459].
- **MetaPalette [640].** Using linear mixture models, MetaPalette derives abundancies from k-mer frequencies computed from the entire read sets.

ii) **Alignment-based abundance estimators**

- **MEGAN CE [641].** MEGAN CE is a major revision of MEGAN, a tool for metagenomic classification described above. Instead of BLAST, it aligns reads using DIAMOND [359], a program for mapping reads to protein databases. Moreover, it provides a server application called MeganServer.
- **BIB [642].** BIB performs Bayesian identification of strain abundancies from NGS reads using Bowtie 2 [184]. Index construction proceeds by detection of strain-specific core genomes and building a read mapping index. The former is done by the clustering of strains and multiple sequence alignment. To compute abundancies, BIB maps reads to the obtained core genomes and estimates frequencies of individual strains from the obtained alignments.
- **SLIMM [643].** Given a set of references for an individual taxonomic group of interest, SLIMM starts with building a non-redundant database. YARA [211] or another read mapper is then used to map all reads to that database. The classification procedure takes the obtained alignments, excludes insufficiently covered genomes, and moves every read mapped to multiple genomes to their LCA. Finally, SLIMM computes abundancies based on the read assignments from alignments.
- **MetaScope [479].** MetaScope uses SSAS, a dedicated spaced-seed-based read mapper, to align reads to Genbank reference genomes. Then it re-assigns those reads aligned to multiple genomes using weighted LCA, and creates an overview XML file. MetaScope is not publicly available, yet.
- **WGSQuikr [644].** WGSQuikr computes frequencies of k-mers of length 7 in a database of bacterial genomes, and then derives a vector of abundancies by solving a system of linear equations in Matlab.
- **MetaFlow [645].** MetaFlow introduces a sophisticated method of abundance estimation from BLAST alignments. The problem is reduced to matching of bipartite graphs constructed from the provided alignments, which is then solved using min-cost network flows.

10.2.3 Metagenome distance estimators

- **Simka [646].** Simka was developed to simplify the analysis of large metagenomic datasets coming from the Tara ocean project [14]. Their processing is complicated by the fact that reference sequences are not available for a vast majority of ocean genomes. As a consequence, large metagenomic data sets have to be analyzed without any reference database in hand. To deal with this obstacle, Simka proceeds with k-mer based methods comparing metagenomic read sets and provides a well-scaling method to compute various ecological distances.

- **MASH [647].** MASH is a k-mer-based method to estimate metagenome distances using MinHash ([384, 123]), a technique based on locality-sensitive hashing. Generally speaking, the program estimates the Jaccard index of two metagenomic datasets with moderate memory and CPU requirements.

- **MetaFast [648].** First, MetaFast estimates the similarity of pairs of metagenomes using de-Bruijn graphs. It constructs the de-Bruijn graphs from the input metagenomes,

detects non-branching paths, merges the resulting graphs into a single one, and then extracts sufficiently big components that are used to compute the metagenome similarity.

- **kWIP [649].** kWIP is a program to estimate the genetic relatedness of species from NGS reads using k-mer weighted inner products. First, kWIP counts k-mers in all datasets, stores them in a probabilistic data structure, and computes population-wide aggregated k-mer frequencies. Finally, it computes the similarity as inner products of frequencies vectors, normalized by Shannon entropy.

10.3 Goal of our work

In our work, we improve k-mer-based methods for metagenomic classification. Specifically, we consider the problem of assigning NGS reads to a taxonomic tree, i.e., the approach used by Kraken [458], LMAT [614], Centrifuge [622], CLARK [624], One Codex [625], CoMeta [626], Kaiju [623], and SMART [627].

In Chapter 11, we demonstrate that spaced seeds can strongly improve the precision of metagenomic classification. We provide a comparative analysis of several measures (hit count, Jaccard index, and coverage) combined with contiguous and spaced k-mers. We introduce Seed-Kraken, a modification of Kraken using spaced seeds, and show that it significantly improves the classification precision compared to Kraken.

In Chapter [Chapter 12], we provide ProPhyle, a k-mer-based metagenomic classifier using BWT-index. The strategy is based on a bottom-up propagation of k-mers in the tree, assembling contigs at each node and matching using a standard full-text search. Unlike Kraken, this method provides lossless k-mer indexing and the resulting index is several folds smaller than Kraken's one.

Chapter 11

Spaced seeds for metagenomics

We show that the metagenomics classification based on the analysis of shared k-mers can be improved by using *spaced k-mers* rather than contiguous k-mers. We support this thesis through a series a different computational experiments, including simulations of large-scale metagenomic projects. We also present Seed-Kraken, a modification of the Kraken classifier using spaced seeds. Scripts and programs used in this study, as well as supplementary material, are available from http://github.com/gregorykucherov/spaced-seeds-for-metagenomics.

11.1 Introduction

The idea of using spaced k-mers goes back to the concept of *spaced seeds* for *seed-and-extend* sequence comparison [463, 462]. There, the idea is to use as a seed (i.e. local match triggering an alignment) a sequence of matches interleaved with "joker positions" holding either matches or mismatches. The pattern specifying the sequence of matches and jokers is called the spaced seed. Remarkably, using spaced seeds instead of contiguous seeds significantly improves the sensibility-selectivity trade-off with almost no incurred computational overhead. This has been first observed in [463] and then extended and formally analysed in a series of further works, see [481, 195] and references therein.

Recently, it has been reported in several works that spaced seeds bring an improvement in alignment-free comparison as well. In [464], it is shown that comparing frequency vectors of *spaced k-mers* (k-mers obtained by sampling must-match positions of one or several spaced seeds), as opposed to contiguous k-mers, leads to a more accurate estimation of phylogenetic distances and, as a consequence, to a more accurate reconstruction of phylogenetic trees. In [465], the authors studied another measure – the number of pairs of matching (not necessarily aligned) spaced k-mers between the input sequences – and showed that it provides an even better estimator of the phylogenetic distance. In [468], it is shown that the number of hits of appropriately chosen spaced seeds in *aligned* sequences and their *coverage* (i.e. the total number of matched positions covered by all hits) provides a much better estimator for the alignment distance than the same measures made with contiguous seeds. From a machine learning perspective, works [466, 467] show that spaced seeds provide better string kernels for SVM-based sequence classification, confirmed by experiments with protein classification (see also [468]).

In this work, we show that using spaced k-mers significantly improves the accuracy of metagenomic classification of NGS reads as well. To support this thesis, we consider

different scenarios. As a test case, we first study the problem of discriminating a read between two genomes, i.e. determining which of the two genomes is "phylogenetically closer" to the read. We then analyse the correlation between the quality of an alignment of a read to a genome and the seed count for this read, defined either as the number of hits or as the coverage. This analysis provides an insight into how well one can estimate the similarity between a read and a genome out of k-mer occurrences. Finally, we make a series of large-scale metagenomic classification experiments with Kraken software [458] extended by the possibility of dealing with spaced seeds. These experiments demonstrate an improved classification accuracy at the genus and family levels caused by the use of spaced seeds instead of contiguous ones.

11.2 Preliminaries

A spaced seed is a binary pattern over symbols `#` and `-` denoting match and joker respectively. For example, seed `#-##` specifies a match followed by either match or mismatch followed by two consecutive matches. A seed acts as a mask for comparing short oligonucleotides, for example sequences `gaat` and `gcat` differ at the 2nd position, but they match via seed `#-##` as the 2nd position is masked out. The number of `#`'s in a seed, called *weight*, defines the number k of matching nucleotides. In the above example, $k = 3$ and the matching (spaced) k-mer is `gat`. In a slightly different terms, a seed is a pattern that specifies a small part of a gapless alignment seen as a binary sequence of matches and mismatches. For example, seed `#-##` occurs in (or *hits*, as usually said) any alignment containing a match followed by another two matches shifted by one position.

When spaced seeds are used for sequence alignment within the *seed-and-extend paradigm* [195], a pair of matching k-mers (or, sometimes, a matching sequence of several closely located k-mers) indicates a potential alignment of interest in which the two k-mers are aligned together. When spaced seeds are used for alignment-free sequence comparison [612], the goal is to estimate the similarity of two sequences based on the *number* of matching k-mers, with no or limited information about their positions in one or both sequences. This measure can be formalized in several different ways.

In the context of metagenomic classification of NGS reads, the goal is to estimate the evolutionary distance between a short read and a long sequence (genome), which can be modeled by the best alignment score of the read against the sequence. Due to large genome size and large number of reference genomes involved in computations, one of the constraints is to avoid keeping track of positions of k-mers in genomes. We can only afford constructing an index to quickly answer queries whether or not a k-mer occurs in a given genome, without information on its positions. On the other hand, k-mer positions in the query read can be included to the analysis. Therefore, we have to derive our estimation from the number of k-mers shared between the read and the genome, together with their positions in the read alone.

One simple estimator is the number of k-mers in the read that occur in the target genome[1], we call this measure the *hit number*. However, one may want to favor cases when matching k-mers cover a larger part of the read, vs. those with matching k-mers clumped together due to overlaps. This leads to the concept of *coverage* [468], earlier used in seed-based alignment algorithms as well [650, 651]. The coverage is the total number of positions covered by all matching k-mers. For example, consider seed `#-##` and read

[1] Here identical k-mers occurring at distinct positions in the read are considered distinct.

gaatcagat. Assume the seed hits at positions 1,4,6, i.e. k-mers g-at, t-ag and a-at occur in the target genome (joker symbol is shown for the sake of clarity). Here, the hit number is 3, and the coverage is 7 as seven positions are covered by hits, namely positions 1,3,4,6,7,8,9. Hit number and coverage are two estimators studied in this work.

11.3 Results

11.3.1 Binary classification

From the machine learning viewpoint, hit number and coverage can be viewed as instances of kernel functions for sequence data (see e.g. [652]). Our first step was to compare their capacity w.r.t. a simple binary classification task. Assume the (impractical) case when our database contains only two genomes. Given a read, we have to decide which of the two genomes the read is closer to. How good can we be at that? Which kernel works better for this task? And are spaced seeds better than contiguous seeds here?

Classifying aligned reads

As mentioned in Introduction, the "distance" of a read to a genome translates naturally to the score of the best alignment of the read to the genome. Given two genomes, we want to tell which of them is closer to a given read.

Consider alignments A_s, A_l of a random read to the two genomes, and assume they are gapless and have mismatch probabilities p_s, p_l respectively, with $p_s < p_l$. Throughout this paper the read length is set to 100, a typical length of Illumina read. Therefore, the alignments can be thought of as random binary strings of length 100 of matches and mismatches, with mismatch probabilities p_s and p_l respectively. Given a seed, we compute two counts C_s and C_l on alignments A_s and A_l respectively, where by 'count' we mean, unless otherwise stated, either the hit number or the coverage. For example, if the seed is #-## and the alignment 111101111 (1 stands for match and 0 for mismatch), then the hit number is 3 and the coverage is 7. Note that in this model, common (spaced) k-mers are assumed to occur necessarily at the same position in the read alignment, although in reality, a k-mer of the read may not be aligned with the same k-mer in the genome. However, in this first experiment, we abstract from this fact.

If $C_s > C_l$ (resp. $C_s < C_l$), then we report a correct (resp. incorrect) classification, otherwise a tie is reported. By iterating this computation, we estimate the probability of correct/incorrect classification for each parameter set.

The results are presented in Tables 11.1a,11.1b,11.1c. In all cases, spaced seeds show a better classification power. In some cases, the difference is striking: for example, if we want to discriminate between alignments with mismatch probabilities 0.1 and 0.2 (Table 11.1b) using seeds of weight 22, then a spaced seed yields 86% of correct classifications (coverage count), whereas the contiguous seed correctly classifies only 65% of cases, whereas the fraction of incorrect classifications is essentially the same. The results also show a slight edge of the coverage count over the hit number, which suggests the superiority of the latter that will be confirmed later on in other experiments.

Classifying unaligned reads

Let us now turn to a more practical setting, where we want to classify reads coming from a genome G between two other genomes G_1 and G_2 based on the phylogenetic closeness.

(a) $p_s = 0.05$, $p_l = 0.1$

	contiguous		spaced	
weight	Hits	Cover	Hits	Cover
16	.85/.14	.85/.14	.87/.11	.88/.11
18	.84/.14	.85/.14	.87/.12	.88/.11
20	.83/.15	.84/.15	.87/.12	.87/.12
22	.82/.16	.83/.15	.86/.12	.87/.12
24	.80/.16	.81/.15	.85/.12	.87/.12

(b) $p_s = 0.1$, $p_l = 0.2$

	contiguous		spaced	
weight	Hits	Cover	Hits	Cover
16	.86/.09	.87/.09	.93/.05	.94/.05
18	.81/.08	.81/.08	.91/.05	.92/.05
20	.74/.07	.74/.07	.89/.05	.90/.05
22	.65/.06	.65/.06	.85/.05	.86/.05
24	.55/.04	.56/.04	.79/.04	.81/.05

(c) $p_s = 0.2$, $p_l = 0.3$

	contiguous		spaced	
weight	Hits	Cover	Hits	Cover
16	.40/.06	.40/.06	.63/.07	.64/.07
18	.28/.04	.28/.03	.50/.05	.50/.05
20	.18/.02	.18/.02	.37/.03	.37/.03
22	.12/.01	.12/.01	.25/.02	.26/.02
24	.08/.00	.08/.00	.17/.01	.18/.01

Table 11.1: **Classification of aligned reads.** Each entry contains a pair "Probability of correct classification/Probability of incorrect classification". The remaining fraction estimates the probability of a tie. Spaced seeds used for hit number: `###-##-#-#---#-#-#--#--#####` (weight 16), `####-##--##--#--#-#-#-##-####` (weight 18), `#####-###---##--#--#-#-#-##-####` (weight 20), `######--####----#--##-#-#--#######` (weight 22), `#######--####----#--##-#-#--########` (weight 24). Spaced seeds used for coverage: `###-###--#-#--#-##--#####` (weight 16), `###-#--###-##--#-#--###-####` (weight 18), `###-#-##-#-##--##--###-#-#####` (weight 20), `####-##-#-#-##-###-#-#--##--#####` (weight 22), `####-##-#-#-##-###-#-#---###--######` (weight 24).

To study this, we implemented the following experimental setup. Using DWGsim read simulator[2], we generate single-end Illumina-like reads from genome G. In all experiments, we assumed 1% of base mutations (substitutions only), and 2% of sequencing errors (DWGsim options `-e 0.02 -r 0.01 -R 0`). Given a seed, (contiguous or spaced) k-mers of G_1 and G_2 are indexed to support existence queries only. For each read, all k-mers are queried against G_1 and G_2 and corresponding counts C_1 and C_2 are computed. If $C_1 > C_2$ (resp. $C_1 < C_2$), the read is classified to be closer to G_1 (resp. G_2), otherwise a tie is reported. Besides considering absolute counts (hit number and coverage), we also considered hit number normalized by the number of distinct k-mers in the corresponding genome (computed at the indexing stage). This measure approximates the Jaccard index [503] and reflects the Bayesian probability of seeing a k-mer relative to the "k-mer-richness" of genomes.

Note that the counts are here computed relative to the whole genome, as it is done in the approach of [614, 458] (see Introduction). This means that the k-mers occurring in the read are looked up in the whole genome, without guarantee, however, that these k-mers are closely located in the genome and contribute to the same read alignment. This makes the seed weight an important parameter, as seeds of low weight may result in a high read count which does not evidence any alignment of the read, due to random character of the k-mer hits.

We experimented with bacterial genomes belonging to *Mycobacterium* genus. Members of this genus present low interspecies genetic variability and their phylogeny remains uncertain [653].

If G coincides with one of G_1, G_2, i.e. reads have to be classified between its source genome and another genome, then all estimators correctly classify nearly all reads as soon as G_1 and G_2 are genomes of distinct species. For example, classifying reads obtained from *Mycobacterium tuberculosis* (H37Rv, acc. NC_018143) against *M. tuberculosis* itself and *M.avium* (104, NC_008595) led to more than 99% of correct classifications for all estimators.

The case when G is distinct from G_1, G_2 appears more interesting. It corresponds to the real-life situation when reads to be classified can come from a genome that is not represented in the database. Here we expect our procedure to determine whether G is phylogenetically closer to G_1 or to G_2.

For example, we classified *M.vanbaalenii* (PYR-1, NC_008726) reads against *M.smegmatis* (MC2 155, NC_018289) and *M.gilvum* (PYR-GCK, NC_009338) genomes. Alternative phylogenies given in [653, Fig.1-4] imply different evolutionary relationships among these three species. Our results, shown in Table 11.2, suggest that *M.vanbaalenii* is closer to *M.gilvum* than to *M.smegmatis*. For non-normalized hit number and coverage estimators, this conclusion is supported by seeds of weight 16 or more, while weight 14 supports the opposite conclusion. This is due to spurious hits that become dominating when the weight drops to 14, and to the larger size of *M.smegmatis* genome (6.99 Mbp) compared to *M.gilvum* (5.62 Mbp). This effect is corrected by Jaccard index due to normalization by the number of distinct k-mers (6.09 M for *M.smegmatis* vs. 4.96 M for *M.gilvum* for the spaced seed of weight 14). Overall, we observe a significantly sharper discrimination produced by spaced seeds compared to contiguous seeds.

We also performed a series of experiments with the large and genetically variable *Bacillus* genus. Table 11.3 shows a demonstrative experiment with members of *Bacillus cereus* group: *B.thuringiensis* (serovar konkukian 97-27, NC_005957), *B.anthracis* (Ames,

[2] https://github.com/nh13/DWGSIM

	weight				
	14	16	18	20	22
contig hit nb	52/41	39/48	24/37	11/24	07/17
contig cover	54/42	44/47	25/37	11/24	06/17
contig Jaccard	30/70	35/61	23/42	11/26	06/18
spaced hit nb	51/40	34/47	20/40	12/32	08/23
spaced cover	53/42	39/51	21/42	12/32	08/25
spaced Jaccard	28/72	32/66	20/50	11/33	08/27

Table 11.2: **Classification of unaligned reads.** Classification of *Mycobacterium vanbaalenii* reads against *Mycobacterium smegmatis* and *Mycobacterium gilvum* genomes. Each entry contains a pair "Fraction (in %) of reads classified closer to *M.smegmatis* / Fraction of reads classified closer to *M.gilvum*".

	weight				
	14	16	18	20	22
contig hit nb	83/14	81/11	79/09	77/08	76/08
contig cover	78/17	80/12	79/09	77/08	76/08
contig Jaccard	87/13	87/11	85/09	83/08	82/08
spaced hit nb	83/13	82/11	80/09	79/09	79/08
spaced cover	80/15	81/11	80/09	79/09	79/08
spaced Jaccard	88/12	88/11	85/09	84/08	84/08

Table 11.3: **Classification of unaligned reads.** Classification of *Bacillus thuringiensis* reads against *Bacillus anthracis* and *Bacillus cereus* genomes. Each entry contains a pair "Fraction (in %) of reads classified closer to *B.anthracis* / Fraction of reads classified closer to *B.cereus* ".

	weight				
	14	16	18	20	22
contig hit nb	47/40	24/25	04/07	0.9/3.8	0.3/2.4
contig cover	49/46	24/25	04/07	0.9/3.8	0.3/2.4
contig Jaccard	37/62	24/30	04/08	0.8/4.0	0.3/2.6
spaced hit nb	49/33	22/27	04/10	1.0/5.4	0.4/3.3
spaced cover	53/39	23/27	04/10	1.0/5.4	0.5/3.3
spaced Jaccard	41/59	22/35	04/10	0.9/4.6	0.4/3.5

Table 11.4: **Classification of unaligned reads.** Classification of *Bacillus licheniformis* reads against *Bacillus anthracis* and *Bacillus pumilus* genomes. Each entry contains a pair "Fraction (in %) of reads classified closer to *B.anthracis* / Fraction of reads classified closer to *B.pumilus* ".

NC_003997), and *B.cereus* (ATCC 14579, NC_004722). The three bacteria are close to the point of being considered to be different lineages of a single *B.cereus* species [654]. The results provide a strong support that the *B.thuringiensis* strain is closer to the *B.anthracis* strain than to the *B.cereus* strain, which agrees with phylogenies reported in the literature [655]. Indeed, the *B.thuringiensis* strain and the *B.anthracis* strain have a much higher pairwise identity rate than the former has with the *B.cereus* strain (estimated DNA-DNA hybridization distance 81% vs. 45%, as computed by GGDC [656]).

However, for species with low sequence similarity, a large majority of reads may have no hits to either genome, and only a small fraction of reads may reveal a significant difference in distance. This situation is illustrated in Table 11.4 showing the results of classification of *B.licheniformis* (ATCC 14580, NC_006270) reads against *B.anthracis* (Ames, NC_003997) and *B.pumilus* genomes (SAFR-032, NC_009848). The results support a higher similarity of *B.licheniformis* to *B.pumilus* than to *B.anthracis*, but the difference is revealed on a very small fraction of reads. The conclusion, however, is significant as those reads represent the majority of reads having any hits to one of the genomes. As with the previous example, this result is confirmed by previously reported phylogenies [655].

In all our experiments reported in this section, spaced seeds showed a better classification capacity. The difference is especially significant in "nontrivial" cases involving relatively dissimilar genomes, such as those illustrated by Tables 11.2 and 11.4. While the difference between hit number and coverage estimators appeared insignificant (in agreement with results of Section 11.3.1), the Jaccard index generally provides a more distinct discrimination and, combined with a spaced seed, appears to be the best estimator.

11.3.2 Correlation of counts with alignment quality

In metagenomic projects, reads to be classified do not necessarily come from genomes stored in the database, but can come from genomes of other species. These species can be genetic variants of species of the database, such as different strains of the same bacteria, but can also come from organisms represented in the database only at the rank of genus or family, or may even have no representatives at all at low taxonomic ranks. Therefore, an accurate mapping of a read to a corresponding clade requires not just assigning it to the appropriate sampled genome, but estimating its distances to each of the genomes in order to locate its position within the whole taxonomic tree.

With this motivation, we turned to the question how well the measured counts correlate

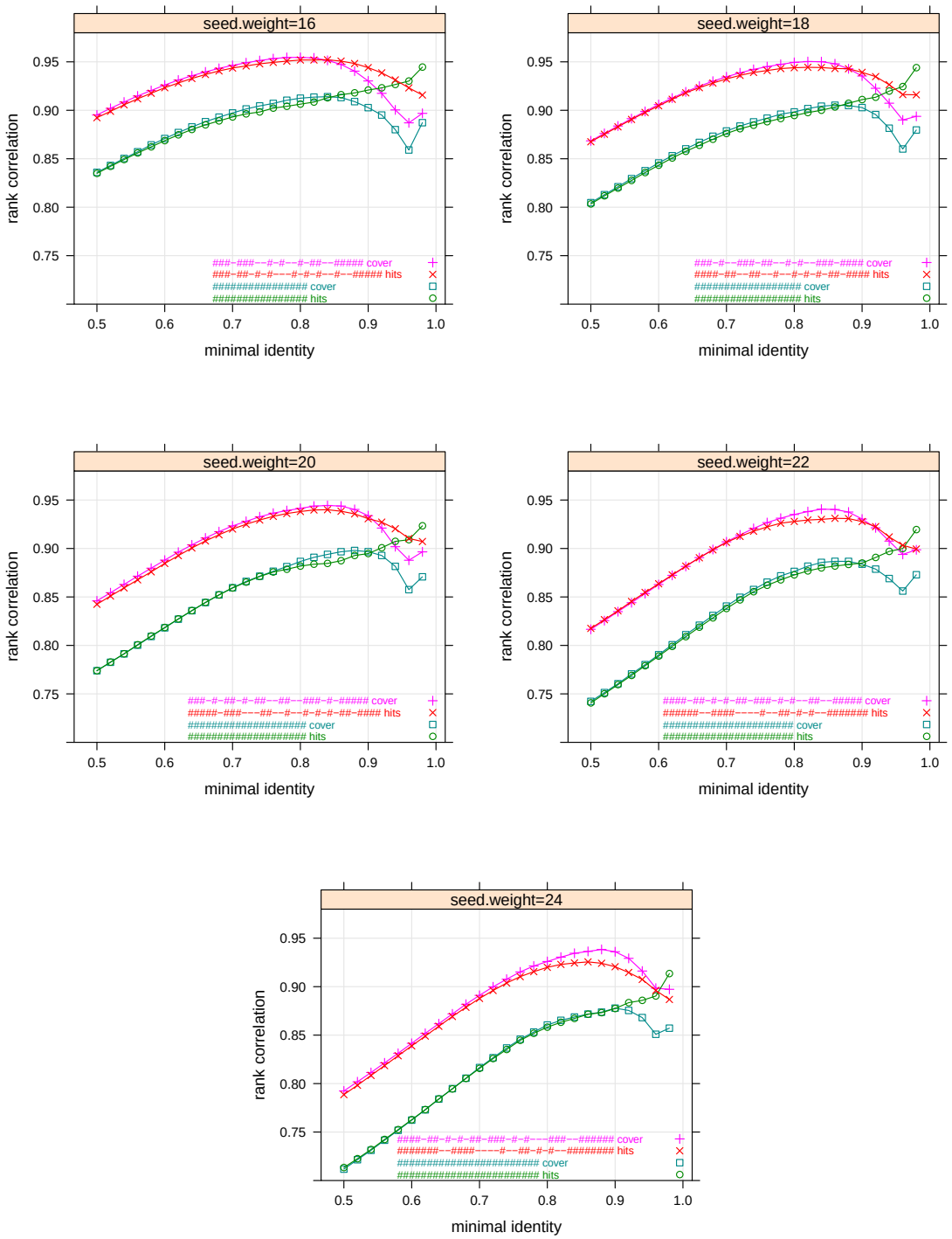


Figure 11.1: **Correlation between score and hit counts.** Spearman's rank correlation between score and counts, depending on the minimal identity rate.

with the alignment score. For a fixed minimal identity rate p_{id}, we randomly sampled gapless alignments of length 100 with identity rate from interval $[p_{id}..1]$, and collected pairs (number of mismatches, count), where, as before, 'count' stands for either the number of hits or the coverage of a given seed. For these data, we computed Spearman's rank correlation.

Results are presented on plots in Figure 11.1. They show that when the identity rate of alignments takes a large range of values (minimal id rate smaller than ≈ 0.9), spaced seeds yield a significantly higher correlation than contiguous seeds, for both hit-number and coverage counts. Furthermore, the coverage count slightly outperforms the hit-number count, especially for spaced seeds and larger weights.

For high-similarity alignments, however, the picture changes: the coverage count loses its performance, with its correlation value sharply decreasing. Furthermore, the correlation of hit-number goes down as well for spaced seeds, while it continues to grow for contiguous seeds ending up by reaching and even slightly outperforming the one for spaced seed. This is due to a larger span of spaced seeds and to their combinatorial properties that cause the hit number values to be less sharply concentrated at certain values, and therefore to be less well correlated with the number of mismatches.

In conclusion, while spaced seeds provide a much better estimator for alignments whose quality ranges over a large interval, for high-quality alignments ($> 90\%$ of identity), the hit number of contiguous seed becomes a better estimator. The superiority of hit-number over coverage for high-quality alignments has also been reported in [468]. Along with Spearman's correlation, we also made an analysis of mutual information computed on the same data (data not shown) that confirmed the above conclusions.

11.3.3 Correlation on real genomes

To validate the conclusions of the previous section in a real-life metagenomics framework and to analyse more closely how well different counts for a read correlate with the best alignment of the read to a real genomic sequence, we implemented another series of experiments.

Given a genome G, we generated a set of Illumina-like single-end reads by selecting random substrings of G of length L ($L = 100$) and introducing k mismatch errors, with k drawn randomly between 1 and 20 for every read. For each read, we computed the counts – hit number and coverage – with respect to genome G under a given seed, similar to Section 11.3.1. Collected data have then been analysed.

This experimental setup has been applied to *Mycobacterium tuberculosis* genome, a typical result (seed weight 20) is shown in Figure 11.2. Each plot shows the density of reads for each pair (number of mismatches, count), depending on the seed (contiguous or spaced) and count (hit-number or coverage). Spearman's and Pearson's correlation coefficients are shown for each plotted dataset.

The plots clearly illustrate the advantage of spaced seeds over contiguous seeds for estimating the alignment quality. Plots for contiguous seeds are more blurred whereas plots for spaced seeds demonstrate a better correlation between the two values. This is confirmed by the absolute values of Spearman's rank correlation coefficient that are significantly higher for spaced seeds, indicating a better statistical dependence. This is further illustrated in Figure 11.3 which shows the same data through the average curve and 95% confidence band. It confirms that spaced seeds produce a dependence with lower deviation from the mean, compared to contiguous seeds.

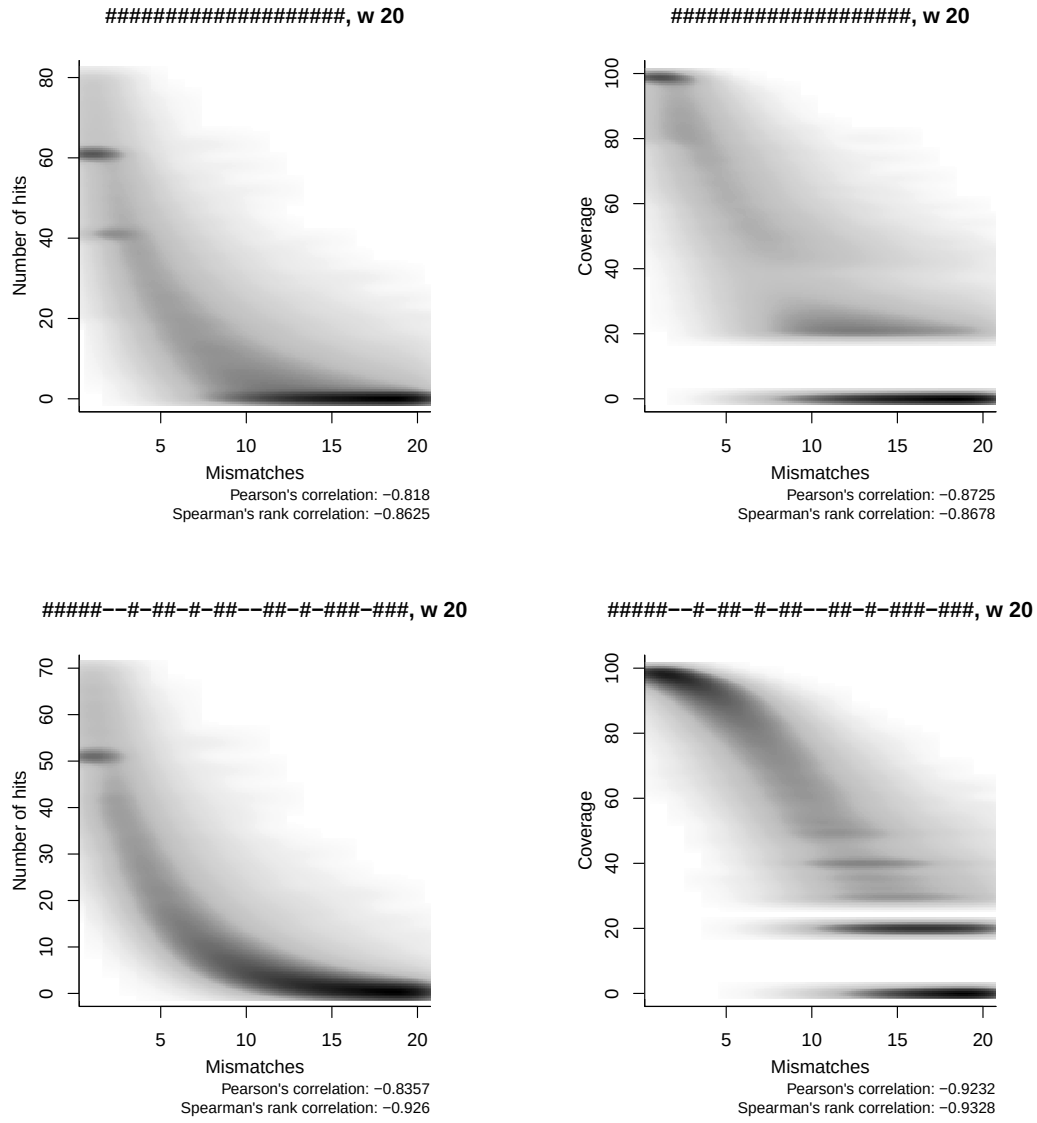


Figure 11.2: **Hit number and coverage vs. number of mismatches.** Hit number (left plots) and coverage (right plots) depending on the number of mismatches in randomly generated reads. Seed is shown above the plot, and Spearman's and Pearson's correlations are shown below. Grayscale shows the density of reads. Experiments made on *M.tuberculosis* genome.

Comparing hit-number and coverage estimators, we observe that coverage yields a slightly better Spearman correlation and a significantly better Pearson correlation, due to a convex shape of the dependence, compared to the more straight dependence for the coverage.

This analysis has been done for several other bacterial genomes, producing similar results. Plots for other genomes and other seed weights can be found at https://github.com/gregorykucherov/spaced-seeds-for-metagenomics.

11.3.4 Large-scale experiments

In order to validate the advantage of spaced seeds in large-scale metagenomic projects, we modified Kraken software [458] to make it work with spaced seeds rather than with contiguous seeds only. The limitation of this comparison is that it only allows estimating the effect of using spaced seeds combined with the Kraken algorithm, and within its implementation. On the other hand, this procedure allows us to estimate the effect of spaced seeds in an unbiased manner, by keeping unchanged all other factors that might influence the results.

Our extended implementation, that we call Seed-Kraken, allows the user to specify a spaced seed as a parameter. For a set of genomes, a database of spaced k-mers matching the seed is constructed, which is later used to classify reads through the original Kraken algorithm. Since Kraken uses k-mer counting Jellyfish program [448] as the k-mer indexing engine, we had also to modify Jellyfish to allow it to deal with spaced k-mers.

Integration of spaced seed into Kraken required a minor modification of the way Kraken deals with complementary sequences. In Kraken, complementary k-mers have a single representative in the index, the lexicographical smallest of the two. With spaced seeds, dealing with complementary sequencing is more delicate, as the complement of a spaced k-mer does not match the same seed but its inverse. To cope with this, we modified Kraken to index each distinct k-mer. We then processed each read in direct and complementary directions separately and select the one which produced more hits. Compared to original Kraken, this procedure takes more index space (additional $\sim 5\%$ in practice) and doubles k-mer query time.

We compared the performance of Kraken and Seed-Kraken on several datasets. First, we performed experiments with three simulated metagenomes HiSeq, MiSeq, and simBA-5 introduced in the original work [458], each containing 10,000 sequences. Furthermore, we created a dataset from Human Microbiome Project data by randomly selecting 50,000 sequences from SRS011086 Tongue dorsum metagenomic sample[3]. Here we only report on results for MiSeq and HMPtongue datasets and refer to the supplementary material for a complete account including results for HiSeq and simBA-5. MiSeq is a merge of Illumina reads of 10 bacterial genomes, and HMPtongue is a random sample of real Illumina whole-metagenome sequences.

Due to resource limitations, the database we used in MiSeq experiments was half of the size of the Kraken's default database (which requires 75 GB of RAM). Our database was obtained by choosing a single representative strain of each bacteria species, except for the species from HiSeq and MiSeq metagenomes for which all strains were included. Overall, this represented 915 genomes of total size 3.3 GB. For HMPtongue dataset, this database was extended with a subset of HMP reference library, 0.8 GB in total, including references for the selected 50,000 sequences.

[3] http://hmpdacc.org/HMSCP/

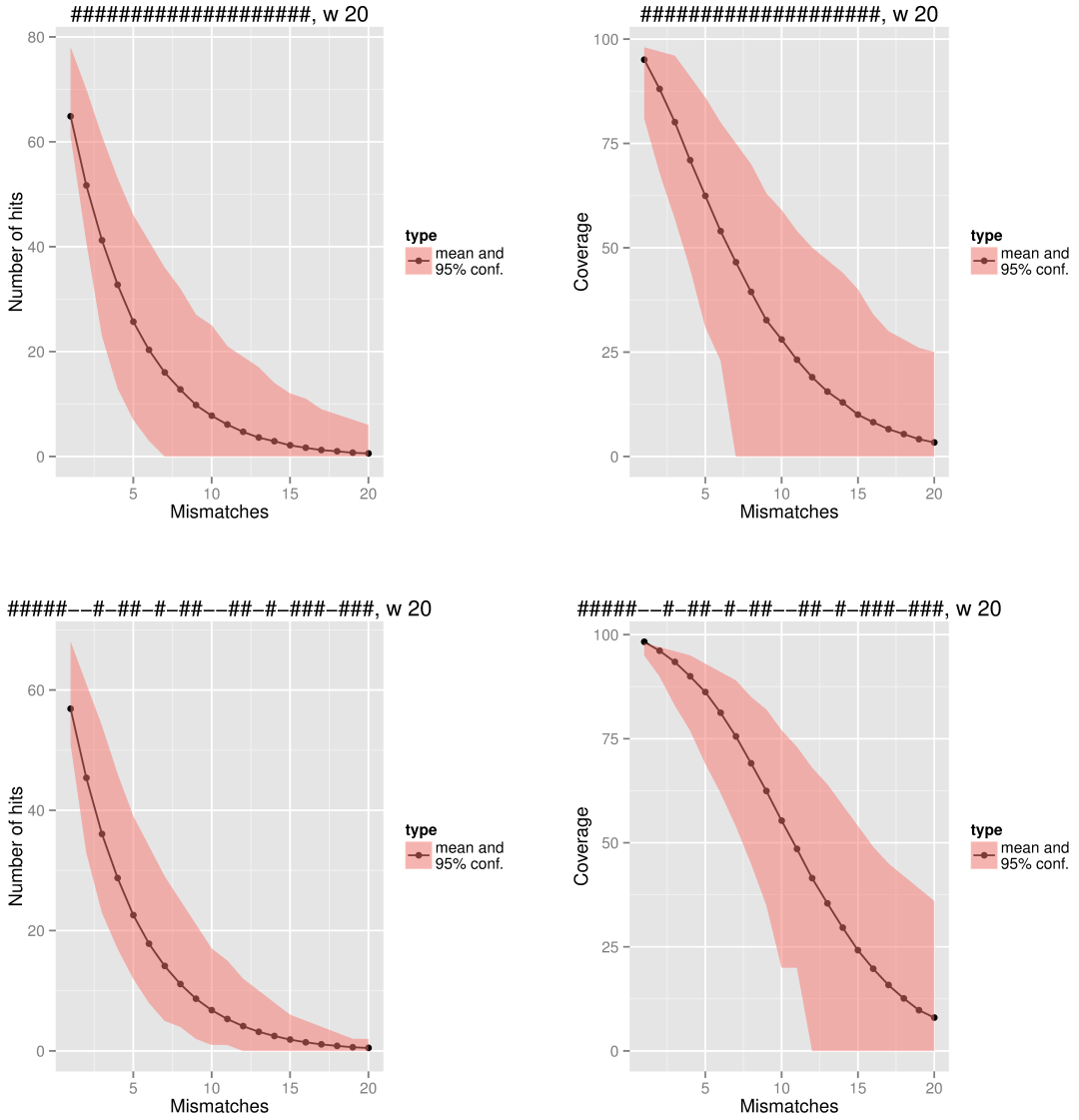


Figure 11.3: **Hit number and coverage vs. number of mismatches.** Same data as in Figure 11.2 shown with averages and 95% confidence intervals

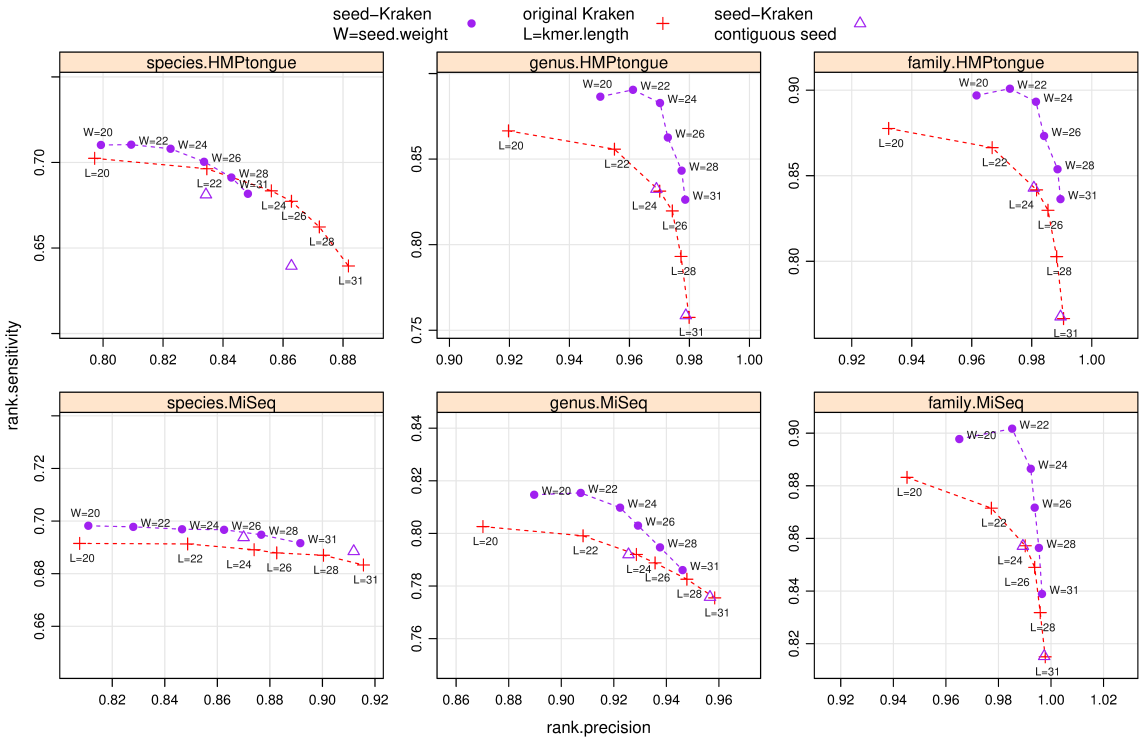


Figure 11.4: **Seed-Kraken vs. Kraken.** Sensitivity/precision of Seed-Kraken (circle points) and original Kraken (cross points) for HMPtongue and MiSeq datasets and three taxonomic levels: species, genus and family. Triangle points correspond to Seed-Kraken run on contiguous seed of weight 24 and 31, plotted to highlight the effect of the change in the assignment algorithm.

For each metagenomic dataset, we measured the sensitivity (percentage of correctly classified reads out of all reads) and precision (percentage of correct classifications out of all classifications) of Kraken and Seed-Kraken at three taxonomic levels: species, genus and family. In each case, this has been done with seeds of different weights between 20 and 31, and for each weight, Seed-Kraken has been run on a few different spaced seeds (see Conclusions).

Figure 11.4 shows sensitivity-precision ROC-curves (Receiver Operating Characteristic) for Seed-Kraken, and for the unmodified Kraken. In the case of Seed-Kraken, the "best performing" seed is charted for each weight. Furthermore, triangle points correspond to Seed-Kraken run on contiguous seeds, plotted in order to measure the effect of our modification in dealing with complementary sequences.

At the levels of genus and family, spaced seeds consistently show a better sensitivity-precision trade-off, with the sensitivity increase of about 2 percentage points for MiSeq and 3-5 points for HMPtongue, for a given precision rate. The results of Seed-Kraken with contiguous seeds (triangle points) confirm that this improvement is due to the use of spaced seeds and not to our slight modification of the assignment algorithm due to complementary sequences. For small weights (20-22), a spaced seed achieves simultaneously a better sensitivity and a better precision than the contiguous seed of the same weight. When the weight grows, the increment in precision disappears reaching the level of the contiguous counterpart, or sometimes coming down below it. However, this is largely compensated by the increase in sensitivity.

For the species level, the picture turns out to be more involved. Here we observe that due to the small modification of the assignment algorithm, Seed-Kraken run with contiguous seeds (triangle points) shows a modified behavior compared to Kraken. Specifically, we observe a drop in precision and a gain in sensitivity, and those are different for MiSeq and HMPtongue datasets. The reason for this is that Seed-Kraken makes more species-level classifications than Kraken but at the same time, makes more inaccurate assignments to a closely related organism (typically, different strain of the same bacteria), which eventually leads to a lower precision. This phenomenon has a bigger impact for rich databases (HMPtongue experiment) compared to "sparse" databases where each species is represented by few organisms (MiSeq). As for the contribution of spaced seeds, we observe an improved sensitivity-precision trade-off here as well. Compared to Seed-Kraken applied to contiguous seeds, this improvement is small for MiSeq but significant for HMPtongue, which shows a correction capacity of spaced seeds w.r.t. erroneous assignments to close strains. Compared to the original Kraken, we obtain a sensitivity increment of about 1% which becomes smaller (MiSeq) or completely disappears (HMPtongue) when the seed weight grows.

As mentioned earlier, the spaced seed corresponding to each plotted Seed-Kraken point has been selected out of a few (usually two to four) seeds tried. The full list of seeds applied in experiments and corresponding results can be found in the supplementary material. Here we just mention that for large weights (24 and more) the span of the seed becomes an important factor, with seeds of large span showing a drop in sensitivity and best seeds being those with relatively few jokers.

Building a Seed-Kraken limited database takes approximately 1 hour on a server with 20 CPU cores, and the resulting size is 26 GB for seed of weight 24, which compares to the 25 GB for original Kraken. The classification running times are longer than for original Kraken by a factor of 3 to 5.

11.4 Conclusions

Through a series of computational experiments, we showed that spaced seeds significantly improve the accuracy of metagenomic classification of short NGS reads. The superiority of spaced seeds for different variants of alignment-free sequence comparison has been recently demonstrated by other authors as well [466, 467, 464, 468]. In this work, we specifically focused on the metagenomics setting characterized by very large volumes of data, both in terms of the number of reads and the size of genomic database. This quantity of data precludes using some alignment-free comparison techniques, and leaves room only for highly time- and space-efficient approaches. Note also that in our setting, we have to compare short sequences (reads) with long ones (whole genomes), which makes an important difference with problems considered in [464, 468, 465]. For example, in the framework of metagenomic classification, it is hardly conceivable taking into account k-mer frequencies [464], as this information would be computationally difficult to utilize.

Another improvement considered in [464, 468, 465] is to use *multiple seeds*, i.e. several seeds simultaneously instead of a single one. This extension is known to bring an advantage in seed-and-extend sequence alignment [657, 476], and [464, 468] show that this improvement applies to alignment-free comparison as well. However, each seed requires to build a separate index for database genomic sequences, and therefore it appears computationally difficult to use multiple seeds in metagenomics, unless some new indexing techniques are designed for this purpose.

In our work, we studied three estimators: hit number, coverage and Jaccard index. Hit number and coverage behave similarly in classification (Section 11.3.1), but Jaccard index generally improves on them in the case of mapping to real genomes (Section 11.3.1), due to the correction w.r.t. the k-mer-richness. Considered as an estimator of alignment quality (Section 11.3.2, 11.3.3), coverage provides a certain advantage over hit number. More subtle estimators can be considered as well, e.g. by taking into account the position of k-mer in the read (reflecting the sequencing error rate), and this provides an issue for further study.

Designing efficient seeds for metagenomic classification is another important issue that goes beyond the present study. Note that optimal spaced seeds for seed-and-extend alignment are generally not optimal for alignment-free k-mer-based comparison [468]. In [468], the authors designed (sub-)optimal seeds maximizing the Pearson correlation between hit-number/coverage count and the alignment quality. Their solution is implemented in Iedera software[4] (http://bioinfo.lifl.fr/yass/iedera) [481]. On the other hand, recent work [492] introduces *quadratic residue seeds* (QR-seeds) for seed-and-extend alignment, which present a good performance and have the advantage of easy design, avoiding the computationally expensive enumeration of Iedera. In our work, we used both Iedera and QR-seeds adapted to our setting. We observed that in most cases, Iedera seeds are superior (being designed specifically for our task) but in a few cases, QR-seeds demonstrated equal or even better performance[5]. This may be due to their large span (cf. supplement material) for which applying Iedera is computationally costly.

We now summarize the main contributions of our work.

- We showed that spaced seeds can drastically improve the success rate of binary classification of alignments into two categories, each defined by a specific mismatch

[4] The latest version of Iedera performs design for Spearman correlation as well.
[5] E.g. best results of Table 11.4 for weights 14-18 were obtained with QR-seeds.

rate. Here the classification is done through "querying" an alignment using a seed as a mask and reporting whether the seed applies at a given position. For example, in discriminating between alignments of length 100 with mismatch rate 0.2 and 0.3, a spaced seed of weight 16 achieves 63% of success while a contiguous seed of the same weight achieves only 40% (Table 11.1c).

Recently, spaced seeds have been shown to define more efficient kernels for SVM classification of both protein [466] and nucleotide (RNA) sequences [468]. Compared to these works, here we demonstrate the superiority of spaced seeds in a very simple classification setting, where sequences have to be classified according to identity rate, without a training stage and without resorting to SVM machinery.

- We showed experimentally that spaced seeds allow for a better classification of NGS reads coming from a genome G between two other genomes G_1 and G_2 of the same genus. Here reads are classified according to the phylogenetic distance between G and G_1 and G and G_2 respectively. We established that in this task, Jaccard index provides an advantage over hit-number and coverage which is especially important for seeds of small weight.

- We studied how well different estimators (coverage/hit-number combined with spaced/-contiguous seed) correlate with the alignment quality, by measuring Spearman's rank correlation coefficient and mutual information coefficient. Here again, we observed a significantly better correlation produced by spaced seeds, but only when the alignment quality varies over a sufficiently large range, starting from identity rate around 0.9 or below. On the other hand, if only high-quality alignment are targeted (id rate at least 0.9) then the correlation produced by spaced seeds becomes lower, with hit-number measure over a *contiguous* seed eventually becoming the best for alignments of id rate about 0.95 or more.

- We also measured the correlation produced on real genomes within the metagenomic classification approach of [614, 458]. For this, we assumed that the "closeness" of a read to a genome is characterized by the quality of the alignment (in our case, the number of mismatches), and computed the correlation produced by different counts on simulated reads. These experiments confirmed the superiority of spaced seeds as well. Moreover, they showed that coverage combined with spaced seeds provides the best option, leading to the highest Spearman's correlation but also to a significantly higher Pearson's correlation. The latter means that this estimator induces a dependency closer to linear, which can be a useful feature for classification algorithms.

- Finally, we compared spaced and contiguous seeds through large-scale metagenomics experiments. We modified Kraken software [458] to make it work with spaced seeds, without modifying the core classification algorithm or any other parts of the software, with the only exception being the way the complementary sequences are dealt with. With just replacing contiguous seeds by spaced seeds, Kraken showed a consistent improvement of specificity/sensitivity trade-off at genus and family levels and a dataset-dependent improvement at the species level.

Real data experiments of Sections 11.3.1, 11.3.3 have been done using SnakeMake [542].

Overall, all our experiments corroborate the thesis of better performance of spaced seeds for metagenomic classification. Many further questions are raised by this work. Our

results remain to be explained with more rigorous probabilistic arguments, similarly to how it has been done for spaced seeds applied to seed-and-extend paradigm [658]. While there are obvious similarities between the two applications, the underlying "mechanisms" seem to be different. One sign of this difference is that optimal spaced seeds for the two problems are not the same, as mentioned earlier.

Experiments with Kraken (Section 11.3.4) give a strong evidence that spaced seeds can improve the classification accuracy in real-life large-scale metagenomic projects. One further improvement would be to implement coverage and Jaccard measures that showed, in general, a better performance compared to the hit number. Introducing spaced seeds rises new issues, such as the construction of an efficient index of the database, or adapting the algorithm of computing the most likely node of the taxonomic tree from counts produced by individual genomes, i.e. leaves of the tree. These questions are a subject for future work.

Chapter 12

ProPhyle: a metagenomic classifier based on Burrows-Wheeler transform

In this chapter, we present ProPhyle, a highly efficient method for k-mer-based metagenomic classification using BWT-index. We show that ProPhyle provides a more expressive index than Kraken [458], with a much smaller memory footprint. The resulting classification method supports multiple similarity measures and different modes of classification. All programs and scripts are available from http://github.com/karel-brinda/prophyle.

12.1 Introduction

Kraken [458], currently the most widely used tool for metagenomic classification, suffers from a series of problems, which often directly follow from its minimalistic algorithm design.

First, Kraken requires an extremely large amount of RAM (e.g., 120 GB for index construction of the default bacterial database). Even though Kraken's memory requirements may look moderate comparing to, e.g., LMAT (requiring 619 GB), they still exclude its usage on standard computers or even on modestly equipped clusters. The main bottleneck, the large memory required during the index construction, is a direct consequence of using k-mer counting and cannot be easily alleviated. The problem is deeply rooted in the approach of Kraken as it relies on LCAs, which has to be computed for all k-mers. This algorithmic task itself is hardly solvable in a small memory (k-mer's LCA is a "global" information that cannot be computed "locally" within the tree). Even with a precomputed index, Kraken still requires more than 75 GB for classification as every single k-mer occupies 12 bytes in the Kraken's hash table.

Second, Kraken uses an insufficiently expressive index. For each k-mer, it represents its LCA only, which appears to be too rough and may result in inaccurate classifications. For instance, Kraken often assigns reads to high taxonomic levels. Unfortunately, this effect is hardly visible from standard comparisons as they provide averaged statistics for entire metagenomes (e.g., [610]), rather than consider individual members of the metagenome. Therefore, it might be unclear how reliable Kraken can be on data from a user-provided metagenome with a particular composition. This drawback is directly inherited from the

simplified index and cannot be easily corrected in Kraken.

Here we present ProPhyle, a k-mer-based classification method using BWT-index with lossless property. ProPhyle avoids the presented drawbacks of Kraken and provides a highly efficient and accurate method, which is suitable to be run even on laptops. As we discuss later, many aspects of this method can be easily improved so the characteristics presented in this chapter are likely to be improved in near future. Before describing the method itself, we first provide a list of properties which we believe that a good metagenomic classifier should have.

12.2 General requirements on metagenomic classifiers

We observe that metagenomic classification strongly resembles read mapping. The latter is already a deeply studied problem with well-developed standards and methodologies. Here, we provide a list of properties which we consider highly desirable for metagenomic classifiers.

1. **Low memory requirements.** The classifier should run on a standard PC equipped with a reasonable amount of RAM. With the emergence of mobile sequencers (e.g., Oxford Nanopore – Minion), this requirement appears even more urgent. We strongly believe that a "point-of-care" taxonomic classification will become a very common task soon, e.g., in the contexts of disease surveillance, water contamination detection, or agriculture.

2. **Ease of usage.** The classifier should be provided as a single easily installable program that performs all steps of the classification process: downloading the taxonomy and the reference genomes, building the index, and performing the classification. It should also support user-specified taxonomies (e.g., user-provided phylogenetic tree).

3. **Support for standard bioinformatics formats.** The classifier should support standard bioinformatics formats. In particular, it should provide the output in SAM [125] and load taxonomies in Newick/NHX. Note that even though the initial purpose of SAM format was storing alignments, its flexibility enables us to use it for many other types of reads' annotations, including the assignments to a taxonomic tree.

4. **Automatic level detection vs. fixed taxonomic level.** The classifier should allow fixing the taxonomic level at which the reads would be assigned. It should also support its automatic detection (i.e., when a read is not specific enough, it is assigned on a node on some higher level).

5. **Support for assignment quality.** Similarly to *mapping quality* in read mapping, assignment quality should be provided with every taxonomic with assignment. In fact, two distinct quality measures can be considered in this case. For classification at a fixed level of taxonomic tree, a *horizontal assignment quality* would express the probability that the read should be assigned to another node on the same level. In case of classification with automatically detected level, a *vertical assignment quality* (expressing reliability of level selection) should be computed as well.

6. **Best hit vs. all hits.** If possible, the classifier should be capable to work in two different modes. The *best-hit mode* should quickly retrieve the node maximizing the underlying measure. The *all-hits mode* should report all hits whose score is close

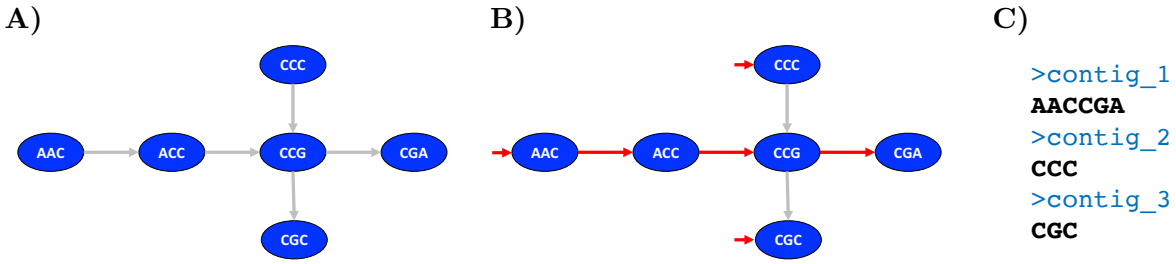


Figure 12.1: **Assembly procedure in ProPhyle.** Contigs are greedily assembled from a set of k-mers. **A)** From a given set of k-mers, first a vertex-centric de-Bruijn graph is constructed. **B)** Until the graph is empty, new contigs are assembled. Iteratively, a random vertex is selected as a seed of the new contig and then greedily extended to both directions. Whenever a node is used, it is immediately removed from the graph. **C)** Each obtained contig is stored as a separate sequence with the same FASTA file.

enough to the best node. While the best-hit computation is faster, the all-hits mode can help to improve the accuracy of quantitative methods working with provided assignments.

7. **Updating prior probabilities.** As an option, the prior probabilities of assignments to individual nodes should be automatically continuously adjusted according to assignments computed so far.

12.3 Design of ProPhyle

In addition to the requirements of the previous section, we impose two more properties that Prophyle should satisfy. First, the index should be lossless, i.e., it should be capable to retrieve an *exhaustive list* of genomes that the queried k-mer occurs in. Then, we want to support several distinct measures including the Jaccard index and the coverage criterion (see the Chapter 11).

Even though we originally intended to use spaced seeds since we have shown that they significantly improve the classification accuracy, we did not find any space efficient index structure for spaced k-mers. So we had to eventually relax on this idea in ProPhyle and the final method works with contiguous k-mers only.

12.3.1 Key algorithmic ideas

Representing a collection of k-mer sets using a BWT-index. Designing a k-mer index for a metagenomic classifier (Figure 10.2) often corresponds to the task of creating a data structure capable to store and retrieve a list of nodes for every k-mer from the reference database, thus, a specific variant of an associative array. For instance, Kraken index stores the lowest common ancestors of each k-mer in the database, so such a node list contains exactly one node for every k-mer.

The most straightforward strategy to represent such associative arrays is based on using hash-tables, but this leads to huge memory footprints (e.g., Kraken uses 12 bytes per k-mer). In order to solve this obstacle, we suggest to replace hash-tables by BWT-indexes [193], similarly to what was done for read mappers in 2009 [181, 182, 226].

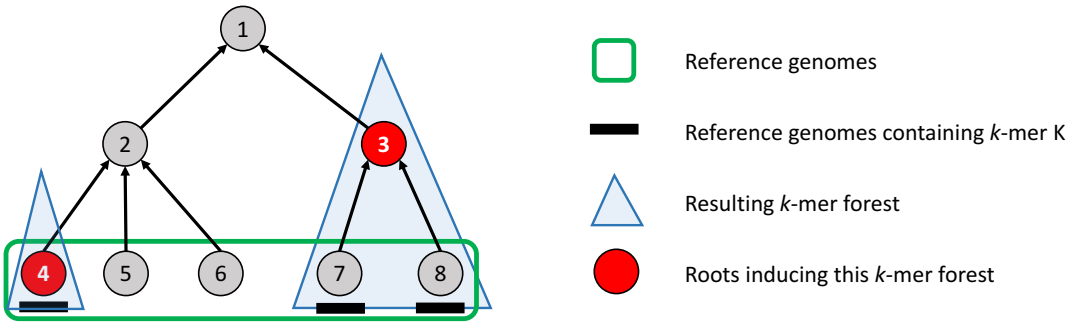


Figure 12.2: **Example of a k-mer forest.** A k-mer K occurs in reference genomes associated with leaves 4, 7, and 8. The subtrees rooted by nodes 3 and 4 form the minimal set of subtrees containing K in all leaves, hence the forest highlighted in blue is the k-mer forest of K. In Prophyle, we represent such a forest using a list of roots of the contained trees/ We can then retrieve all genomes containing K (nodes 4, 7, 8 in this example). Since LCA of nodes 4, 7 and 8 is the root in this example (node 1), genomes at nodes 5 and 6 would be false positives in Kraken-like approaches.

Let us briefly describe how BWT-indexes can help in our situation. To summarize, we have a tree and sets of k-mers in its nodes (in the leaves and also in the internal nodes). The exact strategy of distribution of k-mers to the nodes of the tree is specific to each individual classifier, being strongly related to the classifier's read assignment algorithm. Nevertheless, regardless of the exact distribution strategy, we observe that individual nodes often contain overlapping k-mers. Roughly speaking, this a consequence of the fact that overlapping k-mers are often specific to the same genomes or the same taxonomic clade (with the exception of low-complexity regions). We strongly exploit this property in our indexing structure.

We use the following scheme. First, at every individual node, we assemble contigs from all its associated k-mers. The assembly procedure is performed as a greedy enumeration of disjoint paths in a vertex-centric de-Bruijn graph built from the k-mers assigned to the node. See Figure 12.1 for an illustrative example. The resulting contigs have the property that they contain a k-mer K as a substring if and only if K was a member of the original k-mer set at that node. When the assembly is finished for all nodes, we concatenate all the obtained contigs from all the nodes. We continue by creating a BWT-index [193] for the resulting long string. In addition to the bare BWT-index, we have also to keep track of the positions of contig borders and of the starting positions of segments corresponding to the individual nodes. Note that a similar approach has been used for de-Bruijn graph representation [659, 660].

Querying a k-mer starts by the standard BWT-index search procedure followed by translating the obtained locations to nodes. Then for every obtained node, we have to verify whether the localized k-mer is situated outside contig borders, otherwise the node would be a false positive. Finally, we report the node as a true match.

The presented indexing structure is lossless in that sense that it retrieves a precise list of nodes for every k-mer. Note that this approach is not limited to trees, it can be used basically for arbitrary systems of k-mer sets. As we will show later on real experiments, the resulting footprint of this index is low provided that many k-mers are overlapping

(within the individual k-mers sets to be indexed).

k-mer forests and their construction using k-mer propagation. As it was mentioned above, different classifiers associate k-mers with nodes of the taxonomic tree differently, based on the model used for classification and the heuristic for score computation. To achieve a good accuracy, we want to distribute k-mers in such a way that we could fully reconstruct k-mer sets of individual genomes (*lossless property*). Note that this is impossible with the LCA representation in Kraken. If we leave all k-mers in the leaves, the more frequent k-mers would occupy too much space in the resulting index and also classification would be slower as assignments on higher levels would have to be slowly computed from the level of leaves.

We introduce a representation called a *k-mer forest* that solves our problem. Within this concept, we associate each k-mer K with a minimal set of subtrees of the taxonomic tree such that the leaves of these subtrees are the genomes containing K and every leaf of each of these subtrees contains K. See Figure 12.2 for an example.

In addition to the lossless property, a major advantage of this representation is the low memory requirements for its construction. Using an algorithm called *k-mer propagation*, the k-mer forest can be computed in a series of local modifications of a tree with k-mer sets in its nodes. Whereas Kraken has to store all k-mers in memory to compute their lowest common ancestors, we need to store in memory at one moment only a small number of k-mers specific to a local small subtree.

The k-mer propagation is a bottom up approach. We start with a tree, which has k-mers in its leaves (they contain k-mers of individual genomes). Then we propagate the k-mers from leaves to higher levels. A node can be processed when all of its children have already been processed. Processing a single node proceeds by computing the intersection of children k-mers sets, subtracting the intersection from all these sets, and assigning the intersection at to the node.

12.3.2 ProPhyle index

Overview. ProPhyle uses an indexing structure based on the ideas of Section 12.3.1. First, we propagate k-mers from individual genomes bottom-up along the taxonomic tree, assemble contigs, and merge them into a single reference FASTA file. Then we build a BWT-index from this reference such that querying a k-mer proceeds by the full-text using the index and the retrieved sequence names are translated into nodes. The entire process is illustrated in Figure 12.3.

Assembly and k-mer propagation. Assembly and k-mer propagation are performed using a C++ program called `prophyle-assembler`, which is used recursively on all nodes in the tree. The program loads sequences from several FASTA files, extracts k-mer sets, computes their intersection, assembles it into contigs (see below), and finally removes, using re-assembling, the intersecting k-mers from the children FASTA files (see Figure 12.4).

We parallelize the entire k-mer propagation process using GNU Make (with the `-j` parameter). Dependencies in the input Makefile (automatically generated) mirror the taxonomic tree so the entire task is highly parallelizable. `prophyle-assembler` is first called on the lowest nodes and continues to the root. Most of the computational time is spent on lowest nodes as a majority of k-mers do not propagate at all, or by a single level only.

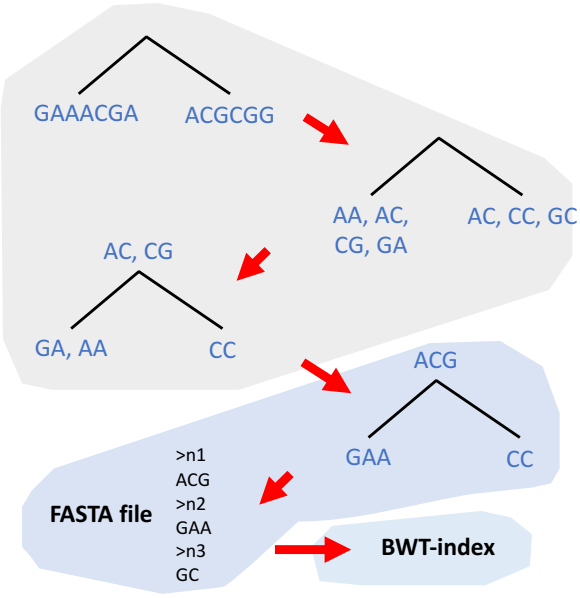


Figure 12.3: **Schematic overview of k-mer propagation and index FASTA construction in ProPhyle.** First, k-mers are propagated bottom-up and assembled into contigs. Then contigs from all nodes are placed into a single FASTA file, which is finally used to construct a BWT-index.

BWA BWT-index. For the full-text search, we used the BWA implementation [181] of BWT-index [193] and adapted it for our purposes. We selected BWA because it is a tool with a comprehensible high-quality code, which is well-debugged, highly optimized (being one of the fastest implementation of BWT-index), well-documented, and at the same time having a reasonable memory footprint.

The construction of BWT-index is performed by BWA directly, whereas querying is implemented in `prophyle-index`, a dedicated tool. It reduces the BWA memory memory footprint by loading only certain parts of the index. By calling low-level BWA functions only, `prophyle-index` can quickly retrieve all nodes containing a contig containing a queried k-mer. The matching procedure can work in two different modes.

In the *restarted search* mode, querying every new k-mer results in a sequence of k operations on the BWT-index (standard search with BWT-index [193]). In addition to BWA index files, no other information needs to be precomputed, but the matching is relatively slow.

In the *rolling window* approach, we exploit the fact that a query of a k-mer can be computed from the query of the previous k-mer, since they have an overlap of $k-1$ letters. First we extend the suffix array interval using a so-called k-LCP bit-array (defined as $k\text{-LCP}[i] = 1 \iff \text{LCP}[i] \geq k-1$, where LCP denotes longest common prefix array), which corresponds to removing a single character of the searched pattern, followed by adding a new character to the query using the standard BWT-index technique. The rolling window is approximately $5\times$ faster compared to the restarted search in our experiments, but the k-LCP bit-array must be computed in advance and loaded into memory before matching. More detailed information about the rolling window approach will appear soon [661].

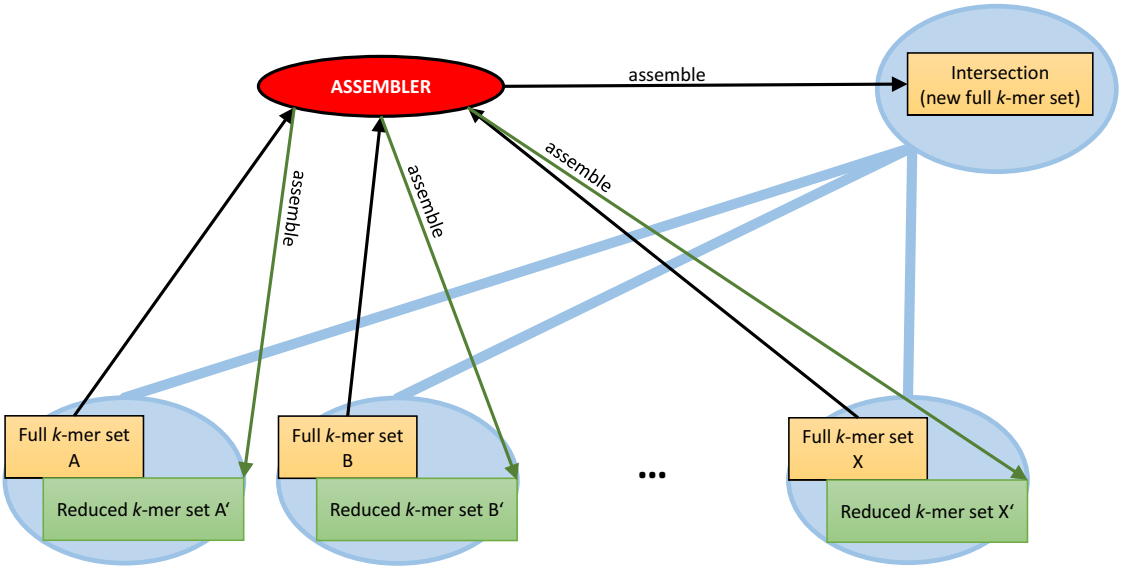
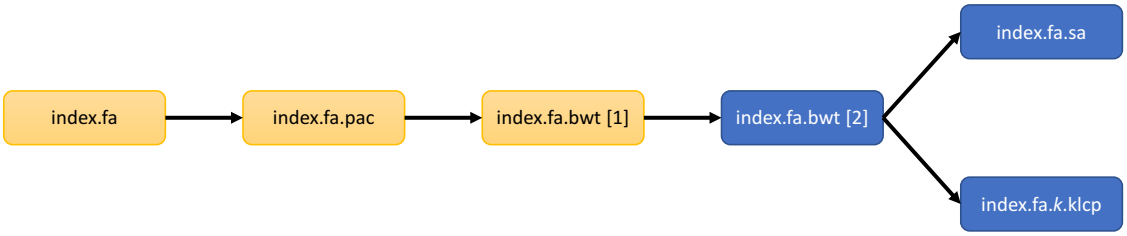


Figure 12.4: **Scheme of k-mer propagation at a single node.** First, *full k-mer sets* are loaded from all children nodes. Then, intersection of all these sets is computed and saved as a *full k-mer set* at the current node. Finally, the obtained intersection is subtracted from *full k-mer sets* of the children nodes so that *reduced k-mer sets* are obtained and saved on disk. This scheme is recursivelly applied to all nodes of the taxonomic tree so k-mers shared between all nodes are propagated bottom-up. For the leaves, *full k-mer sets* are specified by the FASTA files of genomes. All newly created k-mers sets are stored in FASTA files as greedily assembled contigs containing only these k-mers (see Figure 12.1).



File	**Command**	**File description**	**Size** (bits per bp)
index.fa		contigs from k-mer propagation	8+
index.fa.pac	`bwa fa2pac`	packed merged contigs with rev. compl.	4
index.fa.bwt [1]	`bwa pac2bwtgen`	BWT string	4
index.fa.bwt [2]	`bwa bwtupdate`	BWT string and sampled OCC array	8
index.fa.k.klcp	`prophyle-index index`	k-LCP array and its sampling	4
index.fa.sa	`bwa bwt2sa`	sampled suffix array	8

Figure 12.5: **ProPhyle BWT-index construction process and the resulting files.** Yellow and blue nodes correspond to auxiliary files and final index files, respectively. The input index.fa file is obtained from the k-mer propagation step. First, we convert it to `index.fa.pac`, a file containing all contigs merged together, merged with a reverse complement of this long string. This resulting string is encoded using 2 bits per nucleotide encoding. In the next step, we compute Burrows-Wheeler of this string and store it as `index.fa.bwt`. This file is then extending by appending the sampled OCC array to its end. Finally, k-LCP array and sampled suffix array are computed (possibly in parallel using `prophyle-index index`).

Figure 12.5 describes the individual steps of ProPhyle BWT-index construction including the exact commands.

We end this section by summarizing main steps of index construction.

1. Assign k-mers of individual reference sequences to leaves of the taxonomic tree
2. Propagate k-mers
3. Construct the main FASTA file
4. Construct the BWT-index
5. Construct the k-LCP array

12.3.3 ProPhyle classification algorithm

We have developed the first prototype of the ProPhyle assignment algorithm. For every read, based on assignments of its individual k-mers provided by the index, it decides on the assignment to a node of the taxonomic tree, which are then reported in the SAM format [125]. The classifier currently supports two measures, the hit count like Kraken [458] and the coverage criterion like CoMeta [626]. It can work both in the best-hit and the all-hits modes. The program can also emulate the Kraken assignment algorithm.

12.4 Results

We built a ProPhyle index comprising 2,787 bacterial genomes from the RefSeq database [662] for k-mers of length 31, and we compared it to the Kraken index constructed from the same database of genomes (Table 12.1, Figure 12.6). The memory footprint of ProPhyle was measured using GNU Time. We used a computer with 24 cores and 64 GB RAM. Since Kraken's requirements exceed computational resources available in our laboratory, we could not compare both programs directly. Therefore, we present values from Kraken website though we are aware of the fact that they were obtained on a much stronger computer.

We observe that memory requirements of ProPhyle are very moderate compared to Kraken. The decrease from 120 GB to 13 GB of the index construction footprint corresponds to 9× memory reduction. The observed difference is mainly a consequence of the fact that k-mer propagation is local.

The querying footprint of 14.2 GB results from memory allocation for BWT, sampled SA and k-LCP (in sum 13.2 GB), and from small caches (which we have, however, managed to partially reduce). With the current implementation, we compress more than 5× compared to Kraken, which uses approximately 75 GB.

We see that ProPhyle memory requirements allow the metagenomic classification to be performed on standard PC or even laptops, which was the main goal of our work. Moreover, the ProPhyle index is much more expressive than the index of Kraken.

The ProPhyle index construction takes more than 4 hours, mainly due to the slow BWT construction. The first step, k-mer propagation, is well-parallelized and takes only 20 minutes. The other steps currently use non-parallelized algorithms, nevertheless, their parallelization is feasible (see Chapter 13) and we expect that the index construction time could be decreased to approximately 2 hours, which would be fully comparable to Kraken (1.5 hours on a high-performance computer).

Query speed of ProPhyle is measured in terms of *reads per minute* (RPM), the time required for loading the index into memory is excluded from this statistics (similarly to the methodology from [458]). We observe that the rolling window with 681,000 RPM provides more than 5× speed-up over restarted search with approximately 120,000 RPM. Compared to Kraken, the rolling window is still approximately 2× slower. Nevertheless, this slowdown is compensated by low memory requirements and a higher expressiveness of the index.

It is also instructive to look at characteristics of the obtained contigs. Even though their number is relatively high (24 million), they are still relatively long (331 bp in average). The total number of represented k-mers (approximately 7 billions) corresponds to the lower bound on the total length of contigs. Since the current total length is approximately 8 Gbp, we could not diminish index size using a better assembly algorithm by more than 10%. Therefore, the greedy assembly is a successful strategy for this problem.

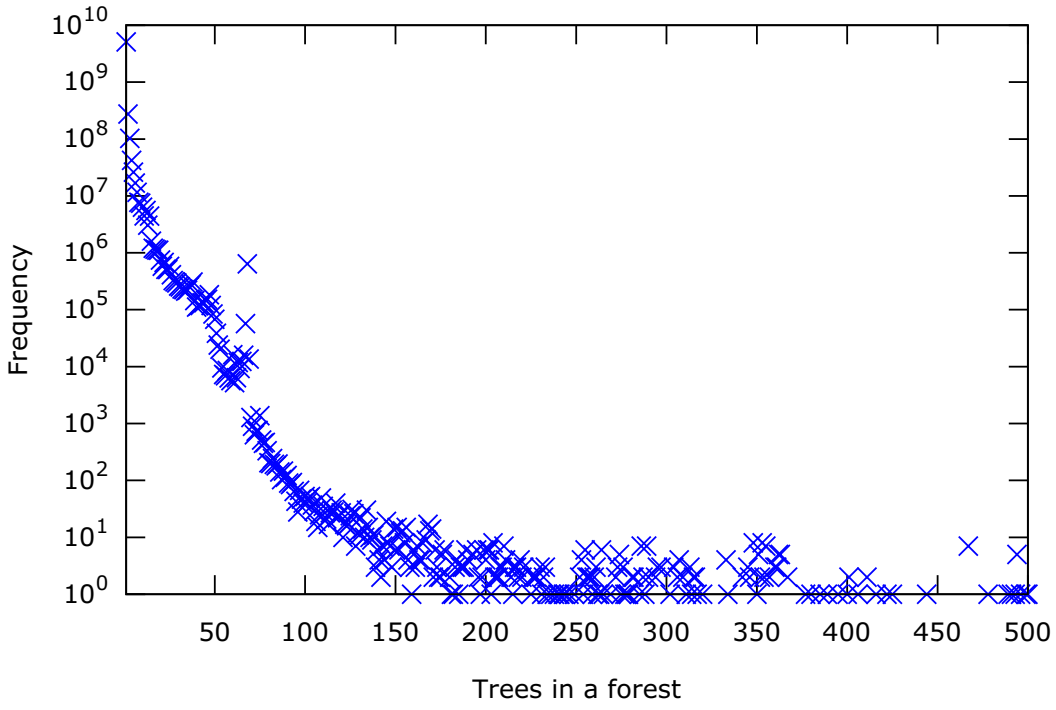


Figure 12.6: **Histogram of k-mer forests in the resulting ProPhyle index.** The index was built from the RefSeq database [662] comprising 2,787 bacterial genomes for k-mers of length 31. The index contains $5,621,116,493$ distinct canonical k-mers. A vast majority of them ($5,091,088,664$; 91%) have a k-mer forest of size 1; $275,641,073$ k-mers a forest of size 2; $102,645,143$ k-mers a forest of size 3; etc.

A) Index construction performance

Step	Elapsed time	Memory peak	CPU usage
ProPhyle index construction	4 h 28 m	13.0 GB	N/A
FASTA construction	27 m	10.2 GB	N/A
k-mer propagation	20 m	10.2 GB	1609%
FASTA merging	7 m	0.1 GB	81%
BWT & OCC construction	2 h 20 m	11.2 GB	N/A
fa2pac	02 m	6.1 GB	60%
pac2bwtgen	2 h 16 m	4.4 GB	100%
bwtupdate	02 m	11.2 GB	53%
k-LCP & SA construction	1 h 41 m	13.0 GB	136%
Kraken index construction♢	≈1 h 32 m	≈120.0 GB	N/A

B) Index querying performance

Method	Index performance	Memory peak
ProPhyle – rolling window	681,031 RPM	14.2 GB
ProPhyle – restarted search	122,650 RPM	12.4 GB
Kraken♢◆	≈1,300,000 RPM	≈75.0 GB

C) Index files

File	Description	Size
bwt	BWT string + OCC array	7.5 GB
klcp	k-LCP bitvector	1.9 GB
sa	sampled suffix array	3.8 GB
ann	textual list of contigs	1.0 GB

D) Contig statistics

Category	Value
Total length of all contigs	7,993,453,766 bp
Average contig length	331 bp
Median contig length	70 bp
Number of contigs	24,170,781
Number of represented k-mers	7,268,330,336
Number of distinct k-mers	5,621,116,493

♢ Values are taken from https://ccb.jhu.edu/software/kraken/MANUAL.html. Measured on a different computer with higher performance, and with a slightly different version of the same bacterial database.

◆ Kraken's RPM comprises also read assignment.

Table 12.1: **Characteristics of the resulting ProPhyle index and comparison to Kraken.** The index was built from the RefSeq database [662] comprising 2,787 bacterial genomes for k-mers of length 31. The computation was performed on a machine with Intel® Xeon® CPU E5-2630 v2 @ 2.60GHz, 24-cores, and 64 GB RAM.

Chapter 13

Discussion

Spaced seeds. We demonstrated that spaced seeds provide better estimators of alignment score than contiguous k-mers and that they can strongly improve the metagenomic classification. The main obstacle preventing the usage of spaced seeds in practice is the lack of compact indexes for spaced k-mers and hash tables are too memory consuming in many scenarios. Even though some recent works [663] have shown a promising direction, no good implementation exists so far.

Classification with ProPhyle. We introduced the ProPhyle classifier and showed that it uses 9× less memory for index construction than Kraken, and approximately 5× less memory for classification. A lower speed of classification is a direct consequence of a better compression of the index, and of the fact that the resulting k-mer index is, unlike Kraken, lossless, i.e., it can retrieve all genomes in which a k-mer occurs.

The current version of ProPhyle is still experimental and, in particular, the component responsible for read assignments is still a prototype, awaiting the upcoming C++ reimplementation Nevertheless, we are confident that the resulting classification method will be more accurate than Kraken as we avoid the "LCA distortion".

Optimizing ProPhyle. The current ProPhyle methods can be refined in many different directions. Query speed, memory footprint, and sensitivity are characteristics in a trade-off. Index construction is out of this trade-off and it can be strongly improved due to the lack of parallelization in most of index construction steps.

1. **Speed of index construction.**

   - **Parallel index construction.** The most time-demanding step of index construction is currently the computation of BWT (see Table 12.1). Even though methods for its parallel construction exist [528, 664, 371], they are not planned to be incorporated in BWA [665]. Similarly, sampled suffix array and k-LCP array construction could be parallelized [661, 665].

2. **Index size.**

   - **Binarized tree.** During the k-mer propagation step, the tree can be binarized in order to improve compression (more k-mers could then be propagated up). "prophyle-assembler" could run a hierarchical clustering based on numbers of

shared k-mer between children nodes and output a new topology of the considered subtree. A similar method is used in Centrifuge [622]. We suppose that the resulting index would be smaller, but more fragmented and therefore slightly slower.

- **Sampled node-id array.** In the standard BWT-index search procedure, strings are located in a suffix array. Then it is necessary to translate these suffix array coordinates to standard coordinates, which is done using a sampled suffix array.

  As we are using standard BWT-index in the heart of our method, we are internally retrieving exact coordinates of k-mers in the reference (i.e., in the sequence of merged contigs). Nevertheless, this information is used only for determining the node id and checking whether the k-mer does not span the border of contigs (in such a case, it is a false positive). In fact, we could relax on precise coordinates and replace the sampled suffix array by sampled node-id array, which could reduce the ProPhyle memory footprint. However, a space efficient method for excluding k-mers on borders still would need to be designed.
- **Compression of BWT.** BWA uses an uncompressed representation of BWT strings. Its compression (implemented, e.g., in SGA [666]) could lead to better memory footprints.

3. **Sensitivity.**

   - **Approximate k-mer matching.** Currently, k-mers are required to have exact matches to the reference sequence that are accounted in hit number or other measures. We could relax on this requirement and introduce a certain error tolerance (similarly to SMART [627]). For instance, if a k-mer is not found in the index but the previous was, we can correct (replace) its last letter by a nucleotide that would make the new k-mer exist in the database. This strategy might improve accuracy and sensitivity for reads with sequencing errors or mutations.

Part IV

Conclusions

Conclusions

In this thesis, we studied methods for mapping and classifying Next-Generation Sequencing data. Let us summarize the main presented results.

We provided the first comprehensive study of dynamic read mapping (Part II). We also developed the first online consensus caller called Ococo (Chapter 7) and a simulator of dynamic mapping (Chapter 8). Using this simulator, we performed a comparative analysis of dynamic mapping, standard static mapping and iterative referencing (Chapter 8).

We studied methods for metagenomic classification (Part III) and provided the first comprehensive study about spaced seeds for this task (Chapter 11). Moreover, we plugged spaced seeds into Kraken, the most popular metagenomic classifier nowadays (Chapter 11). We developed ProPhyle, a metagenomic classifier based on a novel usage of BWT-index for k-mer indexing (Chapter 12).

Dynamic read mapping. We studied the problem of dynamic read mapping, i.e., mapping with continuously updated reference sequence on the basis of previously computed read alignments. To compare dynamic mapping to other mapping approaches (static mapping and iterative referencing), we developed the RNFtools toolkit (Chapter 6). Then we provided Dynamic Mapping Simulator (Chapter 8), a pipeline for comparative evaluation of three mapping methods: static mapping, dynamic mapping (with or without remapping), and iterative referencing. Using Dynamic Mapping Simulator and RNFtools, we showed that the dynamic mapping approach can notably improve read alignment and variant calling (Chapter 8).

We also developed Ococo (Chapter 7), the first online consensus caller since such a component must be present in every dynamic mapper. The program calls the consensus, using compact statistics based on approximate counting, directly from a stream of unsorted alignments. Beyond its application to dynamic mapping, Ococo can be employed as an online SNP caller in various analysis pipelines, enabling calling SNPs from a stream without saving the alignments on disk. In particular, this approach can be useful in combination with the Oxford Nanopore real-time sequencing.

Metagenomic classification. In Chapter 11, we provided a thorough study of spaced seeds for metagenomic classification. The presented results explain how the hit count and the coverage combined with contiguous and spaced k-mers can influence the sensitivity and the accuracy of the resulting classification. The central idea is that classification quality depends on how well we can estimate the alignment score. We show that better estimates are obtained with spaced k-mers rather than contiguous k-mers.

The methods that metagenomic classifiers use, usually correspond to assigning a read to the node maximizing one of these measures (possibly normalized by the number of the k-mers at the node). The k-mer sets for leaves are usually computed directly from the

provided genomes. Since the genomes at internal nodes (corresponding to the evolution ancestors) typically cannot be provided, their composition usually has to be deduced from the genomes at the leaves. Nevertheless, the information about k-mers at individual nodes is usually distorted due to technical limitations of the indexes.

For instance, Kraken classifies reads based on the hit count. With regard to Kraken indexing scheme and its assignment algorithm, the information about k-mer sets is distorted in the following way. Each k-mer is moved to its LCA and subsequently redistributed back to the all nodes descending this LCA. In consequence, the Kraken method can work well only for genomes and clades very little influenced by this k-mer redistribution. Indeed, reads often tend to be assigned at high taxonomic levels. Unfortunately the Kraken paper does not provide much insight into this problem so it remains unclear for what type of data the Kraken assignments are reliable.

We believe that ProPhyle, which we present in Chapter 12, can alleviate these problems as it retrieves exact information about the k-mer occurrences. Moreover, the employed indexing scheme, based on a BWT-index, strongly reduces the associated memory footprint, and allows to construct the index and to classify the reads on laptops. Metagenomic classification on laptops has become even more important nowadays, in the era of mobile sequencers.

Part V

Appendices

Contents - Part V

Appendix A

Languages of lossless spaced seeds

A.1 Introduction

The annual volume of data produced by the Next-Generation Sequencing technologies has been rapidly increasing; even faster than growth of disk storage capacities. Thus, new efficient algorithms and data-structures for processing, compressing and storing these data, are needed.

Similarity search represents the most frequent operation in bioinformatics. In huge DNA databases, a two-phase scheme is the most widely used approach to find all occurrences of a given string up to some Hamming or Levenshtein distance. First of all, most of dissimilar regions are discarded in a fast *filtration phase*. Then, in a *verification phase*, only "hot candidates" on similarity are processed by classical time-consuming algorithms like Smith-Waterman [130] or Needleman-Wunsch [129].

Algorithms for the filtration phase are often based on so-called *seed filters* which make use of the fact that two strings of the same length m being in Hamming distance k must necessarily share some exact patterns. We can represent these patterns as strings over the alphabet $\{\texttt{\#}, \texttt{-}\}$ called *seeds*, where the "matching" symbol `#` corresponds to a matching position and the "joker" symbol `-` to a matching or a mismatching position.

For example, for two strings of length 15, matching within two errors (so-called $(15, 2)$-problem), the shared patterns are `##-#--##-#` or `#####`. For illustration, if we consider that two strings match as `===X=====X=====` (where the symbols `=` and `X` represent respectively matching and a mismatching positions), then the corresponding seed positions can be following:

```
===X=====X=====
.##-#--##-#....
....#####......
```

As the second seed is the longest possible contiguous seed in this case, we easily observe the advantage of spaced seeds in comparison to contiguous seeds; for the same task, we can find spaced seeds with higher number of `#`'s (so-called *weight*).

Two basic characteristics of seeds are *selectivity* and *sensitivity*. Selectivity measures restrictivity of the filter created from the seed. In general, higher weight implies better selectivity of the filter. *Lossless* seeds are those seeds having full sensitivity. They are easier to handle mathematically on one hand, but attain lower weight on the other hand. Therefore, *lossy* seeds are more suitable for practical purposes since a small decrease in sensitivity can be compensated by considerable improvement of selectivity.

Literature. The idea of lossless seeds was originally introduced by Burkhardt and Karkkäinen [462]. Let us remark that lossy spaced seed appeared in the same time in the PatterHunter program [463]. Possible generalization of lossless seeds was studied by Kucherov et al. [486]. The authors studied seed families and the case when more hits of a given seed are required (the pattern is shared at more positions). They proved that for a fixed number k of errors, *optimal seeds* (i.e., seeds with highest possible weight among all seeds solving the problem) must asymptotically satisfy $m - w(m) \in \Theta(m^{\frac{k}{k+1}})$, where $w(m)$ denotes the maximal possible weight of a seeding solving the (m, k)-problem. They also started a systematic study of seeds created by repeating of short patterns. Afterwards, the results on asymptotic properties of optimal seeds were generalized by Farach-Colton et al. [488]. Computational complexity of optimal seed construction was derived by Nicolas and Rivals [483].

Further, the theory on lossless seed was significantly developed by Egidi and Manzini. First, they studied seeds designed from mathematical objects called perfect rulers [489, 490]. The idea of utilization of some type of "rulers" was later independently extended by KB [494] (cyclic rulers) and Edigi and Manzini [493] (difference sets). In [493], these ideas were extended also to seed families. Let us mention that cyclic rulers and difference sets mathematically correspond to each other. Edigi and Manzini [491] also showed possible usage of number-theoretical results on quadratic residues for seed design.

In practice, seeds often find their use in short-read mapping in mappers based on the seed-and-extend paradigm (for more details on read mapping, see for example [200]). ZOOM [282] and PerM [274] are examples of such mappers, which utilize lossless seeds.

A list of papers on spaced seed is regularly maintained by Noé [474].

Our object of study. One of the most important theoretical aspects of lossless seeds are their structural properties. Whereas good lossy seeds usually show irregularity, one can observe that good lossless seeds are often repetitions of short patterns ([486, 274, 494, 493]). The question whether optimal seeds can be constructed in all cases by repeating patterns (being short with respect to seed length) remains open (see [494, Conjecture 1]). Its answering could have practical impacts on bioinformatical software tools development since the search space of programs for lossless seeds design could be significantly cut and also indexes in programs using lossless seeds for approximate string matching could be more memory efficient ([274]).

Results. In this paper, we follow and further develop ideas from [494]. First we transform the problem of seed detection into another criterion (Theorem 1). Then we prove that the sets of seeds obtained after fixing the parameters:

- the number of allowed errors k;
- the seed margin ℓ, which is the difference between the size m of compared strings and the seed length s;

coincide with languages of some sofic subshifts. Therefore, those sets of seeds are recognized by finite automata. We also show how these sofic subshifts can be decomposed into subshifts of finite type. These results provide a new view on lossless seeds and explain their periodic properties.

A.2 Preliminaries

Combinatorics on words. An *alphabet* $\mathcal{A} = \{a_0, \dots, a_{m-1}\}$ is a finite set of symbols called *letters*. A finite sequence of letters from $\mathcal{A}$ is called a *finite word* (over $\mathcal{A}$). The set $\mathcal{A}^*$ of all finite words (including the empty word ε) provided with the operation of concatenation is a free monoid. The concatenation is denoted multiplicatively. If $w = w_0 w_1 \cdots w_{n-1}$ is a finite word over $\mathcal{A}$, we denote its length by $|w| = n$ and use the symbol $|w|_a$ for the number of occurrences of the letter $a \in \mathcal{A}$ in w. We deal also with bi-infinite sequences of letters from $\mathcal{A}$ called *bi-infinite words* $\mathbf{w} = \cdots \mathbf{w}_{-2}\mathbf{w}_{-1}|\mathbf{w}_0\mathbf{w}_1\mathbf{w}_2 \cdots$ over $\mathcal{A}$. The sets of all bi-infinite words over $\mathcal{A}$ is denoted by $\mathcal{A}^{\mathbb{Z}}$.

A finite word w is called a *factor* of a word $\mathbf{u}$ ($\mathbf{u}$ being finite or bi-infinite) if there exist words p and s (finite or one-side infinite) such that $\mathbf{u} = pws$. We say that the word w is a prefix of $\mathbf{u}$ if $p = \varepsilon$, and a suffix of $\mathbf{u}$ if $s = \varepsilon$. For given indexes i and j, the symbol $\mathbf{u}[i, j]$ denotes the factor $\mathbf{u}_i\mathbf{u}_{i+1} \cdots \mathbf{u}_j$ if $i \leq j$, or ε if $i > j$. A concatenation of k words w is denoted by w^k. The set of all *factors* of a word $\mathbf{u}$ ($\mathbf{u}$ being finite or bi-infinite) is called the language of $\mathbf{u}$ and denoted by $\mathcal{L}(\mathbf{u})$. Its subset $\mathcal{L}(\mathbf{u}) \cap \mathcal{A}^n$ containing all factors of $\mathbf{u}$ of length n is denoted by $\mathcal{L}_n(\mathbf{u})$.

Let us remark that this notation will be used extensively in the whole text. For instance $\mathbf{w}[2, 5]^{-4}$ denotes the word created by concatenation of the factor $\mathbf{w}_2\mathbf{w}_3\mathbf{w}_4\mathbf{w}_5$ of a bi-infinite word $\mathbf{w}$ and the word `----`. Similarly, for a finite word v of length n, by $\cdots\texttt{--}|v\texttt{--}\cdots$ we denote the bi-infinite word $\mathbf{u}$ such that for all $i \in \{0, \dots, n-1\}(\mathbf{u}_i = w_i)$ and for all $i \in \mathbb{Z} \setminus \{0, \dots, n-1\}(\mathbf{u}_i = \texttt{-})$. For more information about combinatorics on words, we can refer to Lothaire I [667].

Symbolic dynamics. Consider an alphabet $\mathcal{A}$. We define a *shift* operation σ as $[\sigma(\mathbf{u})]_i = \mathbf{u}_{i+1}$ for all $i \in \mathbb{Z}$. The map σ is invertible, and for all $k \in \mathbb{Z}$, the power σ^k is defined by composition. The map σ is continuous on $\mathcal{A}^{\mathbb{Z}}$, therefore, $(\mathcal{A}^{\mathbb{Z}}, \sigma)$ is a dynamical system, which is called a *full shift*.

A bi-infinite word $\mathbf{u} \in \mathcal{A}^{\mathbb{Z}}$ *avoids* a set of finite words X if $\mathcal{L}(\mathbf{u}) \cap X = \emptyset$. By S_X we denote the set of all bi-infinite words that avoid X and we call it a *subshift*. If X is a regular language, S_X is called *sofic subshift*; if X is finite, S_X is called a *subshift of finite type*. The *language* $\mathcal{L}(S)$ of a subshift S is the union of languages of all bi-infinite words from S. By $\mathcal{L}_n(S)$ we denote the set $\mathcal{L}(S) \cap \mathcal{A}^n$.

It holds that a set $S \subseteq \mathcal{A}^{\mathbb{Z}}$ is a subshift if and only if it is invariant under the shift map σ (that means $\sigma(S) = S$) and it is closed with respect to the Cantor metric on $\mathcal{A}^{\mathbb{Z}}$, which is defined as

$$d(\mathbf{u}, \mathbf{v}) = \begin{cases} 0 & \text{if } \mathbf{u} = \mathbf{v}, \\ 2^{-s} & \text{if } \mathbf{u} \neq \mathbf{v}, \text{ where } s := \min\left\{|i| \;\middle|\; \mathbf{u}_i \neq \mathbf{v}_i\right\}. \end{cases}$$

A *labeled graph* over an alphabet $\mathcal{A}$ is a structure $H = (V, E, s, t, h)$, where V is a finite set of vertices, E is a finite set of edges, $s : E \to V$ is a surjective source map, $t : E \to V$ is a target map, and $\mathrm{lab} : E \to \mathcal{A}$ is a labeling function. A word $\mathbf{w}$ ($\mathbf{w}$ being finite of bi-infinite) is a path in H if $t(\mathbf{w}_i) = s(\mathbf{w}_{i+1})$ for all indexes i. A label $\mathrm{lab}(\mathbf{w})$ of the path $\mathbf{w}$ is defined by $(\mathrm{lab}(\mathbf{w}))_i = \mathrm{lab}(\mathbf{w}_i)$ for all indexes i. For a given graph H, we denote by Σ_H the set of all bi-infinite paths in H.

It holds that a subshift S is sofic if and only if there exists a labeled graph H such that $S = \Sigma_H$. Moreover, S is of finite type if and only if there exists such strongly connected graph. General theory of subshifts is very well summarized in [668].

A.3 Lossless seeds

The binary alphabet $\mathcal{A} = \{\#, -\}$ is called *seed alphabet* and from now on, we will consider only this alphabet. Every finite word over this alphabet is a *seed*. The *weight* of a seed Q is the number of occurrences of the letter # in Q.

Definition 1. *Let m and k be positive integers. Every set $\{i_1, \dots, i_k\} \subseteq \{0, \dots, m-1\}$ is called error combination of k errors. A seed Q such that $|Q| < m$ detects an error combination $\{i_1, \dots, i_k\}$ at position $t \in \{0, \dots, \ell\}$ if for all $j \in \{0, \dots, |Q|-1\}$ it holds $\big(Q_j = \# \implies j + t \notin \{i_1, \dots, i_k\}\big)$.*

The implication expresses the fact that there cannot be any mismatch at positions of the "'matching" symbol #. Bi-infinite or finite words over the seed alphabet can be compared using the following relation.

Definition 2. *On the sets $\mathcal{A}^n$ for all $n \in \mathbb{N}$ and $\mathcal{A}^{\mathbb{Z}}$, we define the relation $\preceq$ as:*

$$u \preceq v \qquad \iff \qquad (u_i = \# \implies v_i = \#) \text{ holds for all posible indexes } i.$$

The relation $\preceq$ is reflexive, transitive, and weakly anti-symmetric, hence it is a partial order. Then we define a seed analogy of the logical function OR applied on bi-infinite words and producing, again, a bi-infinite word.

Definition 3. *Consider k bi-infinite words $\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(k)}$ over $\mathcal{A}$. We define a k-nary operation $\oplus$ as:*

$$(\oplus(\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(k)}))_i = \begin{cases} \# & \text{if } (\mathbf{u}^{(j)})_i = \# \text{ for some } j \in \{1, \dots, k\}, \\ - & \text{otherwise} \end{cases}$$

for all $i \in \mathbb{Z}$.

Using the operations σ and $\oplus$, we can easily decide if a specific error combination is detected by a seed at a given position, or not. The following theorem will be crucial for seed analysis in the rest of the text.

Theorem 1. *Let m and k be positive integers and Q be a seed such that $|Q| < m$. Denote $\mathbf{w} := \cdots --|\text{-}^{\ell}Q--\cdots$. Then Q detects an error combination $\{i_1, \dots, i_k\} \subseteq \{0, \dots, m-1\}$ at a position $t \in \{0, \dots, m - |Q|\}$ if and only if*

$$(\oplus(\sigma^{i_1}(\mathbf{w}), \dots, \sigma^{i_k}(\mathbf{w}))_{\ell - t} = -. \tag{A.1}$$

Proof. Q detects $\{i_1, \dots, i_k\}$ at position t if

$$\forall j \in \{0, \dots, |Q|-1\} : \quad Q_j = \# \implies j + t \notin \{i_1, \dots, i_k\}.$$

This is equivalent to $\forall p \in \{i_1, \dots, i_k\}(\mathbf{w}_{p-t+\ell} = -)$, which is equivalent to (A.1). □

Corollary 1 (of Theorem 1)**.** *A seed Q does not detect an error combination $\{i_1, \dots, i_k\} \subseteq \{0, \dots, m-1\}$ at any position t if and only if*

$$(\oplus(\sigma^{i_1}(\mathbf{w}), \dots, \sigma^{i_k}(\mathbf{w}))[0, \ell] = \#^{\ell+1}, \tag{A.2}$$

where $\mathbf{w} = \cdots --|\text{-}^{\ell}Q--\cdots$ and $\ell = m - |Q|$.

Now we can distinguish more ways of usage seeds. The basic case is the so-called *single seed and single hit* problem when the given seed is required to detect every combination of k errors at least at one position. Nevertheless, we can also utilize *families* of seeds such that every error combination of k errors must be detected by some seed from the given family. In both cases, we could also require *multiple hits*, which means that every combination of k errors would have to be detected by given seed or members of seed families at least at h distinct positions for some fixed h.

In the rest of this text, we study only the case of one seed and one hit, however, the results hold for the other cases. In all cases, Theorem 1 and Corollary 1 are the basic tools for studying seeds as languages of some subshifts.

A.4 Single seed and single hit problem

First let us introduce a rigorous definition of the basis case (single seed and single hit).

Definition 4. *A seed Q solves the (m,k)-problem if every error combination of k errors is detected by Q at some position $t \in \{0,\dots,\ell\}$.*

A verification if a given seed Q solves an (m,k)-problem, can be done directly using Corollary 1. Now we define sets of seeds for which we will later show that they coincide with languages of some subshifts.

Definition 5. *For given ℓ and k, the set of all seeds such that each seed Q solves the $(|Q|+\ell,k)$-problem is denoted by* Seed_k^ℓ.

Example 1.

$$\mathrm{Seed}_2^3 = \{\varepsilon, \texttt{\#}, \texttt{-}, \texttt{\#-}, \texttt{-\#}, \texttt{--}, \texttt{\#--}, \texttt{-\#-}, \texttt{--\#}, \texttt{---}, \texttt{\#--\#}, \texttt{\#---}, \texttt{-\#--}, \texttt{--\#-}, \texttt{---\#}, \texttt{----}, \dots\}$$

As a simple consequence of Corollary 1, we get a full characterization of all seeds solving $(m,1)$-problems ([494, Theorem 5]).

Proposition 1. $\mathrm{Seed}_1^\ell = \{Q \in \mathcal{A}^* \mid \texttt{\#}^{\ell+1} \text{ is not a factor of } Q\}$

Proof. Denote $\mathbf{v} = \cdots\texttt{--}|\texttt{-}^\ell Q\texttt{--}\cdots$ and $\ell = m - |Q|$. Then from Corollary 1 follows that: Q solves the $(m,1)$-problem $\iff \forall i \in \mathbb{Z}\left((\sigma^i(\mathbf{v}))[0,\ell] \neq \texttt{\#}^{\ell+1}\right) \iff Q$ does not contain $\texttt{\#}^{\ell+1}$. □

Nevertheless, describing sets Seed_k^ℓ for $k > 1$ is a more complicated task as we will see below. Let us mention that in the case of two errors, Corollary 1 corresponds to the Laser method [494, Section 4.1] as we illustrate in the following example.

Example 2. *Consider a seed $Q =$ ##-#-----#-## of length 14 and the $(19,2)$-problem. In Figure A.1 we show a corresponding schematic table. Denote $\mathbf{w} := \cdots\texttt{--}|\texttt{-}^5 Q\texttt{--}\cdots$. The words $\oplus(\sigma_i(\mathbf{w}),\sigma_j(\mathbf{w}))$ occur diagonally. It is easily seen from Corollary 1 that Q does not detect the error combination $\{5,13\}$ since $\ell = 5$ and $(\oplus(\sigma^5(\mathbf{w}),\sigma^{13}(\mathbf{w})))[0,5] = \texttt{\#}^6$.*

A.4.1 Functions sh_k and (ℓ,k)-valid bi-infinite words

Inspired by Corollary 1, we define functions sh_k which check the criterion given by (A.2) globally on bi-infinite words.

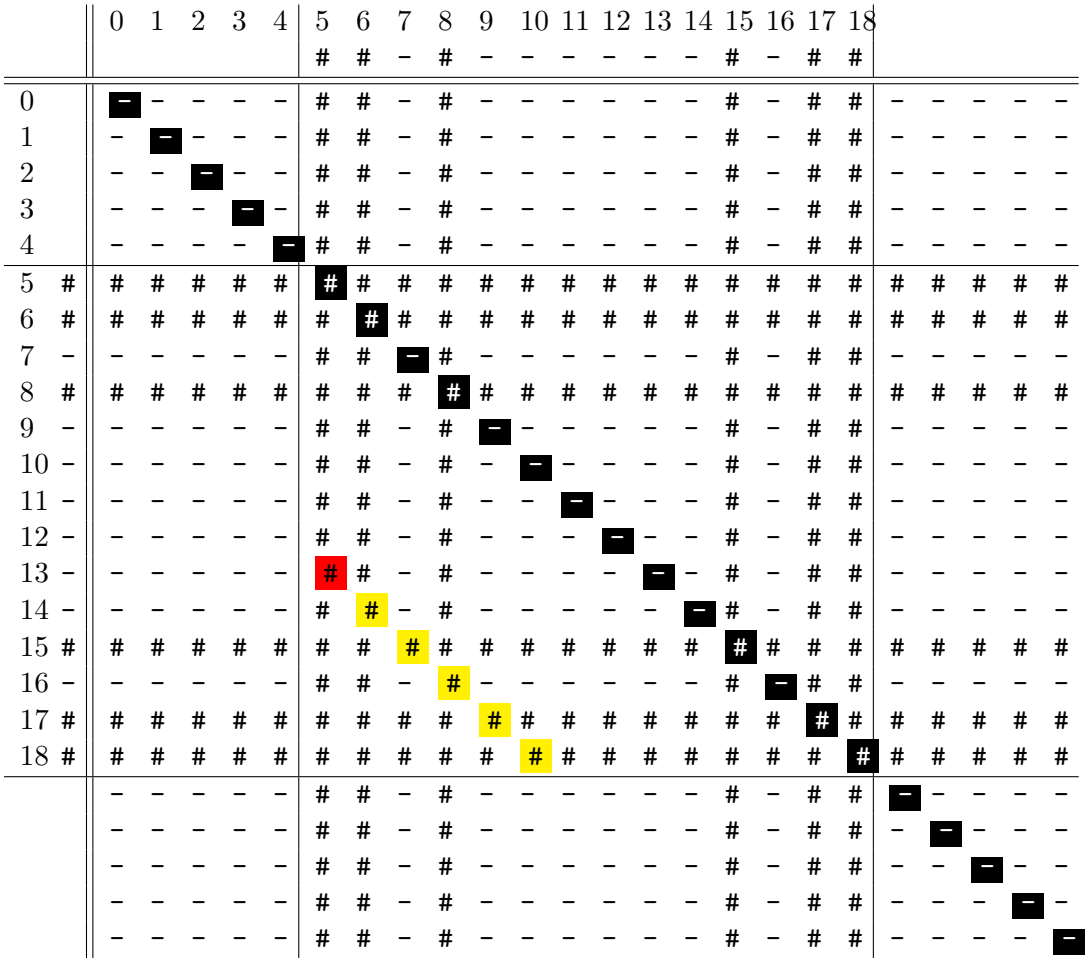


Figure A.1: **Demonstration of the laser method.** Laser method used for the $(19, 2)$-problem and the seed $Q = \texttt{\#\#-\#------\#-\#\#}$ from Example 2.

Definition 6. *Consider a positive integer k. We define a function* $\mathrm{sh}_k : (\mathcal{A}^{\mathbb{Z}})^k \to \mathbb{N}_0 \cup \{+\infty\}$ *as:*

$$\mathrm{sh}_k(\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(k)}) = \sup_{i_1,\ldots,i_k \in \mathbb{Z}} \sup_{p \in \mathbb{N}_0} \left\{ p \mid \mathbf{v}[0, p-1] = \texttt{\#}^p, \text{ where } \mathbf{v} = \oplus\left(\sigma^{i_1}(\mathbf{u}^{(1)}), \ldots, \sigma^{i_k}(\mathbf{u}^{(k)})\right) \right\}. \tag{A.3}$$

We extend the range of the function $\mathrm{sh}_k(\cdot, \ldots, \cdot)$ *to* $(\mathcal{A}^*)^k$. *Finite words w are transformed into bi-infinite words* $\mathbf{v}$ *as* $\mathbf{v} := \cdots \texttt{--}|w\texttt{--}\cdots$.

Informally said, $sh_k(\mathbf{u}^{(1)}, \ldots, \mathbf{u}^{(k)})$ is equal to

- a finite $s \in \mathbb{N}_0$ if after arbitrary "aligning" of the words followed by the logical `OR` operation (in the Laser method the diagonal bi-infinite words), each run of `#`'s has length at most s and the value s is attained for some "alignment";
- $+\infty$ if there exists an "alignment" with run of infinitely many `#`'s (e.g.,

$$\mathrm{sh}_2(\cdots vv|vv\cdots, \cdots ww|ww\cdots)$$

with $v = \texttt{\#\#-}$ and $w = \texttt{\#--}$).

It is readily seen that every function sh_k is symmetric and shift invariant with respect to all variables. The next observations show how to make lower and upper estimates on its value.

Observation 1 (Lower estimate)**.** *Let* $\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(k)}$ *be bi-infinite words. If*

$$\oplus(\sigma^{i_1}(\mathbf{u}^{(1)}), \dots, \sigma^{i_k}(\mathbf{u}^{(k)}))$$

has a factor $\#^p$ *for some* $i_1, \dots, i_k$*; then*

$$\mathrm{sh}_k(\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(k)}) \geq p.$$

Observation 2 (Upper estimate)**.** *Let* $\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(k)}, \mathbf{v}^{(1)}, \dots, \mathbf{v}^{(k)}$ *be bi-infinite words such that* $\mathbf{u}^{(1)} \preceq \mathbf{v}^{(1)}, \dots, \mathbf{u}^{(k)} \preceq \mathbf{v}^{(k)}$*, where* $\preceq$ *is the relation from Definition 2. Then* $\mathrm{sh}_k(\mathbf{u}^{(1)}, \dots, \mathbf{u}^{(k)}) \leq \mathrm{sh}_k(\mathbf{v}^{(1)}, \dots, \mathbf{v}^{(k)})$.

Bi-infinite words for which the sh_k function is bounded by some ℓ, will be the "bricks" our subshifts. Their factors Q are exactly seeds solving $(|Q| + \ell, k)$-problems.

Definition 7. *A bi-infinite word* $\mathbf{u}$ *satisfying* $\mathrm{sh}_k(\mathbf{u}, \dots, \mathbf{u}) \leq \ell$ *is called* $(\ell; k)$*-valid bi-infinite word. By* V_k^ℓ *we denote the set of all* $(\ell; k)$*-valid words.*

Lemma 1. *A seed* Q *solves an* $(|Q| + \ell, k)$*-problem if and only if it is a factor of an* (ℓ, k)*-valid bi-infinite word.*

Proof. $\Longrightarrow$: The word $\mathbf{w} := \cdots --|-^\ell Q -- \cdots \in \mathrm{V}_k^\ell$ is an (ℓ, k)-valid word. must be (ℓ, k)-valid since otherwise Q would not solve the $(|Q| + \ell, k)$-problem by Corollary 1.

$\Longleftarrow$: For a contradiction assume that there exists a factor Q of a bi-infinite word $\mathbf{u}$, which does not solve the $(|Q| + \ell, k)$-problem. Let the non-detected error combination be $\{i_1, \dots, i_k\}$. Denote $\mathbf{w} = \cdots --|-^\ell Q -- \cdots$.

We use shift invariance of sh_k and Observation 2 to get

$$\mathrm{sh}_k(\mathbf{w}, \dots, \mathbf{w}) \leq \mathrm{sh}_k(\mathbf{u}, \dots, \mathbf{u}) \leq \ell. \tag{A.4}$$

Since Q does not detect the error combination $\{i_1, \dots, i_k\}$, it follows from Corollary 1 that

$$(\oplus(\sigma^{i_1}(\mathbf{w}), \dots, \sigma^{i_k}(\mathbf{w}))[0, \ell] = \#^{\ell+1}.$$

Nevertheless, this gives us a lower estimate on $\mathrm{sh}_k(\mathbf{w}, \dots, \mathbf{w})$ – it must be bigger than or equal to $\ell + 1$ which is contradicting (A.4).

□

A.4.2 Subshifts of (ℓ, k)-valid words

The property of (ℓ, k)-validity is preserved under the shift operation. Moreover, the sets V_k^ℓ are subshifts. To prove it, we need to find a criterion for verifying (ℓ, k)-validity based on comparing finite factors of a given bi-infinite word.

Lemma 2. *Let* $\mathbf{u}$ *be a bi-infinite word over the seed alphabet* $\mathcal{A}$*. Then the following statements are equivalent:*

1. $\mathbf{u}$ *is* $(\ell; k)$*-valid;*

2. *each k-tuple $(v^{(1)}, \dots, v^{(k)})$ of factors of $\mathbf{u}$ of length $\ell+1$ satisfies*
$$\mathrm{sh}_k(v^{(1)}, \dots, v^{(k)}) \leq \ell;$$
3. *each k-tuple $(w^{(1)}, \dots, w^{(k)})$ of factors of $\mathbf{u}$ of length $\ell+1$ satisfies*
$$\oplus(w^{(1)}, \dots, w^{(k)}) \neq \texttt{\#}^{\ell+1}. \tag{A.5}$$

Proof. We prove three implications.

$1 \Longrightarrow 2$: Consider any factors $v^{(1)}, \dots, v^{(k)}$. Find their positions $i_1, \dots, i_k$ in $\mathbf{u}$. Since
$$\cdots\texttt{--}|v^{(1)}\texttt{--}\cdots \preceq \sigma^{i_1}(\mathbf{u}), \quad \dots, \quad \cdots\texttt{--}|v^{(k)}\texttt{--}\cdots \preceq \sigma^{i_k}(\mathbf{u}),$$
we obtain from the assumption, shift invariance of sh_k, and Observation 2 that it holds
$$\mathrm{sh}_k(v^{(1)}, \dots, v^{(k)}) \leq \mathrm{sh}_k(\mathbf{u}, \dots, \mathbf{u}) \leq \ell$$
which is what we wanted to prove.

$2 \Longrightarrow 3$: It is an easy consequence of the definition of the sh_k function.

$3 \Longrightarrow 1$: For a contradiction assume that $\mathbf{u}$ is not (ℓ, k)-valid; i.e., there exist integers $i_1, \dots, i_k$ such that $\oplus(\sigma^{i_1}(\mathbf{u}), \dots, \sigma^{i_1}(\mathbf{u}))[0, \ell] = \texttt{\#}^{\ell+1}$. Therefore, the equation (A.5) does not hold for the factors $w^{(1)} := \mathbf{u}[i_1, i_1 + \ell], \dots, w^{(k)} := \mathbf{u}[i_k, i_k + \ell]$. □

In Example 2, such words $w^{(1)}$ and $w^{(2)}$ of length $\ell+1$ are `##-#--` and `--#-##`. Similarly, $v^{(1)}$ and $v^{(2)}$ can be again `##-#--` and `--#-##`, but also for example `-##-#-` and `#-##--`.

The main consequence of Lemma 2 is the fact that every seed must be constructed from reciprocally compatible tiles of length $\ell+1$. To describe this property, we define a relation of compatibility on the set $\mathcal{A}^{\ell+1}$.

Definition 8. *For given ℓ and k, we define the k-nary compatibility relation C_k^ℓ on $\mathcal{A}^{\ell+1}$ as*
$$\mathrm{C}_k^\ell(v^{(1)}, \dots, v^{(k)}) \quad \Longleftrightarrow \quad \mathrm{sh}_k(v^{(1)}, \dots, v^{(k)}) \leq \ell.$$

Corollary 2. *Let $\mathbf{u}$ be a bi-infinite word over the seed alphabet $\mathcal{A}$. The word $\mathbf{u}$ is (ℓ, k)-valid if and only if* $\forall v^{(1)}, \dots, v^{(k)} \in \mathcal{L}_{\ell+1}(\mathbf{u}) \left(\mathrm{C}_k^\ell(v^{(1)}, \dots, v^{(k)})\right)$.

Now let us prove that (ℓ, k)-valid words really form subshifts. We only need to show that (ℓ, k)-valid words are exactly those created from compatible "tiles".

Lemma 3. *Every set V_k^ℓ is a subshift.*

Proof. Take $X := \left\{x \in \mathcal{A}^* \mid \exists\{v^{(1)}, \dots, v^{(k)}\} \subseteq \mathcal{L}_{\ell+1}(x) \left(\neg\mathrm{C}_k^\ell(v^{(1)}, \dots, v^{(k)})\right)\right\}$. The set X contains all finite words containing some incompatible factors. Then it follows from Corollary 2 that $S_X = \mathrm{V}_k^\ell$. □

Example 3. *Even though the seeds $Q^{(1)} =$ `##-#--` and $Q^{(2)} =$ `--#-##` solve the $(11, 2)$-problem, the seed $Q = Q^{(1)}$`--`$Q^{(2)}$ does not solve the $(19, 2)$-problem, as we have seen in Example 2. The reason is that, by Lemma 2, $Q^{(1)}$ and $Q^{(2)}$ are not compatible. Therefore, any seed $\tilde{Q}$ of the form $\tilde{Q} = Q^{(1)}$`-`$^p Q^{(2)}$ for any $p \in \mathbb{N}_0$, cannot solve the $(|\tilde{Q}| + 5, 2)$-problem.*

$$
\begin{array}{rccccccccccc}
v^{(1)} = & Q_0 & Q_1 & \dots & Q_\ell & & & & & & & \\
v^{(2)} = & & Q_1 & Q_2 & \dots & Q_{\ell+1} & & & & & & \\
 & & & \ddots & & & & & & & & \\
 & & & & & & Q_{n-\ell-2} & Q_{n-\ell-1} & \dots & Q_{n-2} & & \\
v^{(n-\ell)} = & & & & & & & Q_{n-\ell-1} & Q_{n-\ell} & \dots & Q_{n-1} & \\
\hline
Q = & Q_0 & Q_1 & Q_2 & \dots & & Q_{n-\ell-2} & Q_{n-\ell-1} & Q_{n-\ell} & \dots & Q_{n-1} &
\end{array}
$$

Figure A.2: **Domino-game-like principle.** The words $v^{(1)}, \dots, v^{(n-\ell)}$ must belong to the same generating set G.

A.4.3 Decomposition into subshifts of finite type

From Example 3 follows that the subshift V_k^ℓ of all (ℓ, k)-valid words is not necessarily of finite type. Nevertheless, it must be a union of subshifts of finite type because we can construct maximal families of reciprocally compatible words of length $\ell + 1$ and then design seeds by domino-game-like principle (see Figure A.2).

Definition 9. *For given positive integers ℓ and k, a subset G of $\mathcal{A}^{\ell+1}$ is called (ℓ, k)-generating set if the following conditions are satisfied:*

1. *for all $v^{(1)}, \dots, v^{(k)} \in G$, it holds $\mathrm{C}_k^\ell(v^{(1)}, \dots, v^{(k)})$;*
2. *it cannot contain any other word from $\mathcal{A}^{\ell+1}$.*

Let us remark that every generating set G is closed with respect to the logical shift with the filling symbol -. It means that after taking arbitrary word from the set G, removing its first (or last), and concatenating - to end (or beginning), we get again a word from G.

It easily seen that every generating set G fully determines a subshift of finite type, which we denote by $S(G)$.

Definition 10. *Consider a seed Q and an (ℓ, k)-generating set G. By $S(G)$, we denote the subshift S_X of finite type given by $X = \mathcal{A}^{\ell+1} \backslash G$. We say that a seed Q is generated by G if $Q \in \mathcal{L}(S(G))$.*

We could easily prove that every bi-infinite word can be created using some generating set as the following observation states.

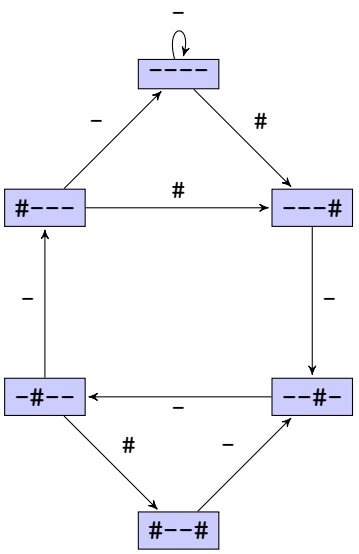
Observation 3. *For every (ℓ, k)-valid bi-infinite word $\mathbf{u}$, there exists an (ℓ, k)-generating set G such that $\mathbf{u} \in S(G)$.*

Example 4. *Continue with the setting from Example 1. Consider the only one $(3, 2)$-generating set $G = \{$#--#, #---, -#--, --#-, ---#, ----$\}$. Since $S(G)$ is of finite type, we can find a strongly connected labeled graph H such that $S(G) = \Sigma_H$, where Σ_H is the set of all bi-infinite paths in H. This graph also determines a finite automaton recognizing the set $\mathcal{L}(S(G))$. We can create such automaton using a de-Bruijn graph and setting the initial state $-^{\ell+1}$, nevertheless, it is not minimal as it is shown in Figure A.3.*

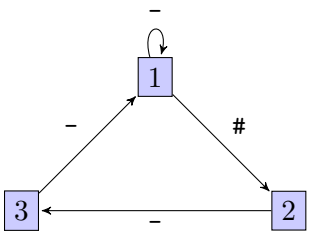
Theorem 2. *Every set Seed_k^ℓ is a regular language.*

Proof. There can be only finite number of (ℓ, k)-generating sets; denote them $G_1, \dots, G_d$. It follows from Observation 3 that $S(G_1) \cup \dots \cup S(G_d) = \mathrm{V}_k^\ell$ and, from Lemma 1, we know that $\mathcal{L}(\mathrm{V}_k^\ell) = \mathrm{Seed}_k^\ell$.

For every $i \in \{1, \dots, d\}$, the set $S(G_i)$ is a subshift of finite type, so every set $\mathcal{L}(S(G_i))$ is a regular language. Since the set Seed_k^ℓ is a union of finitely many regular languages, it is a regular language. □



(A) A graph created as a de-Bruijn graph from the set of vertices G.



(B) The previous graph after minimization.

Figure A.3: **Seed graphs.** Labeled graphs for the subshift $S(G)$ given by the unique $(3,2)$-generating set $G = \{\texttt{\#--\#}, \texttt{\#---}, \texttt{-\#--}, \texttt{--\#-}, \texttt{---\#}, \texttt{----}\}$. Both graphs H must satisfy $\Sigma_H = S(G)$. Every such graph H determines a deterministic recognizing automaton for the set $\mathcal{L}(\Sigma_H)$.

Remark 1. *For $k = 2$ and arbitrary positive ℓ, we can derive all generating sets. Let $V = \{w^{(1)}, \dots, w^{(q)}\}$ denote the set of all seeds of length $\ell+1$ solving the $(2\ell+1, k)$-problem. Consider a graph R given by the adjacency matrix*

$$(M_R)_{i,j} = \begin{cases} 0 & \text{if } \mathrm{C}_2^{\ell}(w^{(i)}, w^{(j)}), \\ 1 & \text{otherwise.} \end{cases}$$

Then the generating sets are maximal independent sets (maximal with respect to inclusion) in the graph R. Similar derivation can be made for cases with $k > 2$ using hypergraphs.

Example 5. *Let $k = 2$. For $\ell \in \{1, \dots, 4\}$, all seeds solving $(\ell+1, 2)$-problems are mutually compatible, which means that there exists an unique $(\ell, 2)$-generating set. We list them out in the following table.*

ℓ	G
1	{--}
2	{#--, -#-, --#, ---}
3	{#--#, #---, -#--, --#-, ---#, ---}
4	{##---, -##--, --##-, ---##, #-#--, -#-#-, --#-#, #--#-, -#--#, #---#, #----, -#---, --#--, ---#-, ----#, -----}

Example 6. *Let $k = 2$ and $\ell = 5$. By Remark 1, we find the graph R. We can partially simplify this graph. We can say that two vertices v and w in this graph are equivalent if $\forall x \in V(\mathrm{C}_k^{\ell}(x, v) \iff \mathrm{C}_k^{\ell}(x, w))$. Then we can put equivalent vertices into one vertex and*

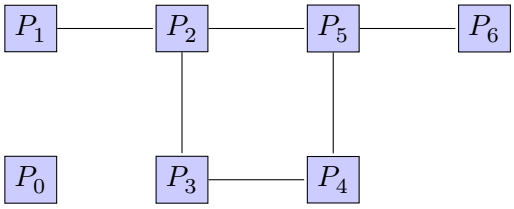


Figure A.4: **Sets of equal seeds.** Graph of sets of equal seeds for searching $(5, 2)$-generating sets in Example 6.

we obtain the graph in Figure A.4, where

$$
\begin{aligned}
P_0 = \{&\texttt{------};\ \texttt{-----\#},\ \texttt{----\#-},\ \texttt{---\#--},\ \texttt{--\#---},\ \texttt{-\#----},\ \texttt{\#-----};\\
&\texttt{----\#\#},\ \texttt{---\#\#-},\ \texttt{--\#\#--},\ \texttt{-\#\#---},\ \texttt{\#\#----};\\
&\texttt{---\#-\#},\ \texttt{--\#-\#-},\ \texttt{-\#-\#--},\ \texttt{\#-\#---};\\
&\texttt{--\#--\#},\ \texttt{-\#--\#-},\ \texttt{\#--\#--};\ \texttt{-\#---\#},\ \texttt{\#---\#-};\\
&\texttt{\#---\#\#};\ \texttt{\#\#---\#};\ \texttt{\#----\#}\},\\
P_1 = \{&\texttt{\#--\#-\#}\}, \qquad P_2 = \{\texttt{--\#\#-\#},\ \texttt{-\#\#-\#-},\ \texttt{\#\#-\#--}\},\\
P_3 = \{&\texttt{-\#\#--\#},\ \texttt{\#\#--\#-}\}, \qquad P_4 = \{\texttt{-\#--\#\#},\ \texttt{\#--\#\#-}\},\\
P_5 = \{&\texttt{--\#-\#\#},\ \texttt{-\#-\#\#-},\ \texttt{\#-\#\#--}\}, \qquad P_6 = \{\texttt{\#-\#--\#}\}.
\end{aligned}
$$

We can observe that P_1, P_2, and P_3 can be obtained by mirroring from P_6, P_5, and P_4, respectively. By finding maximal independent sets in the graph in Figure A.4, we get all $(5, 2)$-generating sets:

$$
\begin{aligned}
G_1 = P_0 \cup P_1 \cup P_3 \cup P_5, G_2 &= P_0 \cup P_1 \cup P_3 \cup P_6,\\
G_3 &= P_0 \cup P_2 \cup P_4 \cup P_6,\\
G_4 &= P_0 \cup P_1 \cup P_4 \cup P_6.
\end{aligned}
$$

A.5 Conclusion

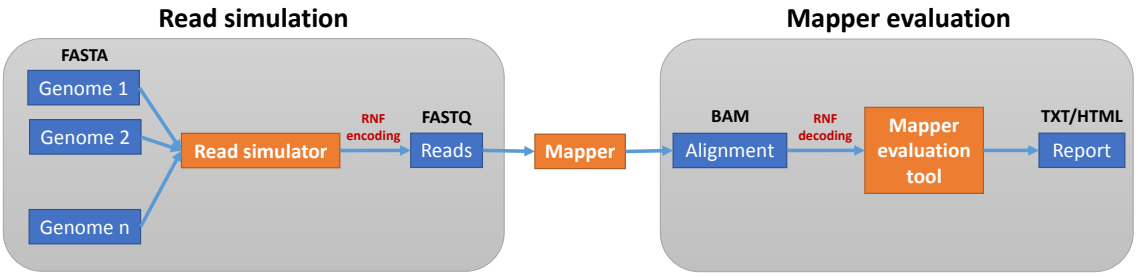
We have found a new criterion for errors detection by seeds (Theorem 1). From this criterion we have proven that lossless seeds coincide with languages of certain sofic subshifts, therefore, they are recognized by finite automata (Theorem 2). We have shown that these subshifts are fully given by the number of allowed errors k and the seed margin ℓ and that they can be further decomposed into subshifts of finite type.

These facts explain why periodically repeated patterns often appear in lossless seeds. This is caused by the fact that these patterns correspond to cycles in the recognizing automata (which correspond to cyclic seeds from [486]). Nevertheless, it remains unclear what is the upper bound on the length of cycles to obtain optimal seeds. For problems with two errors, it was conjectured in [494, Conjecture 1] that to obtain some of optimal seeds, it is sufficient to consider only patterns having length at most $\ell + 1$.

Appendix B

Read Naming Format specification

This chapter provides a standard for naming simulated Next-Generation Sequencing (NGS) reads in order to make read simulators and mapper evaluation tools inter-compatible. The current version is available from https://github.com/karel-brinda/rnf-spec.



B.1 Terminologies and concepts

Read tuple. A tuple of sequences (possibly overlapping) obtained from a sequencing machine from a single fragment of DNA.

Reads. Members of a *read tuple*. For example, every "paired-end read" is a *read tuple* and both of its "ends" are individual *reads* in our notation.

Segments. Substrings of a *read* which are spatially distinct in the reference. They correspond to individual lines in a SAM file [125]. Thus, each *read* has an associated chain of *segments* and we associate a *read tuple* with *segments* of all its *reads*.

Remarks:

- A "single-end read" consists of a single *read* with a single *segment* unless it comes from a region with genomic rearrangment
- A "paired-end read" or a "mate-pair read" consists of two *reads*, each with one *segment* (under the same condition).
- A "strobe read" consists of several *reads*.
- A chimeric *read* (i.e., read corresponding to a genomic fusion, a long deletion, or a translocation) has at least two *segments*.

```
Coor        12345678901234-5678901234567890123456789

Source 1 - reference genome
chr 1       ATGTTAGATAA-GATAGCTGTGCTAGTAGGCAGTCAGCCC
chr 2       ttcttctggaa-gaccttctcctcctgcaaataaa

Source 2 - generator of random sequences

READS:
r001        ATG-TAGATA ->
r002/1         TTAGATAACGA ->
r002/2                                  <- TCAG-CGGG
r003/1                                tgcaaataa ->
r003/2              gaa-gacc-t ->
r004                       ATAGCT............TCAG ->
r005                                   GTAGG ->
             <- agacctt
                         <- TCGACACG
r006       ATATCACATCATTAGACACTA
```

(A) Simulated reads

r. tuple	LRN	SRN
r001	`sim__1__(1,1,F,01,10)__[single-end]`	#1
r002	`sim__2__(1,1,F,04,14),(1,1,R,31,39)__[paired-end]`	#2
r003	`sim__3__(1,2,F,09,17),(1,2,F,25,33)__[mate-pair]`	#3
r004	`sim__4__(1,1,F,15,36)__[spliced],C:[6=12N4=]`	#4
r005	`sim__5__(1,1,R,15,22),(1,1,F,25,29),(1,2,R,05,11)__[chimeric]`	#5
r006	`rnd__6__(2,0,N,00,00)__[random-contamination]`	#6

(B) Long and short read names

Figure B.1: **Example of RNF names.** Simulated *read tuples* and their corresponding RNF names which can be used as read names in the final FASTQ files:
- single-end read (r001);
- paired-end read (r002);
- mate-pair read (r003);
- spliced RNA-seq read (r004);
- chimeric read (r005);
- random contaminating read with unspecified coordinates (r006).

- RNA-seq spliced reads are not considered to be spatially distinct.

Simulator of NGS reads. A program which creates artificial simulated reads from one or more (possibly random) reference genomes.

Evaluation tool of NGS mappers. A program which evaluates alignments of simulated NGS reads with known original genomic positions. It assesses if each individual read is aligned correctly. Finally it usually creates overall statistics.

1**-based coordinate system.** A coordinate system where the first position has number 1 and intervals are closed (the same system is used by the SAM format [125]).

B.2 Read tuple names

To every *read tuple*, two names are assigned: Short read name (SRN) and Long read name (LRN).

SRN contains a hexademical unique *read tuple* ID prefixed by '`#`'.

LRN consists of four parts delimited by double-underscore:

i) a prefix (possibly containing expressive information for a user or a particular string for sorting or randomization of order of tuples),

ii) the *read tuple* ID,

iii) information about origins of all *segments* that constitute *reads* of the *tuple*,

iv) a suffix containing arbitrary comments or extensions (for holding additional information).

Preferred final read names are LRNs. If an LRN exceeds 255 (maximum allowed read length in SAM[125]), SRNs are used instead and a SRN–LRN correspondence file must be created.

B.2.1 Read tuple ID

It is a positive integer, which is unique within a single file with genomic data. These IDs are assigned continuously from 1. Zero is reserved for "not available".

B.2.2 SRN – Short Read Name

Matching regular expression:

```
\#([0-9a-f]+)
```

SRN consists of *read tuple* ID prefixed by '`#`'. It is displayed as zero padded hexadecimals in lowercase such that all SRNs share the same string length within a single file.

B.2.3 LRN – Long Read Name

Matching regular expression:

```
^([!-?A-^`-~]*)__([0-9a-f]+)__([!-?A-^`-~]+)__([!-?A-^`-~]*)$
```

LRN consists of four double-underscore-delimited parts: i) prefix part, ii) *read tuple* ID, iii) segmental part, iv) suffix part.

Prefix part

Matching regular expression:

```
^[!-?A-^`-~]*$
```

It can be an empty string, a string containing expressive "visual" information for the user (e.g., for easy distinguishing random reads from the others), or a string used for randomization of *read tuples* (randomly taken prefix and *read tuples* sorted in lexicographical order).

Length of all prefix parts within a single file must be equal.

Read tuple ID part

Matching regular expression:

```
^[0-9a-f]+$
```

It displays *read tuple* ID as hexadecimals in lowercase. All *read tuple* ID parts are zero padded such that they all share the same string length within a single file.

Segmental part

Matching regular expression:

```
^(?:(\([0-9FRN,]*\))(?:,(?!$)|$))+$
```

Segmental part consists of one or more comma-delimited segments.

Segment

Matching regular expression:

```
^\(([0-9]+),([0-9]+),([FRN]),([0-9]+),([0-9]+)\)$
```

Every segment is parenthesized and consists of five comma-delimited values: i) genome ID, ii) chromosome ID, iii) direction, iv) leftmost coordinate, and v) rightmost coordinate.

Genome ID	ID (positive integer) of the source genome (a randomly generated genome, a genome saved in a FASTA file, etc.) or zero for "not available". All numbers of genomes are displayed as decimals and they are zero padded such that they all share the same string length within a single file.
Chromosome ID	ID (positive integer) of the source chromosome or zero for "not available". All numbers of chromosomes are displayed as decimals and they are zero padded such that they all share the same string length within a single file. IDs are assigned continuously from 1, order chromosomes is the same as in the file, where the genome is saved. In case of a random genome, zero should be used.
Direction	Direction in the reference genome. 'F' = forward direction 'R' = reverse direction 'N' = not available For random reads, 'N' should be used.
Leftmost coordinate	The leftmost coordinate of the segment in the reference in 1-based coordinate system or zero for "not available".
Rightmost coordinate	The rightmost coordinate of the segment in the reference in 1-based coordinate system or zero for "not available".

Suffix part

Matching regular expression:

```
^(?:((?:[a-zA-Z0-9]+:){0,1})\[([!-?A-Z\\^`-~]*)\](?:,(?!$)|$))+$
```

It contains arbitrary number of comma-delimited comments and extensions in any order.

Comment

Matching regular expression:

```
^\[([!-?A-Z\\^`-~]*)\]$
```

Comments are displayed as square-bracketed strings. They can contain, e.g., information about the simulated technology or the program used for simulation.

Extension

Matching regular expression:

```
^([A-Za-z0-9]+):\[([!-?A-Z\\^`-~]*)\]$
```

An extension consist of an extension's code, a colon, and a square-bracketed extension's content. Extensions can supplement the basic set of information provided in segmental part. Some of them are part of this standard, see Section B.4.

B.3 SRN–LRN correspondence file

To encode information about correspondence between SRN and LRN, a special file is created. Its file name is formed of prefix of the FASTQ file(s) and `.sl` suffix.

Examples:

Read files	SRN-LRN correspondence file
`reads_se.fq`	`reads_se.sl`
`reads_se.fastq`	`reads_se.sl`
`reads_pe.1.fq`, `reads_pe.2.fq`	`reads_pe.sl`

It is a tab delimited file with two columns (containing SRN and the corresponding LRN). File is sorted by *read tuple* ID.

B.4 Extensions

Extensions can supplement the basic set of information provided in the segmental part (Section B.2.3).

C – CIGAR strings

Extension's code

C

Extension's content

Matching regular expression:

```
^(?:([0-9]+[=XIDNSHPM]+)(?:,(?!$)|$))+$
```

Specification

The extension can be used to encode edit operations using CIGAR (Compact Idiosyncratic Gapped Alignment Report) strings as they are defined with the SAM specification[1].

CIGAR strings should be provided in the same order as their corresponding segments in the segmental part (Section B.2.3). Adjacent edit operations should be different.

Example

```
demonstration__004__(1,1,F,16,40),(1,1,R,140,150)__C:[6=14N5=,11=],[spliced-PE-read]
```

[1] https://samtools.github.io/hts-specs/SAMv1.pdf

Bibliography

[1] K. Břinda, V. Boeva, and G. Kucherov. "Dynamic read mapping and online consensus calling for better variant detection". In: *arXiv preprints* (2016). URL: http://arxiv.org/abs/1605.09070 (cit. on p. vii).

[2] K. Břinda, V. Boeva, and G. Kucherov. "RNF: a general framework to evaluate NGS read mappers." In: *Bioinformatics* 32.1 (2016), pp. 136–9. DOI: 10.1093/bioinformatics/btv524 (cit. on pp. vii, 54, 56–58).

[3] K. Břinda, M. Sykulski, and G. Kucherov. "Spaced seeds improve k-mer-based metagenomic classification." In: *Bioinformatics* 31.22 (2015), pp. 3584–92. DOI: 10.1093/bioinformatics/btv419 (cit. on pp. vii, 30–32, 76).

[4] K. Břinda. "Languages of lossless seeds". In: *Electronic Proceedings in Theoretical Computer Science* 151 (2014), pp. 139–150. DOI: 10.4204/EPTCS.151.9 (cit. on pp. vii, 31).

[5] P. Červenka et al. "Blind Friendly Maps". In: *Computers Helping People with Special Needs: 15th International Conference, ICCHP 2016, Linz, Austria, July 13-15, 2016, Proceedings, Part II*. 2016, pp. 131–138. DOI: 10.1007/978-3-319-41267-2_18 (cit. on p. vii).

[6] E. T. Cirulli and D. B. Goldstein. "Uncovering the roles of rare variants in common disease through whole-genome sequencing". In: *Nature Reviews Genetics* 11.6 (2010), pp. 415–425. DOI: 10.1038/nrg2779 (cit. on p. 5).

[7] B. E. Stranger, E. A. Stahl, and T. Raj. "Progress and Promise of Genome-Wide Association Studies for Human Complex Trait Genetics". In: *Genetics* 187.2 (2011), pp. 367–383. DOI: 10.1534/genetics.110.120907 (cit. on p. 5).

[8] M. T. Maurano et al. "Systematic Localization of Common Disease-Associated Variation in Regulatory DNA". In: *Science* 337.6099 (2012), pp. 1190–1195. DOI: 10.1126/science.1222794 (cit. on p. 5).

[9] P. Bradley et al. "Rapid antibiotic-resistance predictions from genome sequence data for Staphylococcus aureus and Mycobacterium tuberculosis". In: *Nature Communications* 6 (2015), p. 10063. DOI: 10.1038/ncomms10063 (cit. on p. 5).

[10] K. Schmidt et al. "Identification of bacterial pathogens and antimicrobial resistance directly from clinical urines by nanopore-based metagenomic sequencing". In: *Journal of Antimicrobial Chemotherapy* (2016), dkw397. DOI: 10.1093/jac/dkw397 (cit. on p. 5).

[11] Human Microbiome Project Consortium. "Structure, function and diversity of the healthy human microbiome." In: *Nature* 486.7402 (2012), pp. 207–14. DOI: 10.1038/nature11234 (cit. on p. 5).

[12] S. Sunagawa et al. “Structure and function of the global ocean microbiome”. In: *Science* 348.6237 (2015), pp. 1261359–1261359. DOI: 10.1126/science.1261359 (cit. on p. 5).

[13] J. R. Brum et al. “Patterns and ecological drivers of ocean viral communities”. In: *Science* 348.6237 (2015), pp. 1261498–1261498. DOI: 10.1126/science.1261498 (cit. on p. 5).

[14] C. de Vargas et al. “Eukaryotic plankton diversity in the sunlit ocean”. In: *Science* 348.6237 (2015), pp. 1261605–1261605. DOI: 10.1126/science.1261605 (cit. on pp. 5, 79).

[15] E. Afshinnekoo et al. “Geospatial Resolution of Human and Bacterial Diversity with City-Scale Metagenomics”. In: *Cell Systems* 1.1 (2015), pp. 72–87. DOI: 10.1016/j.cels.2015.01.001 (cit. on p. 5).

[16] Y.-S. Jeon, J. Chun, and B.-S. Kim. “Identification of Household Bacterial Community and Analysis of Species Shared with Human Microbiome”. In: *Current Microbiology* 67.5 (2013), pp. 557–563. DOI: 10.1007/s00284-013-0401-y (cit. on p. 5).

[17] J. Qin et al. “A human gut microbial gene catalogue established by metagenomic sequencing”. In: *Nature* 464.7285 (2010), pp. 59–65. DOI: 10.1038/nature08821 (cit. on pp. 5, 73).

[18] K. A. Lipinski et al. “Cancer Evolution and the Limits of Predictability in Precision Cancer Medicine”. In: *Trends in Cancer* 2.1 (2016), pp. 49–63. DOI: 10.1016/j.trecan.2015.11.003 (cit. on p. 6).

[19] L. Orlando, M. T. P. Gilbert, and E. Willerslev. “Reconstructing ancient genomes and epigenomes”. In: *Nature Reviews Genetics* 16.7 (2015), pp. 395–408. DOI: 10.1038/nrg3935 (cit. on p. 6).

[20] E. Hagelberg, M. Hofreiter, and C. Keyser. “Ancient DNA: the first three decades”. In: *Philosophical Transactions of the Royal Society B: Biological Sciences* 370.1660 (2014), pp. 20130371–20130371. DOI: 10.1098/rstb.2013.0371 (cit. on p. 6).

[21] T. E. King et al. “Identification of the remains of King Richard III”. In: *Nature Communications* 5.5631 (2014), p. 5631. DOI: 10.1038/ncomms6631 (cit. on p. 6).

[22] M. H. Larmuseau et al. “Biohistorical materials and contemporary privacy concerns-the forensic case of King Albert I”. In: *Forensic Science International: Genetics* 24 (2016), pp. 202–210. DOI: 10.1016/j.fsigen.2016.07.008 (cit. on p. 6).

[23] E. Jehaes et al. “Mitochondrial DNA analysis of the putative heart of Louis XVII, son of Louis XVI and Marie-Antoinette”. In: *European Journal of Human Genetics* 9.3 (2001), pp. 185–190. DOI: 10.1038/sj.ejhg.5200602 (cit. on p. 6).

[24] Z. Hawass et al. “Ancestry and Pathology in King Tutankhamun’s Family”. In: *JAMA: The Journal of the American Medical Association* 303.7 (2010), p. 638. DOI: 10.1001/jama.2010.121 (cit. on p. 6).

[25] M. Kayser. “Forensic DNA Phenotyping: Predicting human appearance from crime scene material for investigative purposes”. In: *Forensic Science International: Genetics* 18 (2015), pp. 33–48. DOI: 10.1016/j.fsigen.2015.02.003 (cit. on p. 6).

[26] A. M. Whatmore. “Ancient-Pathogen Genomics: Coming of Age?” In: *mBio* 5.5 (2014), e01676–14–e01676–14. DOI: 10.1128/mBio.01676-14 (cit. on p. 6).

[27] K. M. Harkins and A. C. Stone. "Ancient pathogen genomics: insights into timing and adaptation". In: *Journal of Human Evolution* 79 (2015), pp. 137–149. DOI: 10.1016/j.jhevol.2014.11.002 (cit. on p. 6).

[28] G. Morelli et al. "Yersinia pestis genome sequencing identifies patterns of global phylogenetic diversity". In: *Nature Genetics* 42.12 (2010), pp. 1140–1143. DOI: 10.1038/ng.705 (cit. on p. 6).

[29] Y. Cui et al. "Historical variations in mutation rate in an epidemic pathogen, Yersinia pestis". In: *Proceedings of the National Academy of Sciences* 110.2 (2013), pp. 577–582. DOI: 10.1073/pnas.1205750110 (cit. on p. 6).

[30] D. M. Wagner et al. "Yersinia pestis and the Plague of Justinian 541–543 AD: a genomic analysis". In: *The Lancet Infectious Diseases* 14.4 (2014), pp. 319–326. DOI: 10.1016/S1473-3099(13)70323-2 (cit. on p. 6).

[31] C. P. Andam et al. "Microbial Genomics of Ancient Plagues and Outbreaks". In: *Trends in Microbiology* xx (2016), pp. 1–13. DOI: 10.1016/j.tim.2016.08.004 (cit. on p. 6).

[32] C. J. Adler et al. "Sequencing ancient calcified dental plaque shows changes in oral microbiota with dietary shifts of the Neolithic and Industrial revolutions". In: *Nature Genetics* 45.4 (2013), pp. 450–455. DOI: 10.1038/ng.2536 (cit. on p. 6).

[33] M. Slatkin and F. Racimo. "Ancient DNA and human history". In: *Proceedings of the National Academy of Sciences* 2016.23 (2016), pp. 1–8. DOI: 10.1073/pnas.1524306113 (cit. on p. 6).

[34] J. G. Schraiber and J. M. Akey. "Methods and models for unravelling human evolutionary history". In: *Nature Reviews Genetics* November (2015). DOI: 10.1038/nrg4005 (cit. on p. 6).

[35] M. Meyer et al. "A High-Coverage Genome Sequence from an Archaic Denisovan Individual". In: *Science* 338.6104 (2012), pp. 222–226. DOI: 10.1126/science.1224344 (cit. on p. 6).

[36] K. Prüfer et al. "The complete genome sequence of a Neanderthal from the Altai Mountains." In: *Nature* 505.7481 (2014), pp. 43–9. DOI: 10.1038/nature12886 (cit. on p. 6).

[37] "Partial Genetic Turnover in Neandertals: Continuity in the East and Population Replacement in the West". In: *Molecular Biology and Evolution* 29.8 (2012), pp. 1893–1897. DOI: 10.1093/molbev/mss074 (cit. on p. 6).

[38] R. E. Green et al. "A draft sequence of the Neandertal genome." In: *Science (New York, N.Y.)* 328.5979 (2010), pp. 710–22. DOI: 10.1126/science.1188021 (cit. on p. 6).

[39] R. E. Green et al. "The Neandertal genome and ancient DNA authenticity". In: *The EMBO journal* 28.17 (2009), pp. 2494–2502. DOI: 10.1038/emboj.2009.222 (cit. on p. 6).

[40] A. W. Briggs et al. "Targeted retrieval and analysis of five Neandertal mtDNA genomes". In: *Science* 325.1095-9203 (Electronic) (2009), pp. 318–321. DOI: 10.1126/science.1174462 (cit. on p. 6).

[41] R. E. Green et al. "Analysis of one million base pairs of Neanderthal DNA." In: *Nature* 444.7117 (2006), pp. 330–6. DOI: 10.1038/nature05336 (cit. on p. 6).

[42] A. Bergström et al. “Deep Roots for Aboriginal Australian Y Chromosomes”. In: *Current Biology* (2016), pp. 1–5. DOI: 10.1016/j.cub.2016.01.028 (cit. on p. 6).

[43] P. Skoglund and D. Reich. “A genomic view of the peopling of the Americas”. In: *Current Opinion in Genetics & Development* 41 (2016), pp. 27–35. DOI: 10.1016/j.gde.2016.06.016 (cit. on p. 6).

[44] K. Eaton et al. *Museum of London Report on the DNA Analyses of Four Roman Individuals*. Tech. rep. 2015, pp. 1–23 (cit. on p. 6).

[45] S. Schiffels et al. “Iron Age and Anglo-Saxon genomes from East England reveal British migration history.” In: *Nature communications* 7 (2016), p. 10408. DOI: 10.1038/ncomms10408 (cit. on p. 6).

[46] S. Goodwin, J. D. McPherson, and W. R. McCombie. “Coming of age: ten years of next-generation sequencing technologies”. In: *Nature Reviews Genetics* 17.6 (2016), pp. 333–351. DOI: 10.1038/nrg.2016.49 (cit. on pp. 9, 12, 13).

[47] S. Canzar and S. L. Salzberg. “Short Read Mapping: An Algorithmic Tour”. In: *Proceedings of the IEEE* (2015), pp. 1–23. DOI: 10.1109/JPROC.2015.2455551 (cit. on pp. 10, 17, 23, 24).

[48] J. K. Kulski. “Next-Generation Sequencing — An Overview of the History, Tools, and “Omic” Applications”. In: *Next Generation Sequencing - Advances, Applications and Challenges*. InTech, 2016, pp. 3–60. DOI: 10.5772/61964 (cit. on p. 9).

[49] L. Nederbragt. *Developments in NGS*. Tech. rep. 2016. DOI: 10.6084/m9.figshare.100940.v9 (cit. on p. 11).

[50] S. D. Kahn. “On the future of genomic data.” In: *Science (New York, N.Y.)* 331.6018 (2011), pp. 728–9. DOI: 10.1126/science.1197891 (cit. on p. 12).

[51] L. Liu et al. “Comparison of Next-Generation Sequencing Systems”. In: *Journal of Biomedicine and Biotechnology* 2012 (2012), pp. 1–11. DOI: 10.1155/2012/251364 (cit. on pp. 10–13).

[52] M. Kircher and J. Kelso. “High-throughput DNA sequencing - concepts and limitations”. In: *BioEssays* 32.6 (2010), pp. 524–536. DOI: 10.1002/bies.200900181 (cit. on p. 10).

[53] J. C. Venter. “The Sequence of the Human Genome”. In: *Science* 291.5507 (2001), pp. 1304–1351. DOI: 10.1126/science.1058040 (cit. on p. 10).

[54] E. S. Lander et al. “Initial sequencing and analysis of the human genome.” In: *Nature* 409.6822 (2001), pp. 860–921. DOI: 10.1038/35057062 (cit. on p. 10).

[55] E. R. Mardis. “Next-Generation Sequencing Platforms”. In: *Annual Review of Analytical Chemistry* 6.1 (2013), pp. 287–303. DOI: 10.1146/annurev-anchem-062012-092628 (cit. on p. 11).

[56] M. Schirmer et al. “Illumina error profiles: resolving fine-scale variation in metagenomic sequencing data”. In: *BMC Bioinformatics* 17.1 (2016), p. 125. DOI: 10.1186/s12859-016-0976-y (cit. on p. 12).

[57] M. Schirmer et al. “Insight into biases and sequencing errors for amplicon sequencing with the Illumina MiSeq platform”. In: *Nucleic Acids Research* 43.6 (2015), e37–e37. DOI: 10.1093/nar/gku1341 (cit. on p. 12).

[58] H. Li. “BFC: correcting Illumina sequencing errors”. In: *Bioinformatics* 31.17 (2015), pp. 2885–2887. DOI: 10.1093/bioinformatics/btv290 (cit. on p. 12).

[59] E.-C. Lim et al. "Trowel: a fast and accurate error correction module for Illumina sequencing reads". In: *Bioinformatics* 30.22 (2014), pp. 3264–3265. DOI: 10.1093/bioinformatics/btu513 (cit. on p. 12).

[60] Y. Heo et al. "BLESS: Bloom filter-based error correction solution for high-throughput sequencing reads". In: *Bioinformatics* 30.10 (2014), pp. 1354–1362. DOI: 10.1093/bioinformatics/btu030 (cit. on p. 12).

[61] Y. Heo et al. "BLESS 2: accurate, memory-efficient and fast error correction method". In: *Bioinformatics* 32.15 (2016), pp. 2369–2371. DOI: 10.1093/bioinformatics/btw146 (cit. on p. 12).

[62] L. M. Bragg et al. "Shining a Light on Dark Sequencing: Characterising Errors in Ion Torrent PGM Data". In: *PLoS Computational Biology* 9.4 (2013), e1003031. DOI: 10.1371/journal.pcbi.1003031 (cit. on p. 12).

[63] T. C. Glenn. "Field guide to next-generation DNA sequencers." In: *Molecular ecology resources* 11.5 (2011), pp. 759–69. DOI: 10.1111/j.1755-0998.2011.03024.x (cit. on pp. 12, 13).

[64] H. Breu. *A Theoretical Understanding of 2 Base Color Codes and Its Application to Annotation, Error Detection, and Error Correction*. Tech. rep. Applied Biosystems, 2010. URL: http://www3.appliedbiosystems.com/cms/groups/mcb_marketing/documents/generaldocuments/cms_058265.pdf (cit. on p. 12).

[65] A. Rhoads and K. F. Au. "PacBio Sequencing and Its Applications". In: *Genomics, Proteomics & Bioinformatics* 13.5 (2015), pp. 278–289. DOI: 10.1016/j.gpb.2015.08.002 (cit. on p. 13).

[66] L. Salmela and E. Rivals. "LoRDEC: accurate and efficient long read error correction". In: *Bioinformatics* 30.24 (2014), pp. 3506–3514. DOI: 10.1093/bioinformatics/btu538 (cit. on p. 13).

[67] J. Duda, W. Szpankowski, and A. Grama. "Fundamental Bounds and Approaches to Sequence Reconstruction from Nanopore Sequencers". In: *arXiv preprints* (2016). URL: http://arxiv.org/abs/1601.02420 (cit. on p. 13).

[68] B. Tan et al. "Next-generation sequencing (NGS) for assessment of microbial water quality: current progress, challenges, and future opportunities". In: *Frontiers in Microbiology* 6.SEP (2015). DOI: 10.3389/fmicb.2015.01027 (cit. on p. 13).

[69] J. Quick et al. "Real-time, portable genome sequencing for Ebola surveillance". In: *Nature* 530.7589 (2016), pp. 228–232. DOI: 10.1038/nature16996 (cit. on p. 13).

[70] A. B. R. McIntyre et al. "Nanopore sequencing in microgravity". In: *npj Microgravity* 2.June (2016), p. 16035. DOI: 10.1038/npjmgrav.2016.35 (cit. on p. 13).

[71] S. L. Castro-Wallace et al. "Nanopore DNA Sequencing and Genome Assembly on the International Space Station". In: *bioRxiv preprints* (2016). DOI: 10.1101/077651. URL: http://biorxiv.org/lookup/doi/10.1101/077651 (cit. on p. 13).

[72] M. Loose, S. Malla, and M. Stout. "Real-time selective sequencing using nanopore technology". In: *Nature Methods* 13.9 (2016), pp. 751–754. DOI: 10.1038/nmeth.3930 (cit. on p. 13).

[73] J. T. Simpson et al. "Detecting DNA cytosine methylation using nanopore sequencing". In: *Nature Methods* January (2017), pp. 1–7. DOI: 10.1038/nmeth.4184 (cit. on p. 13).

[74] M. Zhao, D. Liu, and H. Qu. "Systematic review of next-generation sequencing simulators: computational tools, features and perspectives". In: *Briefings in Functional Genomics* (2016), elw012. DOI: 10.1093/bfgp/elw012 (cit. on pp. 14, 15).

[75] M. Escalona, S. Rocha, and D. Posada. "A comparison of tools for the simulation of genomic next-generation sequencing data". In: *Nature Reviews Genetics* 17.8 (2016), pp. 459–469. DOI: 10.1038/nrg.2016.57 (cit. on pp. 14, 15).

[76] Y. Ono, K. Asai, and M. Hamada. "PBSIM: PacBio reads simulator - Toward accurate genome assembly". In: *Bioinformatics* 29.1 (2013), pp. 119–121. DOI: 10.1093/bioinformatics/bts649 (cit. on pp. 14, 15, 43).

[77] X. Hu et al. "pIRS: Profile-based illumina pair-end reads simulator". In: *Bioinformatics* 28.11 (2012), pp. 1533–1535. DOI: 10.1093/bioinformatics/bts187 (cit. on pp. 15, 43).

[78] M. Holtgrewe. *Mason – A Read Simulator for Second Generation Sequencing Data*. Tech. rep. October. Berlin: Institut für Mathematik und Informatik, Freie Universität Berlin, 2010, p. 18 (cit. on pp. 15, 16, 43).

[79] C. Bartenhagen and M. Dugas. "RSVSim: an R/Bioconductor package for the simulation of structural variations". In: *Bioinformatics* 29.13 (2013), pp. 1679–1681. DOI: 10.1093/bioinformatics/btt198 (cit. on p. 15).

[80] M. Qin et al. "SCNVSim: somatic copy number variation and structure variation simulator". In: *BMC Bioinformatics* 16.1 (2015), p. 66. DOI: 10.1186/s12859-015-0502-7 (cit. on p. 15).

[81] J. C. Mu et al. "VarSim: A high-fidelity simulation and validation framework for high-throughput genome sequencing with cancer applications". In: *Bioinformatics* 31.9 (2015), pp. 1469–1471. DOI: 10.1093/bioinformatics/btu828 (cit. on p. 15).

[82] W. Huang et al. "ART: A next-generation sequencing read simulator". In: *Bioinformatics* 28.4 (2012), pp. 593–594. DOI: 10.1093/bioinformatics/btr708 (cit. on pp. 15, 16, 43).

[83] M. Frampton and R. Houlston. "Generation of Artificial FASTQ Files to Evaluate the Performance of Next-Generation Sequencing Pipelines". In: *PLoS ONE* 7.11 (2012), e49110. DOI: 10.1371/journal.pone.0049110 (cit. on p. 15).

[84] A. Shcherbina. "FASTQSim: platform-independent data characterization and in silico read generation for NGS datasets." In: *BMC research notes* 7.1 (2014), p. 533. DOI: 10.1186/1756-0500-7-533 (cit. on pp. 15, 16, 43).

[85] Z. D. Stephens et al. "Simulating next-generation sequencing datasets from empirical mutation and sequencing models". In: *PLoS ONE* 11.11 (2016), pp. 1–18. DOI: 10.1371/journal.pone.0167047 (cit. on p. 15).

[86] Shifu Chen et al. "SeqMaker: A next generation sequencing simulator with variations, sequencing errors and amplification bias integrated". In: *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, 2016, pp. 835–840. DOI: 10.1109/BIBM.2016.7822634 (cit. on p. 15).

[87] R. Luo et al. "LRSim: a Linked Reads Simulator generating insights for better genome partitioning". In: (2015), pp. 1–3. DOI: 10.1101/103549. URL: http://biorxiv.org/content/early/2017/01/26/103549 (cit. on p. 15).

[88] K. E. McElroy, F. Luciani, and T. Thomas. "GemSIM: general, error-model based simulator of next-generation sequencing data". In: *BMC Genomics* 13.1 (2012), p. 74. DOI: 10.1186/1471-2164-13-74 (cit. on pp. 15, 16, 43).

[89] S. Pattnaik et al. "SInC: an accurate and fast error-model based simulator for SNPs, Indels and CNVs coupled with a read generator for short-read sequence data." In: *BMC bioinformatics* 15.1 (2014), p. 40. DOI: 10.1186/1471-2105-15-40 (cit. on pp. 15, 43).

[90] S. Kim, K. Jeong, and V. Bafna. "Wessim: a whole-exome sequencing simulator based on in silico exome capture". In: *Bioinformatics* 29.8 (2013), pp. 1076–1077. DOI: 10.1093/bioinformatics/btt074 (cit. on pp. 15, 16).

[91] S. Caboche et al. "Comparison of mapping algorithms used in high-throughput sequencing: application to Ion Torrent data." In: *BMC genomics* 15 (2014), p. 264. DOI: 10.1186/1471-2164-15-264 (cit. on pp. 15, 43, 44).

[92] C. Yang et al. "NanoSim: nanopore sequence read simulator based on statistical characterization". In: *GigaScience* (2017). DOI: 10.1093/gigascience/gix010 (cit. on p. 15).

[93] H. Lee, J. Gurtowski, and S. Yoo. "Error correction and assembly complexity of single molecule sequencing reads". In: *bioRxiv* (2014), pp. 1–17. DOI: 10.1101/006395 (cit. on p. 15).

[94] E. A. G. Baker et al. "SiLiCO: A Simulator of Long Read Sequencing in PacBio and Oxford Nanopore". In: *bioRxiv preprints* (2016), pp. 1–3. DOI: 10.1101/076901. URL: http://biorxiv.org/lookup/doi/10.1101/076901 (cit. on p. 15).

[95] B. Lau et al. "LongISLND: in silico sequencing of lengthy and noisy datatypes". In: *Bioinformatics* 32.24 (2016), pp. 3829–3832. DOI: 10.1093/bioinformatics/btw602 (cit. on p. 15).

[96] J. Marić. "Long Read RNA-seq Mapper". PhD thesis. University of Zagreb, 2015 (cit. on pp. 15, 26).

[97] B. K. Stöcker, J. Köster, and S. Rahmann. "SimLoRD: Simulation of Long Read Data". In: *Bioinformatics* 32.17 (2016), pp. 2704–2706. DOI: 10.1093/bioinformatics/btw286 (cit. on p. 15).

[98] F. Lysholm, B. Andersson, and B. Persson. "An efficient simulator of 454 data using configurable statistical models". In: *BMC Research Notes* 4.1 (2011), p. 449. DOI: 10.1186/1756-0500-4-449 (cit. on p. 15).

[99] S. Balzer et al. "Characteristics of 454 pyrosequencing data–enabling realistic simulation with flowsim". In: *Bioinformatics* 27.15 (2011), pp. 2171–2171. DOI: 10.1093/bioinformatics/btr384 (cit. on pp. 15, 43).

[100] S. Johnson et al. "A better sequence-read simulator program for metagenomics." In: *BMC bioinformatics* 15 Suppl 9.Suppl 9 (2014), S14. DOI: 10.1186/1471-2105-15-S9-S14 (cit. on p. 16).

[101] F. E. Angly et al. "Grinder: a versatile amplicon and shotgun sequence simulator". In: *Nucleic Acids Research* 40.12 (2012), e94–e94. DOI: 10.1093/nar/gks251 (cit. on p. 16).

[102] D. C. Richter et al. “MetaSim—A Sequencing Simulator for Genomics and Metagenomics”. In: *PLoS ONE* 3.10 (2008), e3373. DOI: 10.1371/journal.pone.0003373 (cit. on p. 16).

[103] B. Jia et al. “NeSSM: A Next-Generation Sequencing Simulator for Metagenomics”. In: *PLoS ONE* 8.10 (2013), e75448. DOI: 10.1371/journal.pone.0075448 (cit. on p. 16).

[104] G. R. Grant et al. “Comparative analysis of RNA-Seq alignment algorithms and the RNA-Seq unified mapper (RUM)”. In: *Bioinformatics* 27.18 (2011), pp. 2518–2528. DOI: 10.1093/bioinformatics/btr427 (cit. on pp. 16, 26).

[105] T. Griebel et al. “Modelling and simulating generic RNA-Seq experiments with the flux simulator”. In: *Nucleic Acids Research* 40.20 (2012), pp. 10073–10083. DOI: 10.1093/nar/gks666 (cit. on p. 16).

[106] S.-m. Lee et al. “PSIM: pattern-based read simulator for RNA-seq analysis”. In: *Multimedia Tools and Applications* 74.16 (2015), pp. 6465–6480. DOI: 10.1007/s11042-014-2108-x (cit. on p. 16).

[107] B. Sipos et al. “Realistic simulations reveal extensive sample-specificity of RNA-seq biases”. In: *arXiv preprints* (2013), pp. 1–8. URL: http://arxiv.org/abs/1308.3172 (cit. on p. 16).

[108] B. Li et al. “RNA-Seq gene expression estimation with read mapping uncertainty”. In: *Bioinformatics* 26.4 (2010), pp. 493–500. DOI: 10.1093/bioinformatics/btp692 (cit. on p. 16).

[109] S. Benidt and D. Nettleton. “SimSeq: a nonparametric approach to simulation of RNA-sequence datasets”. In: *Bioinformatics* 31.13 (2015), pp. 2131–2140. DOI: 10.1093/bioinformatics/btv124 (cit. on p. 16).

[110] M. C. Frith, R. Mori, and K. Asai. “A mostly traditional approach improves alignment of bisulfite-converted DNA.” In: *Nucleic acids research* 40.13 (2012), e100. DOI: 10.1093/nar/gks275 (cit. on pp. 16, 26, 43).

[111] O. J. L. Rackham et al. “WGBSSuite: simulating whole-genome bisulphite sequencing data and benchmarking differential DNA methylation analysis tools”. In: *Bioinformatics* 31.14 (2015), pp. 2371–2373. DOI: 10.1093/bioinformatics/btv114 (cit. on p. 16).

[112] G. Renaud et al. “gargammel: a sequence simulator for ancient DNA”. In: *Bioinformatics* (2016), btw670. DOI: 10.1093/bioinformatics/btw670 (cit. on p. 16).

[113] D. Pratas, A. J. Pinho, and J. M. O. S. Rodrigues. “XS: a FASTQ read simulator.” In: *BMC research notes* 7 (2014), p. 40. DOI: 10.1186/1756-0500-7-40 (cit. on pp. 16, 43).

[114] M. L. Engle and C. Burks. “GenFrag 2.1: new features for more robust fragment assembly benchmarks”. In: *Bioinformatics* 10.5 (1994), pp. 567–568. DOI: 10.1093/bioinformatics/10.5.567 (cit. on p. 16).

[115] G. Myers. “A dataset generator for whole genome shotgun sequencing.” In: *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology* (1999), pp. 202–10 (cit. on p. 16).

[116] C. G. Hur et al. "FASIM: Fragments assembly simulation using biased-sampling model and assembly simulation for microbial genome shotgun sequencing". In: *Journal of Microbiology and Biotechnology* 16.5 (2006), pp. 683–688 (cit. on p. 16).

[117] S. S. Mande, M. H. Mohammed, and T. S. Ghosh. "Classification of metagenomic sequences: methods and challenges". In: *Briefings in Bioinformatics* 13.6 (2012), pp. 669–681. DOI: 10.1093/bib/bbs054 (cit. on pp. 17, 73, 74).

[118] M. Pop. "Genome assembly reborn: recent computational challenges". In: *Briefings in Bioinformatics* 10.4 (2009), pp. 354–366. DOI: 10.1093/bib/bbp026 (cit. on pp. 17, 65).

[119] Q. D. Atkinson and R. D. Gray. "Curious parallels and curious connections–phylogenetic thinking in biology and historical linguistics." In: *Systematic biology* 54.4 (2005), pp. 513–26. DOI: 10.1080/10635150590950317 (cit. on p. 17).

[120] A. Kitchen et al. "Bayesian phylogenetic analysis of Semitic languages identifies an Early Bronze Age origin of Semitic in the Near East". In: *Proceedings of the Royal Society B: Biological Sciences* 276.1668 (2009), pp. 2703–2710. DOI: 10.1098/rspb.2009.0408 (cit. on p. 17).

[121] A. C. Barbrook et al. "The phylogeny of The Canterbury Tales". In: *Nature* 394.6696 (1998), pp. 839–839. DOI: 10.1038/29667 (cit. on p. 17).

[122] J. J. Tehrani. "The phylogeny of Little Red Riding Hood." In: *PloS one* 8.11 (2013), e78871. DOI: 10.1371/journal.pone.0078871 (cit. on p. 17).

[123] A. Z. Broder. "Identifying and Filtering Near-Duplicate Documents". In: *Combinatorial Pattern Matching: 11th Annual Symposium, CPM 2000 Montreal, Canada, June 21–23, 2000 Proceedings*. 2000, pp. 1–10. DOI: 10.1007/3-540-45123-4_1 (cit. on pp. 17, 28, 79).

[124] R. Durbin et al. *Biological Sequence Analysis*. Cambridge: Cambridge University Press, 1998, p. 366. DOI: 10.1017/CBO9780511790492 (cit. on p. 18).

[125] H. Li et al. "The Sequence Alignment/Map format and SAMtools." In: *Bioinformatics* 25.16 (2009), pp. 2078–9. DOI: 10.1093/bioinformatics/btp352 (cit. on pp. 18, 22, 57, 64, 100, 106, 133–135).

[126] S. Henikoff and J. G. Henikoff. "Amino acid substitution matrices from protein blocks." In: *Proceedings of the National Academy of Sciences* 89.22 (1992), pp. 10915–10919. DOI: 10.1073/pnas.89.22.10915 (cit. on p. 18).

[127] S. F. Altschul. "Amino acid substitution matrices from an information theoretic perspective". In: *Journal of Molecular Biology* 219.3 (1991), pp. 555–565. DOI: 10.1016/0022-2836(91)90193-A (cit. on p. 18).

[128] T. Muller, S. Rahmann, and M. Rehmsmeier. "Non-symmetric score matrices and the detection of homologous transmembrane proteins". In: *Bioinformatics* 17.Suppl 1 (2001), S182–S189. DOI: 10.1093/bioinformatics/17.suppl_1.S182 (cit. on p. 18).

[129] S. B. Needleman and C. D. Wunsch. "A general method applicable to the search for similarities in the amino acid sequence of two proteins". In: *Journal of Molecular Biology* 48.3 (1970), pp. 443–453. DOI: 10.1016/0022-2836(70)90057-4 (cit. on pp. 19, 21, 121).

[130] T. F. Smith and M. S. Waterman. "Identification of common molecular subsequences." In: *Journal of molecular biology* 147.1 (1981), pp. 195–7. DOI: 10.1016/0022-2836(81)90087-5 (cit. on pp. 19, 21, 121).

[131] O. Gotoh. "An improved algorithm for matching biological sequences". In: *Journal of Molecular Biology* 162.3 (1982), pp. 705–708. DOI: 10.1016/0022-2836(82)90398-9 (cit. on p. 19).

[132] M. Farrar. "Striped Smith-Waterman speeds database searches six times over other SIMD implementations". In: *Bioinformatics* 23.2 (2007), pp. 156–161. DOI: 10.1093/bioinformatics/btl582 (cit. on p. 19).

[133] G. Myers. "A fast bit-vector algorithm for approximate string matching based on dynamic programming". In: *Journal of the ACM* 46.3 (1999), pp. 395–415. DOI: 10.1145/316542.316550 (cit. on p. 19).

[134] K. Fredriksson. "Row-wise Tiling for the Myers' Bit-Parallel Approximate String Matching Algorithm". In: 2003, pp. 66–79. DOI: 10.1007/978-3-540-39984-1_6 (cit. on p. 19).

[135] K. Kimura, A. Koike, and K. Nakai. "A bit-parallel dynamic programming algorithm suitable for DNA sequence alignment." In: *Journal of bioinformatics and computational biology* 10.4 (2012), p. 1250002. DOI: 10.1142/S0219720012500023 (cit. on p. 19).

[136] H. Xin et al. "Shifted Hamming distance: A fast and accurate SIMD-friendly filter to accelerate alignment verification in read mapping". In: *Bioinformatics* 31.10 (2014), pp. 1553–1560. DOI: 10.1093/bioinformatics/btu856 (cit. on p. 19).

[137] S. Wu and U. Mamber. "Agrep - a fast approximate pattern matching tool". In: *Proceedings of the Winter 1992 USENIX Conference San Francisco USA. Berkeley.* 1992, pp. 153–162 (cit. on p. 19).

[138] H. Hyyrö. "A Bit-Vector Algorithm for Computing Levenshtein and Damerau Edit Distances". In: *Nordic Journal of Computing* 10.1 (2003) (cit. on p. 19).

[139] J. Loving, Y. Hernandez, and G. Benson. "BitPAl: a bit-parallel, general integer-scoring sequence alignment algorithm". In: *Bioinformatics* 30.22 (2014), pp. 3166–3173. DOI: 10.1093/bioinformatics/btu507 (cit. on p. 19).

[140] K. Hou, H. Wang, and W.-c. Feng. "AAlign: A SIMD Framework for Pairwise Sequence Alignment on x86-Based Multi-and Many-Core Processors". In: *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. May. IEEE, 2016, pp. 780–789. DOI: 10.1109/IPDPS.2016.115 (cit. on p. 19).

[141] S. Sheetlin et al. "ALP & FALP: C++ libraries for pairwise local alignment E -values". In: *Bioinformatics* 32.2 (2015), btv575. DOI: 10.1093/bioinformatics/btv575 (cit. on p. 19).

[142] M. Šošić and M. Šikić. "Edlib: a C/C++ library for fast, exact sequence alignment using edit distance". In: *Bioinformatics* (2017), p. 070649. DOI: 10.1093/bioinformatics/btw753 (cit. on p. 19).

[143] J. Daily. "Parasail: SIMD C library for global, semi-global, and local pairwise sequence alignments". In: *BMC Bioinformatics* 17.1 (2016), p. 81. DOI: 10.1186/s12859-016-0930-z (cit. on p. 19).

[144] A. Döring et al. "SeqAn an efficient, generic C++ library for sequence analysis." In: *BMC bioinformatics* 9 (2008), p. 11. DOI: 10.1186/1471-2105-9-11 (cit. on pp. 19, 35).

[145] M. Zhao et al. "SSW library: an SIMD Smith-Waterman C/C++ library for use in genomic applications." In: *PloS one* 8.12 (2013), e82138. DOI: 10.1371/journal.pone.0082138 (cit. on p. 19).

[146] A. Szalkowski et al. "SWPS3 – fast multi-threaded vectorized Smith-Waterman for IBM Cell/B.E. and ×86/SSE2". In: *BMC Research Notes* 1.1 (2008), p. 107. DOI: 10.1186/1756-0500-1-107 (cit. on p. 19).

[147] Y. Liu, D. L. Maskell, and B. Schmidt. "CUDASW++: optimizing Smith-Waterman sequence database searches for CUDA-enabled graphics processing units". In: *BMC Research Notes* 2.1 (2009), p. 73. DOI: 10.1186/1756-0500-2-73 (cit. on p. 19).

[148] Y. Liu, B. Schmidt, and D. L. Maskell. "CUDASW++2.0: enhanced Smith-Waterman protein database search on CUDA-enabled GPUs based on SIMT and virtualized SIMD abstractions". In: *BMC Research Notes* 3.1 (2010), p. 93. DOI: 10.1186/1756-0500-3-93 (cit. on p. 19).

[149] Y. Liu, A. Wirawan, and B. Schmidt. "CUDASW++ 3.0: accelerating Smith-Waterman protein database search by coupling CPU and GPU SIMD instructions". In: *BMC Bioinformatics* 14.1 (2013), p. 117. DOI: 10.1186/1471-2105-14-117 (cit. on p. 19).

[150] Y. Liu and B. Schmidt. "GSWABE: faster GPU-accelerated sequence alignment with optimal alignment retrieval for short DNA sequences". In: *Concurrency and Computation: Practice and Experience* 27.4 (2015), pp. 958–972. DOI: 10.1002/cpe.3371 (cit. on p. 19).

[151] E. F. De O. Sandes et al. "MASA: A Multiplatform Architecture for Sequence Aligners with Block Pruning". In: *ACM Transactions on Parallel Computing* 2.4 (2016), pp. 1–31. DOI: 10.1145/2858656 (cit. on p. 19).

[152] M. Korpar and M. Sikic. "SW#-GPU-enabled exact alignments on genome scale". In: *Bioinformatics* 29.19 (2013), pp. 2494–2495. DOI: 10.1093/bioinformatics/btt410 (cit. on p. 19).

[153] S. a. Manavski and G. Valle. "CUDA compatible GPU cards as efficient hardware accelerators for Smith-Waterman sequence alignment". In: *BMC Bioinformatics* 9.Suppl 2 (2008), S10. DOI: 10.1186/1471-2105-9-S2-S10 (cit. on p. 19).

[154] Y. Liu and B. Schmidt. "SWAPHI: Smith-waterman protein database search on Xeon Phi coprocessors". In: *2014 IEEE 25th International Conference on Application-Specific Systems, Architectures and Processors*. IEEE, 2014, pp. 184–185. DOI: 10.1109/ASAP.2014.6868657 (cit. on p. 19).

[155] Y. Liu et al. "SWAPHI-LS: Smith-Waterman Algorithm on Xeon Phi coprocessors for Long DNA Sequences". In: *2014 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2014, pp. 257–265. DOI: 10.1109/CLUSTER.2014.6968772 (cit. on p. 19).

[156] A. Chacón et al. "Thread-cooperative, bit-parallel computation of levenshtein distance on GPU". In: *Proceedings of the 28th ACM international conference on Supercomputing - ICS '14*. New York, New York, USA: ACM Press, 2014, pp. 103–112. DOI: 10.1145/2597652.2597677 (cit. on p. 19).

[157] H. Li et al. “A fast CUDA implementation of agrep algorithm for approximate nucleotide sequence matching”. In: *Proceedings of the 2011 IEEE 9th Symposium on Application Specific Processors, SASP 2011* (2011), pp. 74–77. DOI: 10.1109/SASP.2011.5941082 (cit. on p. 19).

[158] T. T. Tran, M. Giraud, and J. S. Varré. “Bit-parallel multiple pattern matching”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 7204 LNCS.PART 2 (2012), pp. 292–301. DOI: 10.1007/978-3-642-31500-8_30 (cit. on p. 19).

[159] T. T. Tran, Y. Liu, and B. Schmidt. “Bit-parallel approximate pattern matching: Kepler GPU versus Xeon Phi”. In: *Parallel Computing* 54 (2016), pp. 128–138. DOI: 10.1016/j.parco.2015.11.001 (cit. on p. 19).

[160] D. Lipman and W. Pearson. “Rapid and sensitive protein similarity searches”. In: *Science* 227.4693 (1985), pp. 1435–1441. DOI: 10.1126/science.2983426 (cit. on p. 20).

[161] W. R. Pearson and D. J. Lipman. “Improved tools for biological sequence comparison.” In: *Proceedings of the National Academy of Sciences of the United States of America* 85.8 (1988), pp. 2444–8. DOI: 10.1073/pnas.85.8.2444 (cit. on pp. 20, 21).

[162] W. J. Wilbur and D. J. Lipman. “Rapid similarity searches of nucleic acid and protein data banks.” In: *Proceedings of the National Academy of Sciences* 80.3 (1983), pp. 726–730. DOI: 10.1073/pnas.80.3.726 (cit. on p. 20).

[163] S. F. Altschul et al. “Basic local alignment search tool”. In: *Journal of Molecular Biology* 215.3 (1990), pp. 403–410. DOI: 10.1016/S0022-2836(05)80360-2 (cit. on pp. 20, 21, 26, 78).

[164] S. Altschul. “Gapped BLAST and PSI-BLAST: a new generation of protein database search programs”. In: *Nucleic Acids Research* 25.17 (1997), pp. 3389–3402. DOI: 10.1093/nar/25.17.3389 (cit. on pp. 20, 73).

[165] A. Morgulis et al. “Database indexing for production MegaBLAST searches”. In: *Bioinformatics* 24.16 (2008), pp. 1757–1764. DOI: 10.1093/bioinformatics/btn322 (cit. on p. 20).

[166] Z. Zhang et al. “A greedy algorithm for aligning DNA sequences.” In: *Journal of computational biology : a journal of computational molecular cell biology* 7.1-2 (2000), pp. 203–14. DOI: 10.1089/10665270050081478 (cit. on p. 20).

[167] A. L. Delcher et al. “Alignment of whole genomes”. In: *Nucleic Acids Research* 27.11 (1999), pp. 2369–2376. DOI: 10.1093/nar/27.11.2369 (cit. on p. 20).

[168] A. L. Delcher et al. “Fast algorithms for large-scale genome alignment and comparison.” In: *Nucleic acids research* 30.11 (2002), pp. 2478–2483. DOI: 10.1093/nar/30.11.2478 (cit. on p. 20).

[169] N. Bray, I. Dubchak, and L. Pachter. “AVID: A global alignment program.” In: *Genome research* 13.1 (2003), pp. 97–102. DOI: 10.1101/gr.789803 (cit. on p. 20).

[170] S. Schwartz et al. “Human – Mouse Alignments with BLASTZ”. In: *Genome Research* 13.1 (2003), pp. 103–107. DOI: 10.1101/gr.809403. (cit. on pp. 20, 73).

[171] M. Brudno et al. "LAGAN and Multi-LAGAN: Efficient Tools for Large-Scale Multiple Alignment of Genomic DNA". In: *Genome Research* 13.4 (2003), pp. 721–731. DOI: 10.1101/gr.926603 (cit. on p. 20).

[172] S. Kurtz et al. "Versatile and open software for comparing large genomes." In: *Genome biology* 5.2 (2004), R12. DOI: 10.1186/gb-2004-5-2-r12 (cit. on pp. 20, 34, 76).

[173] R. S. Harris. "Improved pairwise alignment of genomic DNA". PhD thesis. Pennsylvania State University, 2007 (cit. on p. 20).

[174] R. Nakato and O. Gotoh. "Cgaln: fast and space-efficient whole-genome alignment". In: *BMC Bioinformatics* 11.1 (2010), p. 224. DOI: 10.1186/1471-2105-11-224 (cit. on p. 20).

[175] A. K. Hudek and D. G. Brown. "FEAST: Sensitive local alignment with multiple rates of evolution". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 8.3 (2011), pp. 698–709. DOI: 10.1109/TCBB.2010.76 (cit. on p. 20).

[176] S. M. Kiełbasa et al. "Adaptive seeds tame genomic sequence comparison." In: *Genome research* 21.3 (2011), pp. 487–93. DOI: 10.1101/gr.113985.110 (cit. on pp. 20, 24, 25, 30, 31, 42).

[177] M. C. Frith and R. Kawaguchi. "Split-alignment of genomes finds orthologies more accurately." In: *Genome biology* 16.1 (2015), p. 106. DOI: 10.1186/s13059-015-0670-9 (cit. on p. 20).

[178] W. J. Kent. "BLAT—The BLAST-Like Alignment Tool". In: *Genome Research* 12.4 (2002), pp. 656–664. DOI: 10.1101/gr.229202 (cit. on pp. 21, 26, 73).

[179] H. Li, J. Ruan, and R. Durbin. "Mapping short DNA sequencing reads and calling variants using mapping quality scores." In: *Genome research* 18.11 (2008), pp. 1851–1858. DOI: 10.1101/gr.078212.108 (cit. on pp. 21, 24, 26, 27, 29, 49).

[180] A. Dobin et al. "STAR: Ultrafast universal RNA-seq aligner". In: *Bioinformatics* 29.1 (2013), pp. 15–21. DOI: 10.1093/bioinformatics/bts635 (cit. on pp. 21, 26).

[181] H. Li and R. Durbin. "Fast and accurate short read alignment with Burrows-Wheeler transform". In: *Bioinformatics* 25.14 (2009), pp. 1754–1760. DOI: 10.1093/bioinformatics/btp324 (cit. on pp. 21, 22, 24, 25, 27, 29, 34, 35, 53, 74, 78, 101, 104).

[182] B. Langmead et al. "Ultrafast and memory-efficient alignment of short DNA sequences to the human genome." In: *Genome biology* 10.3 (2009), R25. DOI: 10.1186/gb-2009-10-3-r25 (cit. on pp. 21, 22, 25, 27, 29, 35, 74, 101).

[183] H. Li. "Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM". In: *arXiv preprints* (2013), p. 3. URL: http://arxiv.org/abs/1303.3997 (cit. on pp. 21, 22, 24, 25, 42, 46, 47, 53, 57).

[184] B. Langmead and S. L. Salzberg. "Fast gapped-read alignment with Bowtie 2." In: *Nature methods* 9.4 (2012), pp. 357–9. DOI: 10.1038/nmeth.1923 (cit. on pp. 21, 22, 24, 25, 42, 47, 53, 57, 76, 77, 79).

[185] P. Muir et al. “The real cost of sequencing: scaling computation to keep pace with data generation”. In: *Genome Biology* 17.1 (2016), p. 53. DOI: 10.1186/s13059-016-0917-0 (cit. on p. 21).

[186] N. A. Fonseca et al. “Tools for mapping high-throughput sequencing data”. In: *Bioinformatics* 28.24 (2012), pp. 3169–3177. DOI: 10.1093/bioinformatics/bts605 (cit. on p. 22).

[187] *International Consortium Completes Human Genome Project*. Tech. rep. 2003. DOI: 10.1517/phgs.4.3.241.22688 (cit. on p. 20).

[188] H. Li and R. Durbin. “Fast and accurate long-read alignment with Burrows-Wheeler transform”. In: *Bioinformatics* 26.5 (2010), pp. 589–595. DOI: 10.1093/bioinformatics/btp698 (cit. on pp. 22, 25, 47, 53).

[189] T. J. Treangen and S. L. Salzberg. “Repetitive DNA and next-generation sequencing: computational challenges and solutions”. In: *Nature Reviews Genetics* 13.1 (2011), pp. 36–46. DOI: 10.1038/nrg3117 (cit. on p. 22).

[190] M. David et al. “SHRiMP2: Sensitive yet Practical Short Read Mapping”. In: *Bioinformatics* 27.7 (2011), pp. 1011–1012. DOI: 10.1093/bioinformatics/btr046 (cit. on pp. 23, 24, 26, 27, 31, 53).

[191] N. Homer, B. Merriman, and S. F. Nelson. “BFAST: An Alignment Tool for Large Scale Genome Resequencing”. In: *PLoS ONE* 4.11 (2009), e7767. DOI: 10.1371/journal.pone.0007767 (cit. on pp. 23, 26, 27, 31, 53).

[192] G. Navarro and V. Mäkinen. “Compressed full-text indexes”. In: *ACM Computing Surveys* 39.1 (2007), 2–es. DOI: 10.1145/1216370.1216372 (cit. on p. 24).

[193] P. Ferragina and G. Manzini. “Opportunistic data structures with applications”. In: *Proceedings 41st Annual Symposium on Foundations of Computer Science*. IEEE Comput. Soc, 2000, pp. 390–398. DOI: 10.1109/SFCS.2000.892127 (cit. on pp. 24, 25, 35, 101, 102, 104).

[194] S. Marco-Sola et al. “The GEM mapper: fast, accurate and versatile alignment by filtration”. In: *Nature Methods* 9.12 (2012), pp. 1185–1188. DOI: 10.1038/nmeth.2221 (cit. on pp. 24, 25, 42, 53).

[195] D. G. Brown. “A survey of seeding for sequence alignment”. In: *Bioinformatics Algorithms: Techniques and Applications*. 2008 (cit. on pp. 24, 81, 82).

[196] D. G. Brown, M. Li, and B. Ma. “A tutorial of recent developments in the seeding of local alignment.” In: *Journal of bioinformatics and computational biology* 2.4 (2004), pp. 819–42. DOI: 10.1142/S0219720004000983 (cit. on pp. 24, 30).

[197] H. Xin et al. “Optimal seed solver: optimizing seed selection in read mapping”. In: *Bioinformatics* 32.11 (2016), pp. 1632–1642. DOI: 10.1093/bioinformatics/btv670 (cit. on p. 24).

[198] I. Sović et al. “Fast and sensitive mapping of nanopore sequencing reads with GraphMap”. In: *Nature Communications* 7 (2016), p. 11307. DOI: 10.1038/ncomms11307 (cit. on pp. 24, 25, 31).

[199] R. Rabatin, B. Brejová, and T. Vinař. “Using Sequence Ensembles for Seeding Alignments of MinION Sequencing Data”. In: *arXiv preprints* (2016), pp. 1–10. URL: http://arxiv.org/abs/1606.08719 (cit. on p. 24).

[200] H. Li and N. Homer. "A survey of sequence alignment algorithms for next-generation sequencing". In: *Briefings in Bioinformatics* 11.5 (2010), pp. 473–483. DOI: 10.1093/bib/bbq015 (cit. on pp. 24, 122).

[201] P. Ribeca. "Short-Read Mapping". In: *Bioinformatics for High Throughput Sequencing*. New York, NY: Springer New York, 2012, pp. 107–125. DOI: 10.1007/978-1-4614-0782-9_7 (cit. on pp. 24, 41).

[202] K. Reinert et al. "Alignment of Next-Generation Sequencing Reads". In: *Annual Review of Genomics and Human Genetics* 16.1 (2015), p. 150504161622003. DOI: 10.1146/annurev-genom-090413-025358 (cit. on pp. 24, 41).

[203] T. W. Lam et al. "Compressed indexing and local alignment of DNA". In: *Bioinformatics* 24.6 (2008), pp. 791–797. DOI: 10.1093/bioinformatics/btn032 (cit. on pp. 25, 35).

[204] L. J. Quan. "Accurate alignment of sequencing reads from various genomic origins". PhD thesis. National University of Singapore, 2014 (cit. on p. 25).

[205] B. Liu, Y. Gao, and Y. Wang. "LAMSA: fast split read alignment with long approximate matches". In: *Bioinformatics* 2 (2016), btw594. DOI: 10.1093/bioinformatics/btw594 (cit. on p. 25).

[206] P. A. Andrews et al. "MUMdex: MUM-based structural variation detection". In: *bioRxiv preprints* (2016). DOI: 10.1101/078261. URL: http://biorxiv.org/content/biorxiv/early/2016/09/30/078261.full.pdf (cit. on p. 25).

[207] N. Prezza et al. "Fast, accurate, and lightweight analysis of BS-treated reads with ERNE 2". In: *BMC Bioinformatics* 17.S4 (2016), p. 69. DOI: 10.1186/s12859-016-0910-3 (cit. on pp. 25, 26).

[208] E. V. i Zorita. *Statistical models for genome sequence mapping*. Tech. rep. Master thesis. 2016. URL: https://upcommons.upc.edu/bitstream/handle/2117/97991/thesis_EE_ezorita.pdf (cit. on p. 25).

[209] M. Vyverman et al. "A Long Fragment Aligner called ALFALFA". In: *BMC Bioinformatics* 16.1 (2015), p. 159. DOI: 10.1186/s12859-015-0533-0 (cit. on p. 25).

[210] J. Salavert et al. "Fast inexact mapping using advanced tree exploration on backward search methods". In: *BMC Bioinformatics* 16.1 (2015), p. 18. DOI: 10.1186/s12859-014-0438-3 (cit. on pp. 25, 28).

[211] E. Siragusa. "Approximate string matching for high-throughput sequencing". PhD thesis. Free University, 2015 (cit. on pp. 25, 47, 79).

[212] M. Gholami et al. "ARYANA: Aligning Reads by Yet Another Approach". In: *BMC Bioinformatics* 15.Suppl 9 (2014), S12. DOI: 10.1186/1471-2105-15-S9-S12 (cit. on p. 25).

[213] P. Kerpedjiev et al. "Adaptable probabilistic mapping of short reads using position specific scoring matrices". In: *BMC Bioinformatics* 15.1 (2014), p. 100. DOI: 10.1186/1471-2105-15-100 (cit. on p. 25).

[214] Y. Liu, B. Popp, and B. Schmidt. "CUSHAW3: Sensitive and Accurate Base-Space and Color-Space Short-Read Alignment with Hybrid Seeding." In: *PloS one* 9.1 (2014), e86869. DOI: 10.1371/journal.pone.0086869 (cit. on pp. 25, 27).

[215] J. González-Domínguez, Y. Liu, and B. Schmidt. "Parallel and Scalable Short-Read Alignment on Multi-Core Clusters Using UPC++". In: *PLOS ONE* 11.1 (2016), e0145490. DOI: 10.1371/journal.pone.0145490 (cit. on pp. 25, 27).

[216] J. Tarraga et al. "Acceleration of short and long DNA read mapping without loss of accuracy using suffix array". In: *Bioinformatics* 30.23 (2014), pp. 3396–3398. DOI: 10.1093/bioinformatics/btu553 (cit. on p. 25).

[217] N. S. Vo et al. "RandAL: a randomized approach to aligning DNA sequences to reference genomes". In: *BMC Genomics* 15.Suppl 5 (2014), S2. DOI: 10.1186/1471-2164-15-S5-S2 (cit. on p. 25).

[218] C. Raczy et al. "Isaac: ultra-fast whole-genome secondary analysis on Illumina sequencing platforms". In: *Bioinformatics* 29.16 (2013), pp. 2041–2043. DOI: 10.1093/bioinformatics/btt314 (cit. on p. 25).

[219] E. Siragusa, D. Weese, and K. Reinert. "Fast and accurate read mapping with approximate seeds and multiple backtracking". In: *Nucleic Acids Research* 41.7 (2013), e78–e78. DOI: 10.1093/nar/gkt005 (cit. on p. 25).

[220] J. Dufourt et al. "NucBase, an easy to use read mapper for small RNAs". In: *Mobile DNA* 4.1 (2013), p. 1. DOI: 10.1186/1759-8753-4-1 (cit. on p. 25).

[221] C. Tennakoon, R. W. Purbojati, and W. K. Sung. "BatMis: A fast algorithm for k-mismatch mapping". In: *Bioinformatics* 28.16 (2012), pp. 2122–2128. DOI: 10.1093/bioinformatics/bts339 (cit. on p. 25).

[222] M. J. Chaisson and G. Tesler. "Mapping single molecule sequencing reads using basic local alignment with successive refinement (BLASR): application and theory." In: *BMC bioinformatics* 13 (2012), p. 238. DOI: 10.1186/1471-2105-13-238 (cit. on p. 25).

[223] G. G. Faust and I. M. Hall. "GEM: crystal-clear DNA alignment". In: *Nature Methods* 9.12 (2012), pp. 1159–1160. DOI: 10.1038/nmeth.2256 (cit. on pp. 25, 74).

[224] S. Marco-Sola and P. Ribeca. "Efficient Alignment of Illumina-Like High-Throughput Sequencing Reads with the GEnomic Multi-tool (GEM) Mapper". In: *Current Protocols in Bioinformatics*. Vol. 2015. June. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2015, pp. 11.13.1–11.13.20. DOI: 10.1002/0471250953.bi1113s50 (cit. on p. 25).

[225] Y. Liu and B. Schmidt. "Long read alignment based on maximal exact match seeds." In: *Bioinformatics* 28.18 (2012), pp. i318–i324. DOI: 10.1093/bioinformatics/bts414 (cit. on p. 25).

[226] R. Li et al. "SOAP2: an improved ultrafast tool for short read alignment". In: *Bioinformatics* 25.15 (2009), pp. 1966–1967. DOI: 10.1093/bioinformatics/btp336 (cit. on pp. 25, 101).

[227] C. Meek, J. M. Patel, and S. Kasetty. "OASIS: an online and accurate technique for local-alignment searches on biological sequences". In: *Proceedings of the 29th international conference on Very large data bases - Volume 29* (2003), pp. 910–921 (cit. on p. 25).

[228] *White paper on CLC read mapper*. Tech. rep. CLC bio, 2012. URL: http://www.clcbio.com/files/whitepapers/whitepaper-on-CLC-read-mapper.pdf (cit. on p. 25).

[229] H. R. Johnston et al. "PEMapper and PECaller provide a simplified approach to whole-genome sequencing." In: *Proceedings of the National Academy of Sciences of the United States of America* (2017), p. 201618065. DOI: 10.1073/pnas.1618065114 (cit. on pp. 25, 26).

[230] S. Liu, Y. Wang, and F. Wang. "A Fast Read Alignment Method based on seed-and-vote for Next Generation Sequencing". In: *BMC Bioinformatics* 17.220 (2016). DOI: 10.1186/s12859-016-1329-6 (cit. on p. 25).

[231] J. Kim, C. Li, and X. Xie. "Hobbes3: Dynamic generation of variable-length signatures for efficient approximate subsequence mappings". In: *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*. IEEE, 2016, pp. 169–180. DOI: 10.1109/ICDE.2016.7498238 (cit. on p. 25).

[232] C.-L. Xiao et al. "MECAT: an ultra-fast mapping, error correction and de novo assembly tool for single-molecule sequencing reads". In: *bioRxiv preprints* (2016), pp. 1–32. DOI: 10.1101/089250 (cit. on p. 25).

[233] M. R. Amin, S. Skiena, and M. C. Schatz. "NanoBLASTer: Fast alignment and characterization of Oxford Nanopore single molecule sequencing reads". In: *2016 IEEE 6th International Conference on Computational Advances in Bio and Medical Sciences (ICCABS)* (2016), pp. 1–6. DOI: 10.1109/ICCABS.2016.7802776 (cit. on p. 25).

[234] *NextGENe. Next Generation Sequencing Software for Biologists. User's Manual.* Tech. rep. 2016. URL: http://www.softgenetics.com/PDF/NextGENe_UsersManual_web.pdf (cit. on p. 25).

[235] N. H. Tran and X. Chen. "AMAS: optimizing the partition and filtration of adaptive seeds to speed up read mapping". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* (2015), pp. 1–1. DOI: 10.1109/TCBB.2015.2465900 (cit. on p. 25).

[236] H. Cheng et al. "BitMapper: an efficient all-mapper based on bit-vector computing". In: *BMC Bioinformatics* 16.1 (2015), p. 192. DOI: 10.1186/s12859-015-0626-9 (cit. on p. 25).

[237] B. Wolf et al. "DNAseq Workflow in a Diagnostic Context and an Example of a User Friendly Implementation". In: *BioMed Research International* 2015 (2015), pp. 1–11. DOI: 10.1155/2015/403497 (cit. on p. 25).

[238] Y. Yang and J. Liu. "JVM: Java Visual Mapping tool for next generation sequencing read." In: *Advances in experimental medicine and biology* 827 (2015), pp. 11–8. DOI: 10.1007/978-94-017-9245-5_2 (cit. on p. 25).

[239] E. Georganas et al. "merAligner: A Fully Parallel Sequence Aligner". In: *2015 IEEE International Parallel and Distributed Processing Symposium*. IEEE, 2015, pp. 561–570. DOI: 10.1109/IPDPS.2015.96 (cit. on p. 25).

[240] B. Liu et al. "rHAT: fast alignment of noisy long reads with regional hashing". In: *Bioinformatics* 32.11 (2016), pp. 1625–1631. DOI: 10.1093/bioinformatics/btv662 (cit. on p. 25).

[241] S. S. Tithi, L. S. Heath, and L. Zhang. "SNPwise: A SNP-aware short read aligner". In: *7th International Conference on Bioinformatics and Computational Biology (BICoB)*. August. 2015, pp. 1–7 (cit. on pp. 25, 27).

[242] L. Santana-Quintero et al. “HIVE-Hexagon : High-Performance , Parallelized Sequence Alignment for Next-Generation Sequencing Data Analysis”. In: *PLoS ONE* 9.6 (2014). DOI: 10.1371/journal.pone.0099033 (cit. on p. 25).

[243] J. Kim, C. Li, and X. Xie. “Improving read mapping using additional prefix grams”. In: *BMC Bioinformatics* 15.1 (2014), p. 42. DOI: 10.1186/1471-2105-15-42 (cit. on p. 25).

[244] L. P. Dinu, R. T. udor Ionescu, and A. I. Tomescu. “A rank-based sequence aligner with applications in phylogenetic analysis”. In: *PloS one* 9.8 (2014), e104006. DOI: 10.1371/journal.pone.0104006 (cit. on p. 25).

[245] W. P. Lee et al. “MOSAIK: A hash-based algorithm for accurate next-generation sequencing short-read mapping”. In: *PLoS ONE* 9.3 (2014). DOI: 10.1371/journal.pone.0090581 (cit. on pp. 25, 27).

[246] F. Hach et al. “mrsFAST-Ultra: a compact, SNP-aware mapper for high performance sequencing applications.” In: *Nucleic acids research* (2014), pp. 1–7. DOI: 10.1093/nar/gku370 (cit. on pp. 25, 27, 63).

[247] S. Chen, A. Wang, and L. M. Li. “SEME: A Fast Mapper of Illumina Sequencing Reads with Statistical Evaluation”. In: *Journal of Computational Biology* 20.11 (2013), pp. 847–860. DOI: 10.1089/cmb.2013.0111 (cit. on p. 25).

[248] Y. Liao, G. K. Smyth, and W. Shi. “The Subread aligner: fast, accurate and scalable read mapping by seed-and-vote”. In: *Nucleic Acids Research* 41.10 (2013), e108–e108. DOI: 10.1093/nar/gkt214 (cit. on p. 25).

[249] P. M. Gontarz, J. Berger, and C. F. Wong. “SRmapper: A fast and sensitive genome-hashing alignment tool”. In: *Bioinformatics* 29.3 (2013), pp. 316–321. DOI: 10.1093/bioinformatics/bts712 (cit. on p. 25).

[250] N. Prezza et al. “ERNE-BS5: Aligning BS-treated sequences by multiple hits on a 5-letters alphabet”. In: *2012 ACM Conference on Bioinformatics, Computational Biology and Biomedicine, BCB 2012* (2012), pp. 12–19. DOI: 10.1145/2382936.2382938 (cit. on pp. 25, 26).

[251] A. Ahmadi et al. “Hobbes: Optimized gram-based methods for efficient read alignment”. In: *Nucleic Acids Research* 40.6 (2012), pp. 1–11. DOI: 10.1093/nar/gkr1246 (cit. on p. 25).

[252] D. Weese, M. Holtgrewe, and K. Reinert. “RazerS 3: Faster, fully sensitive read mapping”. In: *Bioinformatics* 28.20 (2012), pp. 2592–2599. DOI: 10.1093/bioinformatics/bts505 (cit. on p. 25).

[253] F. Vezzi et al. “rNA: A fast and accurate short reads numerical aligner”. In: *Bioinformatics* 28.1 (2012), pp. 123–124. DOI: 10.1093/bioinformatics/btr617 (cit. on p. 25).

[254] A. Policriti, A. I. Tomescu, and F. Vezzi. “A randomized Numerical Aligner (rNA)”. In: *Journal of Computer and System Sciences* 78.6 (2012), pp. 1868–1882. DOI: 10.1016/j.jcss.2011.12.007 (cit. on p. 25).

[255] J. C. Mu et al. “Fast and accurate read alignment for resequencing”. In: *Bioinformatics* 28.18 (2012), pp. 2366–2373. DOI: 10.1093/bioinformatics/bts450 (cit. on p. 25).

[256] G. G. Faust and I. M. Hall. "YAHA: Fast and flexible long-read alignment with optimal breakpoint detection". In: *Bioinformatics* 28.19 (2012), pp. 2417–2424. DOI: 10.1093/bioinformatics/bts456 (cit. on p. 25).

[257] Y. Li, J. M. Patel, and A. Terrell. "WHAM: A High-Throughput Sequence Alignment Method". In: *ACM Transactions on Database Systems* 37.4 (2012), pp. 1–39. DOI: 10.1145/2389241.2389247 (cit. on p. 25).

[258] S. Misra et al. "Anatomy of a hash-based long read sequence mapping algorithm for next generation DNA sequencing". In: *Bioinformatics* 27.2 (2011), pp. 189–195. DOI: 10.1093/bioinformatics/btq648 (cit. on p. 26).

[259] M. Zaharia, W. Bolosky, and K. Curtis. "Faster and More Accurate Sequence Alignment with SNAP". In: *arXiv preprints* (2011), pp. 1–10. URL: http://arxiv.org/abs/1111.5572 (cit. on p. 26).

[260] G. Lunter and M. Goodson. "Stampy: A statistical algorithm for sensitive and fast mapping of Illumina sequence reads". In: *Genome Research* 21.6 (2011), pp. 936–939. DOI: 10.1101/gr.111120.110 (cit. on pp. 26, 53).

[261] S. Misra et al. "FANGS: High Speed Sequence Mapping for Next generation Sequencers Sanchit". In: *Proceedings of the 2010 ACM Symposium on Applied Computing - SAC '10*. New York, New York, USA: ACM Press, 2010, p. 1539. DOI: 10.1145/1774088.1774419 (cit. on pp. 26, 29).

[262] T. D. Wu and S. Nacu. "Fast and SNP-tolerant detection of complex variants and splicing in short reads." In: *Bioinformatics* 26.7 (2010), pp. 873–81. DOI: 10.1093/bioinformatics/btq057 (cit. on pp. 26, 27, 29, 63).

[263] G. Rizk and D. Lavenier. "GASSST: global alignment short sequence search tool". In: *Bioinformatics* 26.20 (2010), pp. 2534–2540. DOI: 10.1093/bioinformatics/btq485 (cit. on p. 26).

[264] F. Hach et al. "mrsFAST: a cache-oblivious algorithm for short-read mapping." In: *Nature methods* 7.8 (2010), pp. 576–7. DOI: 10.1038/nmeth0810-576 (cit. on pp. 26, 27).

[265] T. Huynh, M. Vlachos, and I. Rigoutsos. "Anchoring millions of distinct reads on the human genome within seconds". In: *Proceedings of the 13th International Conference on Extending Database Technology - EDBT '10* (2010), p. 252. DOI: 10.1145/1739041.1739074 (cit. on p. 26).

[266] K. Frousios et al. "REAL: an efficient REad ALigner for next generation sequencing reads". In: *Proceedings of the First ACM International Conference on Bioinformatics and Computational Biology* (2010), pp. 154–159. DOI: 10.1145/1854776.1854801 (cit. on p. 26).

[267] N. L. Clement et al. "The GNUMAP algorithm: Unbiased probabilistic mapping of oligonucleotides from next-generation sequencing". In: *Bioinformatics* 26.1 (2009), pp. 38–45. DOI: 10.1093/bioinformatics/btp614 (cit. on p. 26).

[268] H. Bao et al. "MapNext: a software tool for spliced and unspliced alignments and SNP detection of short sequence reads." In: *BMC genomics* 10 Suppl 3.Suppl 3 (2009), S13. DOI: 10.1186/1471-2164-10-S3-S13 (cit. on p. 26).

[269] H. L. Eaves and Y. Gao. "MOM: maximum oligonucleotide mapping". In: *Bioinformatics* 25.7 (2009), pp. 969–970. DOI: 10.1093/bioinformatics/btp092 (cit. on p. 26).

[270] E. Rivals et al. “Mpscan: Fast localisation of multiple reads in genomes”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 5724 LNBI (2009), pp. 246–260. DOI: 10.1007/978-3-642-04241-6_21 (cit. on p. 26).

[271] C. Alkan et al. “Personalized copy number and segmental duplication maps using next-generation sequencing.” In: *Nature genetics* 41.10 (2009), pp. 1061–1067. DOI: 10.1038/ng.437 (cit. on p. 26).

[272] H. Xin et al. “Accelerating read mapping with FastHASH.” In: *BMC genomics* 14 Suppl 1.1 (2013), S13. DOI: 10.1186/1471-2164-14-S1-S13 (cit. on p. 26).

[273] D. Campagna et al. “PASS: a program to align short sequences”. In: *Bioinformatics* 25.7 (2009), pp. 967–968. DOI: 10.1093/bioinformatics/btp087 (cit. on p. 26).

[274] Y. Chen, T. Souaiaia, and T. Chen. “PerM: efficient mapping of short sequencing reads with periodic full sensitive spaced seeds”. In: *Bioinformatics* 25.19 (2009), pp. 2514–2521. DOI: 10.1093/bioinformatics/btp486 (cit. on pp. 26, 27, 31, 53, 122).

[275] Y. J. Kim et al. “ProbeMatch: Rapid alignment of oligonucleotides to genome allowing both gaps and mismatches”. In: *Bioinformatics* 25.11 (2009), pp. 1424–1425. DOI: 10.1093/bioinformatics/btp178 (cit. on p. 26).

[276] D. Weese et al. “RazerS - Fast read mapping with sensitivity control”. In: *Genome Research* 19.9 (2009), pp. 1646–1654. DOI: 10.1101/gr.088823.108 (cit. on p. 26).

[277] S. M. Rumble et al. “SHRiMP: Accurate mapping of short color-space reads”. In: *PLoS Computational Biology* 5.5 (2009), pp. 1–11. DOI: 10.1371/journal.pcbi.1000386 (cit. on pp. 26, 27).

[278] A. D. Smith, Z. Xuan, and M. Q. Zhang. “Using quality scores and longer reads improves accuracy of Solexa read mapping.” In: *BMC bioinformatics* 9 (2008), p. 128. DOI: 10.1186/1471-2105-9-128 (cit. on pp. 26, 29).

[279] A. D. Smith et al. “Updates to the RMAP short-read mapping software”. In: *Bioinformatics* 25.21 (2009), pp. 2841–2842. DOI: 10.1093/bioinformatics/btp533 (cit. on p. 26).

[280] H. Jiang and W. H. Wong. “SeqMap: Mapping massive amount of oligonucleotides to the genome”. In: *Bioinformatics* 24.20 (2008), pp. 2395–2396. DOI: 10.1093/bioinformatics/btn429 (cit. on p. 26).

[281] R. Li et al. “SOAP: Short oligonucleotide alignment program”. In: *Bioinformatics* 24.5 (2008), pp. 713–714. DOI: 10.1093/bioinformatics/btn025 (cit. on pp. 26, 29).

[282] H. Lin et al. “ZOOM! Zillions of oligos mapped.” In: *Bioinformatics* 24.21 (2008), pp. 2431–7. DOI: 10.1093/bioinformatics/btn416 (cit. on pp. 26, 27, 31, 122).

[283] T. D. Wu and C. K. Watanabe. “GMAP: A genomic mapping and alignment program for mRNA and EST sequences”. In: *Bioinformatics* 21.9 (2005), pp. 1859–1875. DOI: 10.1093/bioinformatics/bti310 (cit. on p. 26).

[284] L. Noé and G. Kucherov. “YASS: enhancing the sensitivity of DNA similarity search.” In: *Nucleic acids research* 33.Web Server issue (2005), W540–3. DOI: 10.1093/nar/gki478 (cit. on pp. 26, 31).

[285] Zemin Ning et al. “The SSAHA trace server”. In: *Proceedings. 2004 IEEE Computational Systems Bioinformatics Conference, 2004. CSB 2004.* Csb. IEEE, 2004, pp. 519–520. DOI: 10.1109/CSB.2004.1332490 (cit. on p. 26).

[286] M. Lexa and G. Valle. “PRIMEX: Rapid identification of oligonucleotide matches in whole genomes”. In: *Bioinformatics* 19.18 (2003), pp. 2486–2488. DOI: 10.1093/bioinformatics/btg350 (cit. on p. 26).

[287] Z. Ning, A. J. Cox, and J. C. Mullikin. “SSAHA: A fast search method for large DNA databases”. In: *Genome Research* 11.10 (2001), pp. 1725–1729. DOI: 10.1101/gr.194201 (cit. on p. 26).

[288] J. Choi et al. “HIA: a genome mapper using hybrid index-based sequence alignment”. In: *Algorithms for Molecular Biology* 10.1 (2015), p. 30. DOI: 10.1186/s13015-015-0062-4 (cit. on p. 26).

[289] V. L. Galinsky. “YOABS: yet other aligner of biological sequences–an efficient linearly scaling nucleotide aligner”. In: *Bioinformatics* 28.8 (2012), pp. 1070–1077. DOI: 10.1093/bioinformatics/bts102 (cit. on p. 26).

[290] N. Malhis and S. J. M. Jones. “High quality SNP calling using Illumina data at shallow coverage”. In: *Bioinformatics* 26.8 (2010), pp. 1029–1035. DOI: 10.1093/bioinformatics/btq092 (cit. on pp. 26, 49).

[291] N. Malhis et al. “Slider–maximum use of probability information for alignment of short sequence reads and SNP detection”. In: *Bioinformatics* 25.1 (2009), pp. 6–13. DOI: 10.1093/bioinformatics/btn565 (cit. on p. 26).

[292] I. Medina et al. “Highly sensitive and ultrafast read mapping for RNA-seq analysis.” In: *DNA research : an international journal for rapid publication of reports on genes and genomes* (2016), dsv039. DOI: 10.1093/dnares/dsv039 (cit. on p. 26).

[293] S. Saha and S. Rajasekaran. “POMP: a powerful splice mapper for RNA-seq reads”. In: *Proceedings of the 7th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics - BCB ’16*. New York, New York, USA: ACM Press, 2016, pp. 414–421. DOI: 10.1145/2975167.2975210 (cit. on p. 26).

[294] A. Srivastava et al. “RapMap: a rapid, sensitive and accurate tool for mapping RNA-seq reads to transcriptomes”. In: *Bioinformatics* 32.12 (2016), pp. i192–i200. DOI: 10.1093/bioinformatics/btw277 (cit. on p. 26).

[295] T. Bonfert et al. “ContextMap 2: fast and accurate context-based RNA-seq mapping”. In: *BMC Bioinformatics* 16.1 (2015), p. 122. DOI: 10.1186/s12859-015-0557-5 (cit. on p. 26).

[296] T. Bonfert and C. C. Friedel. *Prediction of Poly(A) Sites by Poly(A) Read Mapping*. Vol. 12. 1. 2017, e0170914. DOI: 10.1371/journal.pone.0170914 (cit. on p. 26).

[297] D. Kim, B. Langmead, and S. L. Salzberg. “HISAT: a fast spliced aligner with low memory requirements”. In: *Nature Methods* 12.4 (2015), pp. 357–360. DOI: 10.1038/nmeth.3317 (cit. on pp. 26, 66).

[298] C.-L. Xiao et al. “FANSe2: A Robust and Cost-Efficient Alignment Tool for Quantitative Next-Generation Sequencing Applications”. In: *PLoS ONE* 9.4 (2014), e94250. DOI: 10.1371/journal.pone.0094250 (cit. on p. 26).

[299] Y. S. Butterfield et al. "JAGuaR: Junction alignments to genome for RNA-seq reads". In: *PLoS ONE* 9.7 (2014). DOI: 10.1371/journal.pone.0102398 (cit. on p. 26).

[300] N. Philippe et al. "CRAC: an integrated approach to the analysis of RNA-seq reads." In: *Genome biology* 14.3 (2013), R30. DOI: 10.1186/gb-2013-14-3-r30 (cit. on p. 26).

[301] J. Wu et al. "OLego: Fast and sensitive mapping of spliced mRNA-Seq reads using small seeds". In: *Nucleic Acids Research* 41.10 (2013), pp. 5149–5163. DOI: 10.1093/nar/gkt216 (cit. on p. 26).

[302] D. Kim et al. "TopHat2: accurate alignment of transcriptomes in the presence of insertions, deletions and gene fusions". In: *Genome Biology* 14.4 (2013), R36. DOI: 10.1186/gb-2013-14-4-r36 (cit. on p. 26).

[303] T. Bonfert et al. "A context-based approach to identify the most likely mapping for RNA-seq experiments." In: *BMC bioinformatics* 13 Suppl 6.Suppl 6 (2012), S9. DOI: 10.1186/1471-2105-13-S6-S9 (cit. on p. 26).

[304] G. Zhang et al. "FANSe: an accurate algorithm for quantitative mapping of large scale sequencing reads". In: *Nucleic Acids Research* 40.11 (2012), e83–e83. DOI: 10.1093/nar/gks196 (cit. on p. 26).

[305] J. Hu et al. "OSA: A fast and accurate alignment tool for RNA-Seq". In: *Bioinformatics* 28.14 (2012), pp. 1933–1934. DOI: 10.1093/bioinformatics/bts294 (cit. on p. 26).

[306] Y. Zhang et al. "PASSion: A pattern growth algorithm-based pipeline for splice junction detection in paired-end RNA-seq data". In: *Bioinformatics* 28.4 (2012), pp. 479–486. DOI: 10.1093/bioinformatics/btr712 (cit. on p. 26).

[307] L. Y. Chen et al. "RNASEQR-a streamlined and accurate RNA-seq sequence analysis program". In: *Nucleic Acids Research* 40.6 (2012), pp. 1–12. DOI: 10.1093/nar/gkr1248 (cit. on p. 26).

[308] S. K. Lou et al. "ABMapper: A suffix array-based tool for multi-location searching and splice-junction mapping". In: *Bioinformatics* 27.3 (2011), pp. 421–422. DOI: 10.1093/bioinformatics/btq656 (cit. on p. 26).

[309] S. Huang et al. "SOAPsplice: Genome-wide ab initio detection of splice junctions from RNA-Seq data". In: *Frontiers in Genetics* 2.JULY (2011), pp. 1–12. DOI: 10.3389/fgene.2011.00046 (cit. on p. 26).

[310] M. T. Dimon, K. Sorber, and J. L. DeRisi. "HMMSplicer: A Tool for Efficient and Sensitive Discovery of Known and Novel Splice Junctions in RNA-Seq Data". In: *PLoS ONE* 5.11 (2010). DOI: 10.1371/journal.pone.0013875 (cit. on p. 26).

[311] K. Wang et al. "MapSplice: Accurate mapping of RNA-seq reads for splice junction discovery". In: *Nucleic Acids Research* 38.18 (2010), pp. 1–14. DOI: 10.1093/nar/gkq622 (cit. on p. 26).

[312] G. Jean et al. *RNA-seq read alignments with PALMapper*. SUPP.32. 2010, pp. 1–37. DOI: 10.1002/0471250953.bi1106s32 (cit. on p. 26).

[313] K. F. Au et al. "Detection of splice junctions from paired-end RNA-seq data by SpliceMap". In: *Nucleic Acids Research* 38.14 (2010), pp. 4570–4578. DOI: 10.1093/nar/gkq211 (cit. on p. 26).

[314] D. W. Bryant et al. "Supersplat-spliced RNA-seq alignment". In: *Bioinformatics* 26.12 (2010), pp. 1500–1505. DOI: 10.1093/bioinformatics/btq206 (cit. on p. 26).

[315] N. Fahlgren et al. "Computational and analytical framework for small RNA profiling by high-throughput sequencing". In: *RNA* 15.5 (2009), pp. 992–1002. DOI: 10.1261/rna.1473809 (cit. on p. 26).

[316] F. De Bona et al. "Optimal spliced alignments of short sequence reads". In: *Bioinformatics* 24.16 (2008), pp. i174–i180. DOI: 10.1093/bioinformatics/btn300 (cit. on p. 26).

[317] R. Mott. "EST_GENOME: a program to align spliced DNA sequences to unspliced genomic DNA". In: *Comput Appl Biosci* 13.4 (1997), pp. 477–478. DOI: 10.1093/bioinformatics/13.4.477 (cit. on p. 26).

[318] P. G. Engström et al. "Systematic evaluation of spliced alignment programs for RNA-seq data." In: *Nature methods* 10.12 (2013), pp. 1185–91. DOI: 10.1038/nmeth.2722 (cit. on p. 26).

[319] G. Baruzzo et al. "Simulation-based comprehensive benchmarking of RNA-seq aligners". In: *Nature Methods* 14.2 (2016), pp. 135–139. DOI: 10.1038/nmeth.4106 (cit. on p. 26).

[320] E. Y. Harris, R. Ounit, and S. Lonardi. "BRAT-nova: fast and accurate mapping of bisulfite-treated reads". In: *Bioinformatics* 32.17 (2016), pp. 2696–2698. DOI: 10.1093/bioinformatics/btw226 (cit. on p. 26).

[321] H. Chen, A. D. Smith, and T. Chen. "WALT: fast and accurate read mapping for bisulfite sequencing". In: *Bioinformatics* (2016), btw490. DOI: 10.1093/bioinformatics/btw490 (cit. on pp. 26, 31).

[322] D. Ryan and D. Ehninger. "Bison: bisulfite alignment on nodes of a cluster". In: *BMC Bioinformatics* 15.1 (2014), p. 337. DOI: 10.1186/1471-2105-15-337 (cit. on p. 26).

[323] A. Manconi et al. "GPU-BSM: A GPU-based tool to map bisulfite-treated read". In: *PLoS ONE* 9.5 (2014). DOI: 10.1371/journal.pone.0097277 (cit. on pp. 26, 28).

[324] R. Sun, Y. Tian, and X. Chen. "TAMeBS: A sensitive bisulfite-sequencing read mapping tool for DNA methylation analysis". In: *Proceedings - 2014 IEEE International Conference on Bioinformatics and Biomedicine, IEEE BIBM 2014* (2014), pp. 176–181. DOI: 10.1109/BIBM.2014.6999148 (cit. on p. 26).

[325] J.-Q. Lim et al. "BatMeth: improved mapper for bisulfite sequencing reads on DNA methylation". In: *Genome Biology* 13.10 (2012), R82. DOI: 10.1186/gb-2012-13-10-r82 (cit. on p. 26).

[326] H. Q. Dinh et al. "Advanced methylome analysis after Bisulfite deep sequencing: An example in Arabidopsis". In: *PLoS ONE* 7.7 (2012). DOI: 10.1371/journal.pone.0041528 (cit. on p. 26).

[327] K. D. Hansen et al. "BSmooth: from whole genome bisulfite sequencing reads to differentially methylated regions". In: *Genome Biology* 13.10 (2012), R83. DOI: 10.1186/gb-2012-13-10-r83 (cit. on p. 26).

[328] Y. Xi et al. “RRBSMAP: A fast, accurate and user-friendly alignment tool for reduced representation bisulfite sequencing”. In: *Bioinformatics* 28.3 (2012), pp. 430–432. DOI: 10.1093/bioinformatics/btr668 (cit. on p. 26).

[329] C. Otto, P. F. Stadler, and S. Hoffmann. “Fast and sensitive mapping of bisulfite-treated sequencing data”. In: *Bioinformatics* 28.13 (2012), pp. 1698–1704. DOI: 10.1093/bioinformatics/bts254 (cit. on p. 26).

[330] F. Krueger and S. R. Andrews. “Bismark: A flexible aligner and methylation caller for Bisulfite-Seq applications”. In: *Bioinformatics* 27.11 (2011), pp. 1571–1572. DOI: 10.1093/bioinformatics/btr167 (cit. on p. 26).

[331] C. Coarfa et al. “Pash 3.0: A versatile software package for read mapping and integrative analysis of genomic and epigenomic variation using massively parallel DNA sequencing.” In: *BMC bioinformatics* 11.1 (2010), p. 572. DOI: 10.1186/1471-2105-11-572 (cit. on p. 26).

[332] Y. Xi and W. Li. “BSMAP: whole genome bisulfite sequence MAPping program.” In: *BMC bioinformatics* 10.1 (2009), p. 232. DOI: 10.1186/1471-2105-10-232 (cit. on p. 26).

[333] M. Zeschnigk et al. “Massive parallel bisulfite sequencing of CG-rich DNA fragments reveals that methylation of many X-chromosomal CpG islands in female blood DNA is incomplete.” In: *Human molecular genetics* 18.8 (2009), pp. 1439–48. DOI: 10.1093/hmg/ddp054 (cit. on p. 26).

[334] J. Tárraga et al. “A parallel and sensitive software tool for methylation analysis on multicore platforms”. In: *Bioinformatics* 31.19 (2015), pp. 3130–3138. DOI: 10.1093/bioinformatics/btv357 (cit. on p. 26).

[335] C. Hong et al. “Probabilistic alignment leads to improved accuracy and read coverage for bisulfite sequencing data”. In: *BMC Bioinformatics* 14.1 (2013), p. 337. DOI: 10.1186/1471-2105-14-337 (cit. on p. 26).

[336] D. Campagna et al. “PASS-bis: A bisulfite aligner suitable for whole methylome analysis of Illumina and SOLiD reads”. In: *Bioinformatics* 29.2 (2013), pp. 268–270. DOI: 10.1093/bioinformatics/bts675 (cit. on p. 26).

[337] E. Y. Harris et al. “BRAT-BW: Efficient and accurate mapping of bisulfite-treated reads”. In: *Bioinformatics* 28.13 (2012), pp. 1795–1796. DOI: 10.1093/bioinformatics/bts264 (cit. on p. 26).

[338] Y. Saito, J. Tsuji, and T. Mituyama. “Bisulfighter: accurate detection of methylated cytosines and differentially methylated regions”. In: *Nucleic Acids Research* 42.6 (2014), e45–e45. DOI: 10.1093/nar/gkt1373 (cit. on p. 26).

[339] W. Guo et al. “BS-Seeker2: a versatile aligning pipeline for bisulfite sequencing data.” In: *BMC genomics* 14 (2013), p. 774. DOI: 10.1186/1471-2164-14-774 (cit. on p. 26).

[340] B. Pedersen et al. “MethylCoder: Software pipeline for bisulte-treated sequences”. In: *Bioinformatics* 27.17 (2011), pp. 2435–2436. DOI: 10.1093/bioinformatics/btr394 (cit. on p. 26).

[341] P.-Y. Chen, S. Cokus, and M. Pellegrini. “BS Seeker: precise mapping for bisulfite sequencing”. In: *BMC Bioinformatics* 11.1 (2010), p. 203. DOI: 10.1186/1471-2105-11-203 (cit. on p. 26).

[342] B. Kreck et al. "B-SOLANA: An approach for the analysis of two-base encoding bisulfite sequencing data". In: *Bioinformatics* 28.3 (2012), pp. 428–429. DOI: 10.1093/bioinformatics/btr660 (cit. on p. 26).

[343] A. Chatterjee et al. "Comparison of alignment software for genome-wide bisulphite sequence data". In: *Nucleic Acids Research* 40.10 (2012). DOI: 10.1093/nar/gks150 (cit. on p. 26).

[344] H. Tran et al. "Objective and Comprehensive Evaluation of Bisulfite Short Read Mapping Tools". In: *Advances in Bioinformatics* 2014 (2014), pp. 1–11. DOI: 10.1155/2014/472045 (cit. on p. 26).

[345] G. Kunde-Ramamoorthy et al. "Comparison and quantitative verification of mapping algorithms for whole-genome bisulfite sequencing". In: *Nucleic Acids Research* 42.6 (2014), e43–e43. DOI: 10.1093/nar/gkt1325 (cit. on p. 26).

[346] J. Tsuji and Z. Weng. "Evaluation of preprocessing, mapping and postprocessing algorithms for analyzing whole genome bisulfite sequencing data". In: *Briefings in bioinformatics* August (2015), bbv103. DOI: 10.1093/bib/bbv103 (cit. on p. 26).

[347] P. Wulfridge et al. "Choice of reference genome can introduce massive bias in bisulfite sequencing data". In: *bioRxiv preprints* (2016), pp. 1–31. DOI: 10.1101/076844. URL: http://biorxiv.org/lookup/doi/10.1101/076844 (cit. on p. 26).

[348] D. Lee et al. "Fast and accurate mapping of Complete Genomics reads". In: *Methods* 79-80.2 (2015), pp. 3–10. DOI: 10.1016/j.ymeth.2014.10.012 (cit. on p. 27).

[349] M. T. Chou et al. "Tailor: A computational framework for detecting non-templated tailing of small silencing RNAs". In: *Nucleic Acids Research* 43.17 (2015). DOI: 10.1093/nar/gkv537 (cit. on p. 27).

[350] A.-K. Emde et al. "MicroRazerS: rapid alignment of small RNA reads". In: *Bioinformatics* 26.1 (2010), pp. 123–124. DOI: 10.1093/bioinformatics/btp601 (cit. on p. 27).

[351] K. Prüfer et al. "PatMaN: Rapid alignment of short sequences to large databases". In: *Bioinformatics* 24.13 (2008), pp. 1530–1531. DOI: 10.1093/bioinformatics/btn223 (cit. on p. 27).

[352] M. Ziemann, A. Kaspi, and A. El-Osta. "Evaluation of microRNA alignment techniques". In: *RNA* 22.8 (2016), pp. 1120–1138. DOI: 10.1261/rna.055509.115 (cit. on p. 27).

[353] F. Hormozdiari et al. "Sensitive and fast mapping of di-base encoded reads". In: *Bioinformatics* 27.14 (2011), pp. 1915–1921. DOI: 10.1093/bioinformatics/btr303 (cit. on p. 27).

[354] D. L. A. Wood et al. "X-MATE: A flexible system for mapping short read data". In: *Bioinformatics* 27.4 (2011), pp. 580–581. DOI: 10.1093/bioinformatics/btq698 (cit. on p. 27).

[355] M. Gîrdea et al. "Designing Efficient Spaced Seeds for SOLiD Read Mapping". In: *Advances in Bioinformatics* 2010 (2010), pp. 1–12. DOI: 10.1155/2010/708501 (cit. on pp. 27, 31).

[356] N. Cloonan et al. "RNA-MATE: A recursive mapping strategy for high-throughput RNA-sequencing data". In: *Bioinformatics* 25.19 (2009), pp. 2615–2616. DOI: 10.1093/bioinformatics/btp459 (cit. on p. 27).

[357] B. D. Ondov et al. “Efficient mapping of Applied Biosystems SOLiD sequence data to a reference genome for functional genomic applications”. In: *Bioinformatics* 24.23 (2008), pp. 2776–2777. DOI: 10.1093/bioinformatics/btn512 (cit. on p. 27).

[358] M. Rho et al. “Diverse CRISPRs evolving in human microbiomes”. In: *PLoS Genetics* 8.6 (2012). DOI: 10.1371/journal.pgen.1002441 (cit. on p. 27).

[359] B. Buchfink, C. Xie, and D. H. Huson. “Fast and sensitive protein alignment using DIAMOND.” In: *Nature methods* 12.1 (2015), pp. 59–60. DOI: 10.1038/nmeth.3176 (cit. on pp. 27, 31, 79).

[360] Y. Zhao, H. Tang, and Y. Ye. “RAPSearch2: A fast and memory-efficient protein similarity search tool for next-generation sequencing data”. In: *Bioinformatics* 28.1 (2012), pp. 125–126. DOI: 10.1093/bioinformatics/btr595 (cit. on p. 27).

[361] Y. Ye, J.-H. Choi, and H. Tang. “RAPSearch: a fast protein similarity search tool for short reads”. In: *BMC Bioinformatics* 12.1 (2011), p. 159. DOI: 10.1186/1471-2105-12-159 (cit. on p. 27).

[362] X. Peng et al. “Re-alignment of the unmapped reads with base quality score”. In: *BMC Bioinformatics* 16.Suppl 5 (2015), S8. DOI: 10.1186/1471-2105-16-S5-S8 (cit. on p. 27).

[363] T. Turki and U. Roshan. “MaxSSmap: a GPU program for mapping divergent short reads to genomes with the maximum scoring subsequence”. In: *BMC Genomics* 15.1 (2014), p. 969. DOI: 10.1186/1471-2164-15-969 (cit. on pp. 27, 28).

[364] K. Katoh and M. C. Frith. “Adding unaligned sequences into an existing alignment using MAFFT and LAST”. In: *Bioinformatics* 28.23 (2012), pp. 3144–3146. DOI: 10.1093/bioinformatics/bts578 (cit. on p. 27).

[365] T. Marschall et al. “Computational pan-genomics: status, promises and challenges”. In: *Briefings in Bioinformatics* (2016), bbw089. DOI: 10.1093/bib/bbw089 (cit. on pp. 27, 28, 66).

[366] A. M. Novak et al. “Genome Graphs”. In: 94040 (2017), pp. 1–26. DOI: 10.1101/101378 (cit. on p. 27).

[367] K. D. Rand et al. “Coordinates and Intervals in Graph-based Reference Genomes”. In: (2016), pp. 1–10 (cit. on p. 27).

[368] B. Paten, A. Novak, and D. Haussler. “Mapping to a Reference Genome Structure”. In: *arXiv preprints* (2014). URL: http://arxiv.org/abs/1404.5010 (cit. on pp. 27, 66).

[369] A. M. Novak et al. “Canonical, stable, general mapping using context schemes”. In: *Bioinformatics* (2015), btv435. DOI: 10.1093/bioinformatics/btv435 (cit. on pp. 27, 66).

[370] J. Siren, N. Valimaki, and V. Makinen. “Indexing Graphs for Path Queries with Applications in Genome Research”. In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics*. Lecture Notes in Computer Science 11.2 (2014), pp. 375–388. DOI: 10.1109/TCBB.2013.2297101 (cit. on pp. 27, 35, 66).

[371] J. Sirén. “Indexing Variation Graphs”. In: *2017 Proceedings of the Ninteenth Workshop on Algorithm Engineering and Experiments (ALENEX)*. Philadelphia, PA: Society for Industrial and Applied Mathematics, 2017, pp. 13–27. DOI: 10.1137/1.9781611974768.2 (cit. on pp. 27, 35, 111).

[372] B. Kramer et al. *VATRAM: VAriant Tolerant ReAd Mapper*. Tech. rep. Dortmund, 2015. DOI: 10.17877/DE290R-16435 (cit. on pp. 27, 28).

[373] J. Quedenfeld and S. Rahmann. "Variant tolerant read mapping using min-hashing". In: *arXiv preprints* (2017), pp. 1–19. URL: http://arxiv.org/abs/1702.01703 (cit. on pp. 27, 28).

[374] S. T. Sherry et al. "dbSNP: the NCBI database of genetic variation." In: *Nucleic acids research* 29.1 (2001), pp. 308–11. DOI: 10.1093/nar/29.1.308 (cit. on pp. 27, 63).

[375] L. Huang, V. Popic, and S. Batzoglou. "Short read alignment with populations of genomes". In: *Bioinformatics* 29.13 (2013), pp. i361–i370. DOI: 10.1093/bioinformatics/btt215 (cit. on pp. 27, 66).

[376] K. Schneeberger et al. "Simultaneous alignment of short reads against multiple genomes." In: *Genome biology* 10.9 (2009), R98. DOI: 10.1186/gb-2009-10-9-r98 (cit. on pp. 27, 66).

[377] A. Limasset et al. "Read mapping on de Bruijn graphs". In: *BMC Bioinformatics* 17.1 (2016), p. 237. DOI: 10.1186/s12859-016-1103-9 (cit. on pp. 27, 66).

[378] B. Liu et al. "deBGA: read alignment with de Bruijn graph-based seed and extension". In: *Bioinformatics* 32.21 (2016), pp. 3224–3232. DOI: 10.1093/bioinformatics/btw371 (cit. on p. 27).

[379] S. Maciuca et al. "A Natural Encoding of Genetic Variation in a Burrows-Wheeler Transform to Enable Mapping and Genome Inference". In: 2016, pp. 222–233. DOI: 10.1007/978-3-319-43681-4_18 (cit. on p. 27).

[380] C. S. Iliopoulos et al. "An algorithm for mapping short reads to a dynamically changing genomic sequence". In: *Journal of Discrete Algorithms* 10 (2012), pp. 15–22. DOI: 10.1016/j.jda.2011.08.006 (cit. on pp. 28, 54).

[381] J. Pritt. *Efficiently Improving the Reference Genome for DNA Read Alignment*. Tech. rep. 2013 (cit. on pp. 28, 54).

[382] D. Yorukoglu et al. "Compressive mapping for next-generation sequencing". In: *Nature Biotechnology* 34.4 (2016), pp. 374–376. DOI: 10.1038/nbt.3511 (cit. on p. 28).

[383] B. Berger, N. M. Daniels, and Y. W. Yu. "Computational biology in the 21st century". In: *Communications of the ACM* 59.8 (2016), pp. 72–80. DOI: 10.1145/2957324 (cit. on p. 28).

[384] A. Broder. "On the resemblance and containment of documents". In: *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No.97TB100171)*. IEEE Comput. Soc, 1997, pp. 21–29. DOI: 10.1109/SEQUEN.1997.666900 (cit. on pp. 28, 79).

[385] M. Roberts et al. "Reducing storage requirements for biological sequence comparison". In: *Bioinformatics* 20.18 (2004), pp. 3363–3369. DOI: 10.1093/bioinformatics/bth408 (cit. on p. 28).

[386] A. Shrikumar et al. "A fast approximate algorithm for mapping long reads to large reference databases". In: (2017), pp. 1–16. DOI: 10.1101/103812 (cit. on p. 28).

[387] H. Li. "Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences". In: *Bioinformatics* 32.14 (2016), pp. 2103–2110. DOI: 10.1093/bioinformatics/btw152 (cit. on p. 28).

[388] V. Popic and S. Batzoglou. "Privacy-Preserving Read Mapping Using Locality Sensitive Hashing and Secure Kmer Voting". In: *bioRxiv preprints* (2016), p. 046920. DOI: 10.1101/046920. URL: http://biorxiv.org/content/early/2016/04/03/046920.abstract%20http://biorxiv.org/lookup/doi/10.1101/046920 (cit. on p. 28).

[389] M. Glunčić and V. Paar. "Direct mapping of symbolic DNA sequence into frequency domain in global repeat map algorithm." In: *Nucleic acids research* 41.1 (2013), pp. 1–17. DOI: 10.1093/nar/gks721 (cit. on p. 28).

[390] B. Zitová and J. Flusser. "Image registration methods: a survey". In: *Image and Vision Computing* 21.11 (2003), pp. 977–1000. DOI: 10.1016/S0262-8856(03)00137-9 (cit. on p. 28).

[391] J. Lorenzo-Ginori et al. "Digital Signal Processing in the Analysis of Genomic Sequences". In: *Current Bioinformatics* 4.1 (2009), pp. 28–40. DOI: 10.2174/157489309787158134 (cit. on p. 28).

[392] A. Tapinos et al. "Alignment by numbers: sequence assembly using compressed numerical representations". In: *bioRxiv preprints* (2014), p. 011940. DOI: 10.1101/011940. URL: http://biorxiv.org/lookup/doi/10.1101/011940 (cit. on p. 28).

[393] J. Duda. "Distortion-Resistant Hashing for rapid search of similar DNA subsequence". In: *arXiv preprints* (2016), pp. 1–5. URL: http://arxiv.org/abs/1602.05889 (cit. on p. 28).

[394] M. S. Lindner et al. "HiLive – Real-Time Mapping of Illumina Reads while Sequencing". In: *Bioinformatics* (2016), btw659. DOI: 10.1093/bioinformatics/btw659 (cit. on p. 28).

[395] E. J. Houtgast et al. "An Efficient GPU-Accelerated Implementation of Genomic Short Read Mapping with BWA-MEM". In: *ACM SIGARCH Computer Architecture News* 44.4 (2017), pp. 38–43. DOI: 10.1145/3039902.3039910 (cit. on p. 28).

[396] E. F. O. Sandes et al. "CUDAlign 4.0: Incremental Speculative Traceback for Exact Chromosome-Wide Alignment in GPU Clusters". In: *IEEE Transactions on Parallel and Distributed Systems* 9219.c (2016), pp. 1–1. DOI: 10.1109/TPDS.2016.2515597 (cit. on p. 28).

[397] P. Wojciechowski et al. "G-MAPSEQ – a new method for mapping reads to a reference genome". In: *Foundations of Computing and Decision Sciences* 41.2 (2016). DOI: 10.1515/fcds-2016-0007 (cit. on p. 28).

[398] R. Wilton et al. "Arioc: high-throughput read alignment with GPU-accelerated exploration of the seed-and-extend search space". In: *PeerJ* 3 (2015), e808. DOI: 10.7717/peerj.808 (cit. on p. 28).

[399] B. S. Pedersen et al. "Fast and accurate alignment of long bisulfite-seq reads". In: *arXiv preprints* 00.00 (2014), pp. 1–2. URL: http://arxiv.org/abs/1401.1129 (cit. on p. 28).

[400] E. F. D. O. Sandes et al. "CUDAlign 3.0: Parallel biological sequence comparison in large GPU clusters". In: *Proceedings - 14th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2014* (2014), pp. 160–169. DOI: 10.1109/CCGrid.2014.18 (cit. on p. 28).

[401] Y. Liu and B. Schmidt. "CUSHAW2-GPU: Empowering faster gapped short-read alignment using GPU computing". In: *IEEE Design and Test* 31.1 (2014), pp. 31–39. DOI: 10.1109/MDAT.2013.2284198 (cit. on p. 28).

[402] J. Köster and S. Rahmann. "Massively parallel read mapping on GPUs with the q -group index and PEANUT". In: *PeerJ* 2 (2014), e606. DOI: 10.7717/peerj.606 (cit. on p. 28).

[403] I. A. Wright and S. A. Travers. "RAMICS: Trainable, high-speed and biologically relevant alignment of high-throughput sequencing reads to coding DNA". In: *Nucleic Acids Research* 42.13 (2014). DOI: 10.1093/nar/gku473 (cit. on p. 28).

[404] F. J. Sedlazeck, P. Rescheneder, and A. von Haeseler. "NextGenMap: fast and accurate read mapping in highly polymorphic genomes". In: *Bioinformatics* 29.21 (2013), pp. 2790–2791. DOI: 10.1093/bioinformatics/btt468 (cit. on p. 28).

[405] R. Luo et al. "SOAP3-dp: Fast, Accurate and Sensitive GPU-Based Short Read Aligner". In: *PLoS ONE* 8.5 (2013). DOI: 10.1371/journal.pone.0065632 (cit. on p. 28).

[406] W. Frohmberg et al. "G-DNA – a highly efficient multi-GPU/MPI tool for aligning nucleotide reads". In: *Bulletin of the Polish Academy of Sciences: Technical Sciences* 61.4 (2013), pp. 989–992. DOI: 10.2478/bpasts-2013-0106 (cit. on pp. 28, 29).

[407] A. Abu-Doleh et al. "Masher: Mapping Long(er) Reads with Hash-based Genome Indexing on GPUs". In: *Proceedings of the International Conference on Bioinformatics, Computational Biology and Biomedical Informatics - BCB'13*. New York, New York, USA: ACM Press, 2013, pp. 341–350. DOI: 10.1145/2506583.2506641 (cit. on p. 28).

[408] P. Klus et al. "BarraCUDA - a fast short read sequence aligner using graphics processing units". In: *BMC Research Notes* 5.1 (2012), p. 27. DOI: 10.1186/1756-0500-5-27 (cit. on p. 28).

[409] W. B. Langdon et al. "Improving CUDA DNA Analysis Software with Genetic Programming". In: *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference - GECCO '15*. New York, New York, USA: ACM Press, 2015, pp. 1063–1070. DOI: 10.1145/2739480.2754652 (cit. on p. 28).

[410] Y. Liu, B. Schmidt, and D. L. Maskell. "CUSHAW: a CUDA compatible short read aligner to large genomes based on the Burrows-Wheeler transform". In: *Bioinformatics* 28.14 (2012), pp. 1830–1837. DOI: 10.1093/bioinformatics/bts276 (cit. on p. 28).

[411] M. Lu et al. "High-performance short sequence alignment with GPU acceleration". In: *Distributed and Parallel Databases* 30.5-6 (2012), pp. 385–399. DOI: 10.1007/s10619-012-7099-x (cit. on p. 28).

[412] J. S. Torres et al. "Using GPUs for the Exact Alignment of Short-Read Genetic Sequences by Means of the Burrows-Wheeler Transform". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 9.4 (2012), pp. 1245–1256. DOI: 10.1109/TCBB.2012.49 (cit. on p. 28).

[413] C.-M. Liu et al. "SOAP3: ultra-fast GPU-based parallel alignment tool for short reads". In: *Bioinformatics* 28.6 (2012), pp. 878–879. DOI: 10.1093/bioinformatics/bts061 (cit. on p. 28).

[414] J. Blom et al. "Exact and complete short-read alignment to microbial genomes using Graphics Processing Unit programming". In: *Bioinformatics* 27.10 (2011), pp. 1351–1358. DOI: 10.1093/bioinformatics/btr151 (cit. on p. 28).

[415] E. F. O. Sandes and A. C. M. de Melo. "CUDAlign: Using GPU to Accelerate the Comparison of Megabase Genomic Sequences". In: *Proceedings of the 15th ACM SIGPLAN symposium on Principles and practice of parallel programming - PPoPP '10*. Vol. 45. 5. New York, New York, USA: ACM Press, 2010, p. 137. DOI: 10.1145/1693453.1693473 (cit. on p. 28).

[416] A. M. Aji, L. Zhang, and W. C. Feng. "GPU-RMAP: Accelerating short-read mapping on graphics processors". In: *Proceedings - 2010 13th IEEE International Conference on Computational Science and Engineering, CSE 2010* (2010), pp. 168–175. DOI: 10.1109/CSE.2010.29 (cit. on p. 28).

[417] M. C. Schatz et al. "High-throughput sequence alignment using Graphics Processing Units." In: *BMC bioinformatics* 8 (2007), p. 474. DOI: 10.1186/1471-2105-8-474 (cit. on p. 28).

[418] M. S. Nobile et al. "Graphics processing units in bioinformatics, computational biology and systems biology". In: *Briefings in Bioinformatics* May (2016), bbw058. DOI: 10.1093/bib/bbw058 (cit. on p. 28).

[419] J. M. Abuín et al. "SparkBWA: Speeding Up the Alignment of High-Throughput DNA Sequencing Data". In: *PLOS ONE* 11.5 (2016), e0155461. DOI: 10.1371/journal.pone.0155461 (cit. on p. 29).

[420] J. M. Abuín et al. "BigBWA: approaching the Burrows–Wheeler aligner to Big Data technologies". In: *Bioinformatics* (2015), btv506. DOI: 10.1093/bioinformatics/btv506 (cit. on p. 29).

[421] R. V. Pandey and C. Schlötterer. "DistMap: A Toolkit for Distributed Short Read Mapping on a Hadoop Cluster". In: *PLoS ONE* 8.8 (2013), e72614. DOI: 10.1371/journal.pone.0072614 (cit. on p. 29).

[422] L. Pireddu, S. Leo, and G. Zanetti. "Seal: A distributed short read mapping and duplicate removal tool". In: *Bioinformatics* 27.15 (2011), pp. 2159–2160. DOI: 10.1093/bioinformatics/btr325 (cit. on p. 29).

[423] T. Nguyen, W. Shi, and D. Ruden. "CloudAligner: A fast and full-featured MapReduce based tool for sequence mapping." In: *BMC research notes* 4.1 (2011), p. 171. DOI: 10.1186/1756-0500-4-171 (cit. on p. 29).

[424] M. C. Schatz. "CloudBurst: highly sensitive read mapping with MapReduce." In: *Bioinformatics* 25.11 (2009), pp. 1363–9. DOI: 10.1093/bioinformatics/btp236 (cit. on p. 29).

[425] M. C. Schatz. *BlastReduce: high performance short read mapping with MapReduce*. Tech. rep. 2008 (cit. on p. 29).

[426] F. Jarlier et al. *QUASART: QUick Algorithms for Sequencing data Analysis with massive Reads Toolbox*. To appear. (cit. on p. 29).

[427] P. Darren et al. "Speeding Up Large-Scale Next Generation Sequencing Data Analysis with pBWA". In: *Journal of Applied Bioinformatics & Computational Biology* 1 (2012), pp. 1–6. DOI: 10.4172/2329-9533.1000101 (cit. on p. 29).

[428] S. Misra et al. "pFANGS: Parallel high speed sequence mapping for next generation 454-roche sequencing reads". In: *Proceedings of the 2010 IEEE International Symposium on Parallel and Distributed Processing, Workshops and Phd Forum, IPDPSW 2010* (2010). DOI: 10.1109/IPDPSW.2010.5470894 (cit. on p. 29).

[429] E. B. Fernandez et al. "FHAST: FPGA-Based Acceleration of Bowtie in Hardware". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 12.5 (2015), pp. 973–981. DOI: 10.1109/TCBB.2015.2405333 (cit. on p. 29).

[430] Y. Chen, B. Schmidt, and D. L. Maskell. "A hybrid short read mapping accelerator". In: *BMC Bioinformatics* 14.1 (2013), p. 67. DOI: 10.1186/1471-2105-14-67 (cit. on p. 29).

[431] C. Nelson et al. "Shepard: A fast exact match short read aligner". In: *Tenth ACM/IEEE International Conference on Formal Methods and Models for Codesign (MEMCODE2012)*. IEEE, 2012, pp. 91–94. DOI: 10.1109/MEMCOD.2012.6292304 (cit. on p. 29).

[432] D. Lavenier, J.-F. Roy, and D. Furodet. "DNA mapping using Processor-in-Memory architecture". In: *2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, 2016, pp. 1429–1435. DOI: 10.1109/BIBM.2016.7822732 (cit. on p. 29).

[433] J. Kim et al. *Genome Read In-Memory (GRIM) Filter: Fast Location Filtering in DNA Read Mapping using Emerging Memory Technologies*. Tech. rep. 2017. URL: https://people.inf.ethz.ch/omutlu/pub/GRIM-genome-read-in-memory-filter_psb17-poster.pdf (cit. on p. 29).

[434] Y. Cui et al. "B-MIC : An Ultrafast Three-Level Parallel Sequence Aligner". In: *Interdisciplinary Sciences: Computational Life Sciences* 8.28 (2016). DOI: 10.1007/s12539-015-0278-5 (cit. on p. 29).

[435] R. Luo et al. "MICA: A fast short-read aligner that takes full advantage of Many Integrated Core Architecture (MIC)". In: *BMC Bioinformatics* 16.Suppl 7 (2015), S10. DOI: 10.1186/1471-2105-16-S7-S10 (cit. on p. 29).

[436] C. Nelson et al. "RAMPS: A Reconfigurable Architecture for Minimal Perfect Sequencing". In: *IEEE Transactions on Parallel and Distributed Systems* 27.10 (2016), pp. 3029–3043. DOI: 10.1109/TPDS.2015.2513053 (cit. on p. 29).

[437] *Convey Computer Burrows-Wheeler Aligner Personality*. Tech. rep. 2011. URL: http://media.marketwire.com/attachments/201110/30059_BWADatasheet.pdf (cit. on p. 29).

[438] N. Ahmed et al. "Heterogeneous hardware/software acceleration of the BWA-MEM DNA alignment algorithm". In: *2015 IEEE/ACM International Conference on Computer-Aided Design, ICCAD 2015* (2016), pp. 240–246. DOI: 10.1109/ICCAD.2015.7372576 (cit. on p. 29).

[439] B. E. Blaisdell. "A measure of the similarity of sets of sequences not requiring sequence alignment." In: *Proceedings of the National Academy of Sciences* 83.14 (1986), pp. 5155–5159. DOI: 10.1073/pnas.83.14.5155 (cit. on p. 29).

[440] S. Vinga and J. Almeida. "Alignment-free sequence comparison–a review". In: *Bioinformatics* 19.4 (2003), pp. 513–523. DOI: 10.1093/bioinformatics/btg005 (cit. on p. 29).

[441] K. Song et al. "New developments of alignment-free sequence comparison: Measures, statistics and next-generation sequencing". In: *Briefings in Bioinformatics* 15.3 (2014), pp. 343–353. DOI: 10.1093/bib/bbt067 (cit. on p. 29).

[442] S. Vinga. "Information theory applications for biological sequence analysis". In: *Briefings in Bioinformatics* 15.3 (2014), pp. 376–389. DOI: 10.1093/bib/bbt068 (cit. on p. 29).

[443] M. Comin, A. Leoni, and M. Schimd. "Clustering of reads with alignment-free measures and quality values." In: *Algorithms for molecular biology : AMB* 10 (2015), p. 4. DOI: 10.1186/s13015-014-0029-x (cit. on p. 29).

[444] M. Comin and M. Schimd. "Fast comparison of genomic and meta-genomic reads with alignment-free measures based on quality values". In: *BMC Medical Genomics* 9.S1 (2016), p. 36. DOI: 10.1186/s12920-016-0193-6 (cit. on p. 29).

[445] N. N. Bugaenko, A. N. Gorban, and M. G. Sadovsky. *Maximum Entropy Method in Analysis of Genetic Text and Measurement of its Information Content*. 1998. DOI: 10.1023/A:1009637019316 (cit. on p. 29).

[446] B. Schölkopf, K. Tsuda, and J.-P. Vert. *Kernel Methods in Computational Biology*. MIT Press, 2004 (cit. on p. 29).

[447] D. Belazzougui and F. Cunial. "A Framework for Space-Efficient String Kernels". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 9133. 2015, pp. 13–25. DOI: 10.1007/978-3-319-19929-0_2 (cit. on p. 29).

[448] G. Marçais and C. Kingsford. "A fast, lock-free approach for efficient parallel counting of occurrences of k-mers". In: *Bioinformatics* 27.6 (2011), pp. 764–770. DOI: 10.1093/bioinformatics/btr011 (cit. on pp. 29, 91).

[449] P. Melsted and J. K. Pritchard. "Efficient counting of k-mers in DNA sequences using a bloom filter". In: *BMC Bioinformatics* 12.1 (2011), p. 333. DOI: 10.1186/1471-2105-12-333 (cit. on p. 29).

[450] G. Rizk, D. Lavenier, and R. Chikhi. "DSK: k-mer counting with very low memory usage". In: *Bioinformatics* 29.5 (2013), pp. 652–653. DOI: 10.1093/bioinformatics/btt020 (cit. on p. 29).

[451] S. Deorowicz, A. Debudaj-Grabysz, and S. Grabowski. "Disk-based k-mer counting on a PC." In: *BMC bioinformatics* 14.1 (2013), p. 160. DOI: 10.1186/1471-2105-14-160 (cit. on p. 29).

[452] R. S. Roy, D. Bhattacharya, and A. Schliep. "Turtle: Identifying frequent k-mers with cache-efficient algorithms". In: *Bioinformatics* 30.14 (2014), pp. 1950–1957. DOI: 10.1093/bioinformatics/btu132 (cit. on p. 30).

[453] Q. Zhang et al. "These Are Not the K-mers You Are Looking For: Efficient Online K-mer Counting Using a Probabilistic Data Structure". In: *PLoS ONE* 9.7 (2014), e101271. DOI: 10.1371/journal.pone.0101271 (cit. on p. 30).

[454] S. Deorowicz et al. “KMC 2: fast and resource-frugal k-mer counting”. In: *Bioinformatics* 31.10 (2015), pp. 1569–1576. DOI: 10.1093/bioinformatics/btv022 (cit. on p. 30).

[455] Y. Li and XifengYan. “MSPKmerCounter: A Fast and Memory Efficient Approach for K-mer Counting”. In: *arXiv preprints* (2015), pp. 1–7. URL: http://arxiv.org/abs/1505.06550 (cit. on p. 30).

[456] M. R. Crusoe et al. “The khmer software package: enabling efficient nucleotide sequence analysis”. In: *F1000Research* (2015), pp. 1–12. DOI: 10.12688/f1000research.6924.1 (cit. on p. 30).

[457] A.-A. Mamun, S. Pal, and S. Rajasekaran. “KCMBT: a k -mer Counter based on Multiple Burst Trees”. In: *Bioinformatics* 32.18 (2016), pp. 2783–2790. DOI: 10.1093/bioinformatics/btw345 (cit. on p. 30).

[458] D. E. Wood and S. L. Salzberg. “Kraken: ultrafast metagenomic sequence classification using exact alignments.” In: *Genome biology* 15.3 (2014), R46. DOI: 10.1186/gb-2014-15-3-r46 (cit. on pp. 30, 73–76, 80, 82, 85, 91, 96, 99, 106, 107).

[459] L. Schaeffer et al. “Pseudoalignment for metagenomic read assignment”. In: *Bioinformatics* (2017), pp. 1–9. DOI: 10.1093/bioinformatics/btx106 (cit. on pp. 30, 78).

[460] N. L. Bray et al. “Near-optimal probabilistic RNA-seq quantification.” In: *Nature biotechnology* (2016). DOI: 10.1038/nbt.3519 (cit. on pp. 30, 78).

[461] S. Burkhardt and J. Kärkkäinen. “Better Filtering with Gapped q-Grams”. In: *Combinatorial Pattern Matching*. Vol. 56. 1. 2001, pp. 73–85. DOI: 10.1007/3-540-48194-X_6 (cit. on pp. 30, 31).

[462] S. Burkhardt and J. Karkkäinen. “Better Filtering with Gapped q-Grams”. In: *Fundamenta Informaticae* 56.1-2 (2003), pp. 51–70 (cit. on pp. 30, 31, 81, 122).

[463] B. Ma, J. Tromp, and M. Li. “PatternHunter: faster and more sensitive homology search”. In: *Bioinformatics* 18.3 (2002), pp. 440–445. DOI: 10.1093/bioinformatics/18.3.440 (cit. on pp. 30, 31, 81, 122).

[464] C.-A. Leimeister et al. “Fast alignment-free sequence comparison using spaced-word frequencies”. In: *Bioinformatics* 30.14 (2014), pp. 1991–1999. DOI: 10.1093/bioinformatics/btu177 (cit. on pp. 30, 32, 81, 95).

[465] B. Morgenstern et al. “Estimating evolutionary distances between genomic sequences from spaced-word matches”. In: *Algorithms for Molecular Biology* 10.1 (2015), p. 5. DOI: 10.1186/s13015-015-0032-x (cit. on pp. 30, 32, 81, 95).

[466] T. Onodera and T. Shibuya. “The Gapped Spectrum Kernel for Support Vector Machines”. In: *Machine Learning and Data Mining in Pattern Recognition*. 2013, pp. 1–15. DOI: 10.1007/978-3-642-39712-7_1 (cit. on pp. 30, 81, 95, 96).

[467] M. Ghandi et al. “Enhanced Regulatory Sequence Prediction Using Gapped k-mer Features”. In: *PLoS Computational Biology* 10.7 (2014), e1003711. DOI: 10.1371/journal.pcbi.1003711 (cit. on pp. 30, 81, 95).

[468] L. Noé and D. E. Martin. “A Coverage Criterion for Spaced Seeds and Its Applications to Support Vector Machine String Kernels and k -Mer Distances”. In: *Journal of Computational Biology* 21.12 (2014), pp. 947–963. DOI: 10.1089/cmb.2014.0173 (cit. on pp. 30, 32, 81, 82, 89, 95, 96).

[469] B. Brejová, D. G. Brown, and T. Vinař. "Vector seeds: An extension to spaced seeds". In: *Journal of Computer and System Sciences* 70.3 (2005), pp. 364–380. DOI: 10.1016/j.jcss.2004.12.008 (cit. on p. 30).

[470] D. Mak, Y. Gelfand, and G. Benson. "Indel seeds for homology search". In: *Bioinformatics* 22.14 (2006), e341–e349. DOI: 10.1093/bioinformatics/btl263 (cit. on p. 30).

[471] "On Subset Seeds for Protein Alignment". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 6.3 (2009), pp. 483–494. DOI: 10.1109/TCBB.2009.4 (cit. on p. 30).

[472] M. Csűrös and B. Ma. "Rapid Homology Search with Two-Stage Extension and Daughter Seeds". In: *COCOON 2005 - Eleventh International Computing and Combinatorics Conference*. Vol. Springer L. 2005, pp. 104–114. DOI: 10.1007/11533719_13 (cit. on p. 30).

[473] M. Csuros and B. Ma. "Rapid Homology Search with Neighbor Seeds". In: *Algorithmica* 48.2 (2007), pp. 187–202. DOI: 10.1007/s00453-007-0062-y (cit. on p. 30).

[474] L. Noé. *Spaced seeds bibliography*. Tech. rep. URL: http://www.lifl.fr/~noe/spaced_seeds.html (cit. on pp. 30, 122).

[475] M. Li et al. "PatternHunter II: highly sensitive and fast homology search." In: *Genome informatics. International Conference on Genome Informatics* 14.03 (2003), pp. 164–75. DOI: 10.1142/S0219720004000661 (cit. on p. 31).

[476] M. Li et al. "Patternhunter II: highly sensitive and fast homology search." In: *Journal of bioinformatics and computational biology* 2.3 (2004), pp. 417–39. DOI: 10.1142/S0219720004000661 (cit. on pp. 31, 95).

[477] V. H. Nguyen and D. Lavenier. "PLAST: parallel local alignment search tool for database comparison". In: *BMC Bioinformatics* 10.1 (2009), p. 329. DOI: 10.1186/1471-2105-10-329 (cit. on p. 31).

[478] M. Startek et al. "Efficient alternatives to PSI-BLAST". In: *Bulletin of the Polish Academy of Sciences: Technical Sciences* 60.3 (2012), pp. 495–505. DOI: 10.2478/v10175-012-0063-0 (cit. on p. 31).

[479] B. Buchfink, D. H. Huson, and C. Xie. "MetaScope - Fast and accurate identification of microbes in metagenomic sequencing data". In: *arXiv preprints* (2015), pp. 1–12. URL: http://arxiv.org/abs/1511.08753 (cit. on pp. 31, 79).

[480] R. Ounit and S. Lonardi. "Higher classification sensitivity of short metagenomic reads with CLARK-S". In: *bioRxiv preprints* (2016). DOI: 10.1101/053462. URL: http://biorxiv.org/lookup/doi/10.1101/053462 (cit. on pp. 31, 77).

[481] G. Kucherov, L. Noé, and M. Roytberg. "A unifying framework for seed sensitivity and its application to subset seeds". In: *Journal of Bioinformatics and Computational Biology* 04.02 (2006), pp. 553–569. DOI: 10.1142/S0219720006001977 (cit. on pp. 31, 32, 81, 95).

[482] L. Hahn et al. "RasBhari: optimizing spaced seeds for database searching, read mapping and alignment-free sequence comparison". In: *arXiv preprints* (2015), pp. 1–17. URL: http://arxiv.org/abs/1511.04001 (cit. on pp. 31, 32).

[483] F. Nicolas and E. Rivals. "Hardness of Optimal Spaced Seed Design". In: *Combinatorial Pattern Matching*. Vol. 74. Lecture Notes in Computer Science 5. Springer Berlin Heidelberg, 2005, pp. 144–155. DOI: 10.1007/11496656_13 (cit. on pp. 31, 122).

[484] F. Nicolas and E. Rivals. "Hardness of optimal spaced seed design". In: *Journal of Computer and System Sciences* 74.5 (2008), pp. 831–849. DOI: 10.1016/j.jcss.2007.10.001 (cit. on p. 31).

[485] G. Kucherov, L. Noé, and M. Roytberg. "Multi-seed Lossless Filtration". In: *Combinatorial Pattern Matching*. 2004, pp. 297–310. DOI: 10.1007/978-3-540-27801-6_22 (cit. on p. 31).

[486] G. Kucherov, L. Noe, and M. Roytberg. "Multiseed Lossless Filtration". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 2.1 (2005), pp. 51–61. DOI: 10.1109/TCBB.2005.12 (cit. on pp. 31, 122, 131).

[487] M. Farach-Colton et al. "Optimal Spaced Seeds for Faster Approximate String Matching". In: *Automata, Languages and Programming*. 2005, pp. 1251–1262. DOI: 10.1007/11523468_101 (cit. on p. 31).

[488] M. Farach-Colton et al. "Optimal spaced seeds for faster approximate string matching". In: *Journal of Computer and System Sciences* 73.7 (2007), pp. 1035–1044. DOI: 10.1016/j.jcss.2007.03.007 (cit. on pp. 31, 122).

[489] L. Egidi and G. Manzini. "Spaced Seeds Design Using Perfect Rulers". In: *String Processing and Information Retrieval*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011, pp. 32–43. DOI: 10.1007/978-3-642-24583-1_5 (cit. on pp. 31, 122).

[490] L. Egidi and G. Manzini. "Spaced Seed Design Using Perfect Rulers". In: *Fundamenta informaticae* 131 (2014), pp. 187–203. DOI: 10.3233/FI-2014-1009 (cit. on pp. 31, 122).

[491] L. Egidi and G. Manzini. "Better spaced seeds using Quadratic Residues". In: *Journal of Computer and System Sciences* 79.7 (2013), pp. 1144–1155. DOI: 10.1016/j.jcss.2013.03.002 (cit. on pp. 31, 122).

[492] L. Egidi and G. Manzini. "Multiple seeds sensitivity using a single seed with threshold." In: *Journal of bioinformatics and computational biology* 13.4 (2015), p. 1550011. DOI: 10.1142/S0219720015500110 (cit. on pp. 31, 95).

[493] L. Egidi and G. Manzini. "Design and analysis of periodic multiple seeds". In: *Theoretical Computer Science* 522 (2014), pp. 62–76. DOI: 10.1016/j.tcs.2013.12.007 (cit. on pp. 31, 122).

[494] K. Břinda. *Lossless seeds for approximate string matching*. 2013 (cit. on pp. 31, 122, 125, 131).

[495] B. Ma and M. Li. "On the complexity of the spaced seeds". In: *Journal of Computer and System Sciences* 73.7 (2007), pp. 1024–1034. DOI: 10.1016/j.jcss.2007.03.008 (cit. on p. 32).

[496] B. Ma and H. Yao. "Seed optimization for i.i.d. similarities is no easier than optimal Golomb ruler design". In: *Information Processing Letters* 109.19 (2009), pp. 1120–1124. DOI: 10.1016/j.ipl.2009.07.008 (cit. on p. 32).

[497] U. Keich et al. “On spaced seeds for similarity search”. In: *Discrete Applied Mathematics* 138.3 (2004), pp. 253–263. DOI: 10.1016/S0166-218X(03)00382-2 (cit. on p. 32).

[498] G. Kucherov, L. Noe, and Y. Ponty. “Estimating seed sensitivity on homogeneous alignments”. In: *Proceedings. Fourth IEEE Symposium on Bioinformatics and Bioengineering*. IEEE, 2004, pp. 387–394. DOI: 10.1109/BIBE.2004.1317369 (cit. on p. 32).

[499] K. P. Choi and L. Zhang. “Sensitivity analysis and efficient method for identifying optimal spaced seeds”. In: *Journal of Computer and System Sciences* 68.1 (2004), pp. 22–40. DOI: 10.1016/j.jcss.2003.04.002 (cit. on p. 32).

[500] L. Ilie, S. Ilie, and A. M. Bigvand. “SpEED: Fast computation of sensitive spaced seeds”. In: *Bioinformatics* 27.17 (2011), pp. 2433–2434. DOI: 10.1093/bioinformatics/btr368 (cit. on p. 32).

[501] J. Buhler, U. Keich, and Y. Sun. “Designing seeds for similarity search in genomic DNA”. In: *Journal of Computer and System Sciences* 70.3 (2005), pp. 342–363. DOI: 10.1016/j.jcss.2004.12.003 (cit. on p. 32).

[502] R. Ounit and S. Lonardi. “Higher Classification Accuracy of Short Metagenomic Reads by Discriminative Spaced k-mers”. In: *Algorithms in Bioinformatics*. 15th Inter. Springer Berlin Heidelberg, 2015, pp. 286–295. DOI: 10.1007/978-3-662-48221-6_21 (cit. on p. 32).

[503] E. Bao et al. “SEED: efficient clustering of next-generation sequences”. In: *Bioinformatics* 27.18 (2011), pp. 2502–9. DOI: 10.1093/bioinformatics/btr447 (cit. on pp. 32, 85).

[504] D. Gusfield. *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*. Cambridge University Press, 2007 (cit. on pp. 33, 34).

[505] V. Makinen et al. *Genome-Scale Algorithm Design*. Cambridge: Cambridge University Press, 2015. DOI: 10.1017/CBO9781139940023 (cit. on p. 34).

[506] U. Manber and G. Myers. “Suffix Arrays: A New Method for On-Line String Searches”. In: *SIAM Journal on Computing* 22.5 (1993), pp. 935–948. DOI: 10.1137/0222058 (cit. on p. 34).

[507] M. I. Abouelhoda, S. Kurtz, and E. Ohlebusch. “Replacing suffix trees with enhanced suffix arrays”. In: *Journal of Discrete Algorithms* 2.1 SPEC. ISS. (2004), pp. 53–86. DOI: 10.1016/S1570-8667(03)00065-0 (cit. on p. 34).

[508] M. Salson et al. “Dynamic extended suffix arrays”. In: *Journal of Discrete Algorithms* 8.2 (2010), pp. 241–257. DOI: 10.1016/j.jda.2009.02.007 (cit. on pp. 34, 53, 54).

[509] A. M. S. Shrestha, M. C. Frith, and P. Horton. “A bioinformatician’s guide to the forefront of suffix array construction algorithms.” In: *Briefings in bioinformatics* 15.2 (2014), pp. 138–54. DOI: 10.1093/bib/bbt081 (cit. on p. 34).

[510] S. Gog et al. “From Theory to Practice: Plug and Play with Succinct Data Structures”. In: 2014, pp. 326–337. DOI: 10.1007/978-3-319-07959-2_28 (cit. on pp. 34, 35).

[511] N. J. Larsson and K. Sadakane. *Faster suffix sorting*. Tech. rep. 1999 (cit. on p. 34).

[512] N. J. Larsson and K. Sadakane. "Faster suffix sorting". In: *Theoretical Computer Science* 387.3 (2007), pp. 258–272. DOI: 10.1016/j.tcs.2007.07.017 (cit. on p. 34).

[513] M. Burrows and D. J. Wheeler. *A Block-sorting Lossless Data Compression Algorithm*. Tech. rep. 1994 (cit. on p. 34).

[514] G. Rosone and M. Sciortino. "The Burrows-Wheeler Transform between Data Compression and Combinatorics on Words". In: *The Nature of Computation. Logic, Algorithms, Applications: 9th Conference on Computability in Europe, CiE 2013, Milan, Italy, July 1-5, 2013. Proceedings*. 2013, pp. 353–364. DOI: 10.1007/978-3-642-39053-1_42 (cit. on p. 34).

[515] S. Mantaci, A. Restivo, and M. Sciortino. "Burrows–Wheeler transform and Sturmian words". In: *Information Processing Letters* 86.5 (2003), pp. 241–246. DOI: 10.1016/S0020-0190(02)00512-4 (cit. on p. 34).

[516] A. Restivo and G. Rosone. "Burrows–Wheeler transform and palindromic richness". In: *Theoretical Computer Science* 410.30-32 (2009), pp. 3018–3026. DOI: 10.1016/j.tcs.2009.03.008 (cit. on p. 34).

[517] A. Restivo and G. Rosone. "Balancing and clustering of words in the Burrows–Wheeler transform". In: *Theoretical Computer Science* 412.27 (2011), pp. 3019–3032. DOI: 10.1016/j.tcs.2010.11.040 (cit. on p. 34).

[518] M. Crochemore, J. Désarménien, and D. Perrin. "A note on the Burrows–Wheeler transformation". In: *Theoretical Computer Science* 332.1-3 (2005), pp. 567–572. DOI: 10.1016/j.tcs.2004.11.014 (cit. on p. 34).

[519] G. Manzini. "An analysis of the Burrows—Wheeler transform". In: *Journal of the ACM* 48.3 (2001), pp. 407–430. DOI: 10.1145/382780.382782 (cit. on p. 35).

[520] G. Kucherov, L. Tóthmérész, and S. Vialette. "On the combinatorics of suffix arrays". In: *Information Processing Letters* 113.22-24 (2013), pp. 915–920. DOI: 10.1016/j.ipl.2013.09.009 (cit. on p. 35).

[521] M. Kufleitner. "On Bijective Variants of the Burrows-Wheeler Transform". In: *arXiv preprints* (2009), p. 15. URL: http://arxiv.org/abs/0908.0239 (cit. on p. 35).

[522] J. Y. Gil and D. A. Scott. "A Bijective String Sorting Transform". In: *arXiv preprints* (2012), pp. 1–16. URL: http://arxiv.org/abs/1201.3077 (cit. on p. 35).

[523] J. W. Daykin and W. Smyth. "A bijective variant of the Burrows–Wheeler Transform using V-order". In: *Theoretical Computer Science* 531 (2014), pp. 77–89. DOI: 10.1016/j.tcs.2014.03.014 (cit. on p. 35).

[524] M. Salson et al. "A four-stage algorithm for updating a Burrows–Wheeler transform". In: *Theoretical Computer Science* 410.43 (2009), pp. 4350–4359. DOI: 10.1016/j.tcs.2009.07.016 (cit. on pp. 35, 53).

[525] D. Belazzougui et al. "Versatile Succinct Representations of the Bidirectional Burrows-Wheeler Transform". In: vol. 250345. 2013, pp. 133–144. DOI: 10.1007/978-3-642-40450-4_12 (cit. on p. 35).

[526] J. Siren. "Burrows-Wheeler Transform for Terabases". In: *2016 Data Compression Conference (DCC)*. IEEE, 2016, pp. 211–220. DOI: 10.1109/DCC.2016.17 (cit. on pp. 35, 66).

[527] P. Bonizzoni et al. "A New Lightweight Algorithm to compute the BWT and the LCP array of a Set of Strings". In: *arXiv preprints* (2016), pp. 1–18. URL: http://arxiv.org/abs/1607.08342 (cit. on p. 35).

[528] Y. Liu, T. Hankeln, and B. Schmidt. "Parallel and Space-Efficient Construction of Burrows-Wheeler Transform and Suffix Array for Big Genome Data". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 13.3 (2016), pp. 592–598. DOI: 10.1109/TCBB.2015.2430314 (cit. on pp. 35, 111).

[529] D. Adjeroh, T. Bell, and A. Mukherjee. *The Burrows-Wheeler Transform: Data Compression, Suffix Arrays, and Pattern Matching*. Boston, MA: Springer US, 2008, p. 353. DOI: 10.1007/978-0-387-78909-5 (cit. on p. 35).

[530] J. C. Na et al. "FM-index of alignment: A compressed index for similar strings". In: *Theoretical Computer Science* 638 (2016), pp. 159–170. DOI: 10.1016/j.tcs.2015.08.008 (cit. on p. 35).

[531] J. C. Na et al. "FM-index of Alignment with Gaps". In: *arXiv preprints* (2016). URL: http://arxiv.org/abs/1606.03897 (cit. on p. 35).

[532] T. W. Lam et al. "High throughput short read alignment via bi-directional BWT". In: *2009 IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2009* (2009), pp. 31–36. DOI: 10.1109/BIBM.2009.42 (cit. on p. 35).

[533] G. Kucherov, K. Salikhov, and D. Tsur. "Approximate String Matching Using a Bidirectional Index". In: *Combinatorial pattern matching*. Vol. 8486. Elsevier B.V., 2014, pp. 222–231. DOI: 10.1007/978-3-319-07566-2_23 (cit. on p. 35).

[534] C. Pockrandt, M. Ehrhardt, and K. Reinert. "Constant-time and space-efficient unidirectional and bidirectional FM-indices using EPR-dictionaries". In: *arXiv preprints* (2016), pp. 1–13. URL: http://arxiv.org/abs/1608.02413 (cit. on p. 35).

[535] D. Belazzougui et al. "Relative FM-Indexes". In: *String Processing and Information Retrieval, 21st International Symposium, SPIRE 2014, Ouro Preto, Brazil, October 20-22, 2014. Proceedings*. Lecture No. Springer International Publishing, 2014. Chap. Relative F, pp. 52–64. DOI: 10.1007/978-3-319-11918-2_6 (cit. on pp. 35, 54).

[536] W. Gerlach. "Dynamic FM-Index for a Collection of Texts with Application to Space-efficient Construction of the Compressed Suffix Array". PhD thesis. Bielefeld University, 2007 (cit. on pp. 35, 53).

[537] M. a. DePristo et al. "A framework for variation discovery and genotyping using next-generation DNA sequencing data". In: *Nature Genetics* 43.5 (2011), pp. 491–498. DOI: 10.1038/ng.806 (cit. on pp. 41, 49).

[538] G. A. Van der Auwera et al. "From FastQ Data to High-Confidence Variant Calls: The Genome Analysis Toolkit Best Practices Pipeline". In: *Current Protocols in Bioinformatics*. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2013, pp. 11.10.1–11.10.33. DOI: 10.1002/0471250953.bi1110s43 (cit. on p. 41).

[539] D. Y. C. Brandt et al. "Mapping Bias Overestimates Reference Allele Frequencies at the HLA Genes in the 1000 Genomes Project Phase I Data". In: *G3: Genes|Genomes|Genetics* 5.5 (2015), pp. 931–941. DOI: 10.1534/g3.114.015784 (cit. on p. 42).

[540] M. Holtgrewe et al. "A novel and well-defined benchmarking method for second generation read mapping." In: *BMC bioinformatics* 12.1 (2011), p. 210. DOI: 10.1186/1471-2105-12-210 (cit. on p. 43).

[541] M. Smolka et al. "Teaser: Individualized benchmarking and optimization of read mapping results for NGS data". In: *Genome Biology* 16.1 (2015), p. 235. DOI: 10.1186/s13059-015-0803-1 (cit. on p. 43).

[542] J. Köster and S. Rahmann. "Snakemake-a scalable bioinformatics workflow engine". In: *Bioinformatics* 28.19 (2012), pp. 2520–2522. DOI: 10.1093/bioinformatics/bts480 (cit. on pp. 46, 57, 96).

[543] G. T. Marth et al. "A general approach to single-nucleotide polymorphism discovery". In: *Nature Genetics* 23.4 (1999), pp. 452–456. DOI: 10.1038/70570 (cit. on pp. 49, 51).

[544] R. Li et al. "SNP detection for massively parallel whole-genome resequencing." In: *Genome research* 19.6 (2009), pp. 1124–32. DOI: 10.1101/gr.088013.108 (cit. on p. 49).

[545] D. C. Koboldt et al. "VarScan: variant detection in massively parallel sequencing of individual and pooled samples." In: *Bioinformatics* 25.17 (2009), pp. 2283–5. DOI: 10.1093/bioinformatics/btp373 (cit. on p. 49).

[546] R. Nielsen et al. "Genotype and SNP calling from next-generation sequencing data". In: *Nature Reviews Genetics* 12.6 (2011), pp. 443–451. DOI: 10.1038/nrg2986 (cit. on p. 49).

[547] C. A. Albers et al. "Dindel: accurate indel calls from short-read data." In: *Genome research* 21.6 (2011), pp. 961–73. DOI: 10.1101/gr.112326.110 (cit. on p. 49).

[548] F. Xu et al. "A fast and accurate SNP detection algorithm for next-generation sequencing data." In: *Nature communications* 3 (2012), p. 1258. DOI: 10.1038/ncomms2256 (cit. on p. 49).

[549] E. Garrison and G. Marth. "Haplotype-based variant detection from short-read sequencing". In: *arXiv preprints* (2012), p. 9. URL: http://arxiv.org/abs/1207.3907 (cit. on pp. 49, 51).

[550] R. A. Farrer et al. "Using False Discovery Rates to Benchmark SNP-callers in next-generation sequencing projects". In: *Scientific Reports* 3 (2013), pp. 1–6. DOI: 10.1038/srep01512 (cit. on p. 49).

[551] P. Danecek et al. "The variant call format and VCFtools". In: *Bioinformatics* 27.15 (2011), pp. 2156–2158. DOI: 10.1093/bioinformatics/btr330 (cit. on p. 49).

[552] P. Ferragina and G. Manzini. "Indexing compressed text". In: *Journal of the ACM* 52.4 (2005), pp. 552–581. DOI: 10.1145/1082036.1082039 (cit. on p. 53).

[553] N. Gupta et al. "Efficient Index Maintenance Under Dynamic Genome Modification". In: *arXiv preprints* (2016), pp. 1–12. URL: http://arxiv.org/abs/1604.03132 (cit. on pp. 54, 67).

[554] A. Ghanayim and D. Geiger. *Iterative Referencing for Improving the Interpretation of DNA Sequence Data*. Tech. rep. Israel: Computer Science Department, Technion, 2013 (cit. on p. 56).

[555] B. E. Dutilh, M. A. Huynen, and M. Strous. "Increasing the coverage of a metapopulation consensus genome by iterative read mapping and assembly". In: *Bioinformatics* 25.21 (2009), pp. 2878–2881. DOI: 10.1093/bioinformatics/btp377 (cit. on p. 56).

[556] T. D. Otto et al. "Iterative correction of reference Nucleotides (iCORN) using second generation sequencing technology". In: *Bioinformatics* 26.14 (2010), pp. 1704–1707. DOI: 10.1093/bioinformatics/btq269 (cit. on p. 56).

[557] I. J. Tsai, T. D. Otto, and M. Berriman. "Improving draft assemblies by iterative mapping and assembly of short reads to eliminate gaps". In: *Genome Biology* 11.4 (2010), R41. DOI: 10.1186/gb-2010-11-4-r41 (cit. on p. 56).

[558] B. E. Dutilh et al. "Iterative Read Mapping and Assembly Allows the Use of a More Distant Reference in Metagenome Assembly". In: *Handbook of Molecular Microbial Ecology I*. Vol. I. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2011, pp. 379–385. DOI: 10.1002/9781118010518.ch43 (cit. on p. 56).

[559] H. Tae et al. "Improved variation calling via an iterative backbone remapping and local assembly method for bacterial genomes". In: *Genomics* 100.5 (2012), pp. 271–276. DOI: 10.1016/j.ygeno.2012.07.015 (cit. on p. 56).

[560] C. Hahn, L. Bachmann, and B. Chevreux. "Reconstructing mitochondrial genomes directly from genomic next-generation sequencing reads–a baiting and iterative mapping approach." In: *Nucleic acids research* 41.13 (2013), e129. DOI: 10.1093/nar/gkt371 (cit. on p. 56).

[561] K. McElroy, T. Thomas, and F. Luciani. "Deep sequencing of evolving pathogen populations: applications, errors, and bioinformatic solutions". In: *Microbial Informatics and Experimentation* 4.1 (2014), p. 1. DOI: 10.1186/2042-5783-4-1 (cit. on p. 56).

[562] T. Hackl et al. "proovread: large-scale high-accuracy PacBio correction through iterative short read consensus". In: *Bioinformatics* 30.21 (2014), pp. 3004–3011. DOI: 10.1093/bioinformatics/btu392 (cit. on pp. 56, 65).

[563] B. M. P. Verbist et al. "VirVarSeq: a low-frequency virus variant detection pipeline for Illumina sequencing using adaptive base-calling accuracy filtering". In: *Bioinformatics* 31.1 (2015), pp. 94–101. DOI: 10.1093/bioinformatics/btu587 (cit. on p. 56).

[564] H. Ode et al. "Quasispecies Analyses of the HIV-1 Near-full-length Genome With Illumina MiSeq". In: *Frontiers in Microbiology* 6.November (2015), pp. 1–11. DOI: 10.3389/fmicb.2015.01258 (cit. on p. 56).

[565] H. Li. "A statistical framework for SNP calling, mutation discovery, association mapping and population genetical parameter estimation from sequencing data". In: *Bioinformatics* 27.21 (2011), pp. 2987–2993. DOI: 10.1093/bioinformatics/btr509 (cit. on p. 57).

[566] O. Tange. "GNU Parallel: the command-line power tool". In: *;login: The USENIX Magazine* 36.1 (2011), pp. 42–47 (cit. on p. 57).

[567] H. Li. "Improving SNP discovery by base alignment quality". In: *Bioinformatics* 27.8 (2011), pp. 1157–1158. DOI: 10.1093/bioinformatics/btr076 (cit. on p. 60).

[568] P. J. A. Cock et al. "SAM/BAM format v1.5 extensions for de novo assemblies". In: *bioRxiv preprints* (2015), p. 020024. DOI: 10.1101/020024. URL: http://biorxiv.org/lookup/doi/10.1101/020024 (cit. on p. 63).

[569] D. C. Koboldt et al. "VarScan 2: somatic mutation and copy number alteration discovery in cancer by exome sequencing." In: *Genome research* 22.3 (2012), pp. 568–76. DOI: 10.1101/gr.129684.111 (cit. on pp. 64, 65).

[570] V. Boeva et al. "Control-FREEC: a tool for assessing copy number and allelic content using next-generation sequencing data". In: *Bioinformatics* 28.3 (2012), pp. 423–425. DOI: 10.1093/bioinformatics/btr670 (cit. on pp. 64, 65).

[571] C. T. Brown et al. "A Reference-Free Algorithm for Computational Normalization of Shotgun Sequencing Data". In: *arXiv preprints* (2012), pp. 1–18. URL: http://arxiv.org/abs/1203.4802 (cit. on p. 64).

[572] A. Roberts and L. Pachter. "Streaming fragment assignment for real-time analysis of sequencing experiments". In: *Nature Methods* 10.1 (2012), pp. 71–73. DOI: 10.1038/nmeth.2251 (cit. on p. 64).

[573] P. Melsted and B. V. Halldorsson. "KmerStream: streaming algorithms for k-mer abundance estimation". In: *Bioinformatics* 30.24 (2014), pp. 3541–3547. DOI: 10.1093/bioinformatics/btu713 (cit. on p. 64).

[574] Q. Zhang, S. Awad, and C. Brown. *Crossing the streams : a framework for streaming analysis of short DNA sequencing reads*. 2015. DOI: 10.7287/peerj.preprints.890.v1 (cit. on p. 64).

[575] M. D. Cao et al. "Scaffolding and Completing Genome Assemblies in Real-time with Nanopore Sequencing". In: *bioRxiv preprints* (2016). DOI: 10.1101/054783. URL: http://biorxiv.org/lookup/doi/10.1101/054783 (cit. on p. 64).

[576] M. D. Cao et al. "Streaming algorithms for identification of pathogens and antibiotic resistance potential from real-time MinIONTM sequencing". In: *GigaScience* 5.1 (2016), p. 32. DOI: 10.1186/s13742-016-0137-2 (cit. on p. 64).

[577] M. D. Cao et al. "Realtime analysis and visualization of MinION sequencing data with npReader". In: *Bioinformatics* 32.5 (2016), pp. 764–766. DOI: 10.1093/bioinformatics/btv658 (cit. on p. 64).

[578] M. Gerstung et al. "Reliable detection of subclonal single-nucleotide variants in tumour cell populations". In: *Nature Communications* 3 (2012), p. 811. DOI: 10.1038/ncomms1814 (cit. on p. 65).

[579] R. Goya et al. "SNVMix: predicting single nucleotide variants from next-generation sequencing of tumors." In: *Bioinformatics* 26.6 (2010), pp. 730–6. DOI: 10.1093/bioinformatics/btq040 (cit. on p. 65).

[580] A. Roth et al. "JointSNVMix: a probabilistic model for accurate detection of somatic mutations in normal/tumour paired next-generation sequencing data." In: *Bioinformatics* 28.7 (2012), pp. 907–13. DOI: 10.1093/bioinformatics/bts053 (cit. on p. 65).

[581] C. T. Saunders et al. "Strelka: accurate somatic small-variant calling from sequenced tumor-normal sample pairs." In: *Bioinformatics* 28.14 (2012), pp. 1811–7. DOI: 10.1093/bioinformatics/bts271 (cit. on p. 65).

[582] D. E. Larson et al. “SomaticSniper: identification of somatic point mutations in whole genome sequencing data.” In: *Bioinformatics* 28.3 (2012), pp. 311–7. DOI: 10.1093/bioinformatics/btr665 (cit. on p. 65).

[583] E. Bareke et al. “A novel mathematical basis for predicting somatic single nucleotide variants from next-generation sequencing”. In: *Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Biomedicine - BCB ’12*. BCB ’12. New York, New York, USA: ACM Press, 2012, pp. 533–535. DOI: 10.1145/2382936.2383012 (cit. on p. 65).

[584] J. Bian et al. “SNVHMM: predicting single nucleotide variants from next generation sequencing.” In: *BMC bioinformatics* 14.1 (2013), p. 225. DOI: 10.1186/1471-2105-14-225 (cit. on p. 65).

[585] J. F. Sathirapongsasuti et al. “Exome sequencing-based copy-number variation and loss of heterozygosity detection: ExomeCNV”. In: *Bioinformatics* 27.19 (2011), pp. 2648–2654. DOI: 10.1093/bioinformatics/btr462 (cit. on p. 65).

[586] V. Boeva et al. “Control-free calling of copy number alterations in deep-sequencing data using GC-content normalization”. In: *Bioinformatics* 27.2 (2011), pp. 268–269. DOI: 10.1093/bioinformatics/btq635 (cit. on p. 65).

[587] S. A. Forbes et al. “COSMIC: exploring the world’s knowledge of somatic mutations in human cancer.” In: *Nucleic acids research* 43.Database issue (2015), pp. D805–11. DOI: 10.1093/nar/gku1075 (cit. on p. 65).

[588] D. Laehnemann, A. Borkhardt, and A. C. McHardy. “Denoising DNA deep sequencing data-high-throughput sequencing errors and their correction.” In: *Briefings in bioinformatics* 17.1 (2016), pp. 154–79. DOI: 10.1093/bib/bbv029 (cit. on p. 65).

[589] N. J. Loman, J. Quick, and J. T. Simpson. “A complete bacterial genome assembled de novo using only nanopore sequencing data”. In: *Nature Methods* 12.8 (2015), pp. 733–735. DOI: 10.1038/nmeth.3444 (cit. on p. 65).

[590] M. Pop et al. “Comparative genome assembly.” In: *Briefings in bioinformatics* 5.3 (2004), pp. 237–48 (cit. on p. 65).

[591] T. Rausch et al. “A consistency-based consensus algorithm for de novo and reference-guided sequence assembly of short reads”. In: *Bioinformatics* 25.9 (2009), pp. 1118–1124. DOI: 10.1093/bioinformatics/btp131 (cit. on p. 65).

[592] K. Schneeberger et al. “Reference-guided assembly of four diverse Arabidopsis thaliana genomes”. In: *Proceedings of the National Academy of Sciences* 108.25 (2011), pp. 10249–10254. DOI: 10.1073/pnas.1107739108 (cit. on p. 65).

[593] S. Gnerre et al. “Assisted assembly: how to improve a de novo genome assembly by using related species”. In: *Genome Biology* 10.8 (2009), R88. DOI: 10.1186/gb-2009-10-8-r88 (cit. on p. 65).

[594] F. Vezzi, F. Cattonaro, and A. Policriti. “e-RGA: enhanced Reference Guided Assembly of Complex Genomes”. In: *EMBnet.journal* 17.1 (2011), p. 46. DOI: 10.14806/ej.17.1.208 (cit. on p. 65).

[595] J. Nijkamp et al. “Integrating genome assemblies with MAIA”. In: *Bioinformatics* 27.13 (2011), pp. i433–i439. DOI: 10.1093/bioinformatics/btq366 (cit. on p. 65).

[596] G. G. Silva et al. "Combining de novo and reference-guided assembly with scaffold_builder". In: *Source Code for Biology and Medicine* 8.1 (2013), p. 23. DOI: 10.1186/1751-0473-8-23 (cit. on p. 65).

[597] E. Bao, T. Jiang, and T. Girke. "AlignGraph: algorithm for secondary de novo genome assembly guided by closely related references". In: *Bioinformatics* 30.12 (2014), pp. i319–i328. DOI: 10.1093/bioinformatics/btu291 (cit. on p. 65).

[598] F. Bertels et al. "Automated Reconstruction of Whole-Genome Phylogenies from Short-Sequence Reads". In: *Molecular Biology and Evolution* 31.5 (2014), pp. 1077–1088. DOI: 10.1093/molbev/msu088 (cit. on p. 66).

[599] A. Dilthey et al. "Improved genome inference in the MHC using a population reference graph". In: *Nature Genetics* 47.6 (2015), pp. 682–688. DOI: 10.1038/ng.3257 (cit. on p. 66).

[600] N. Nguyen et al. "Building a Pan-Genome Reference for a Population". In: *Journal of Computational Biology* 22.5 (2015), pp. 387–401. DOI: 10.1089/cmb.2014.0146 (cit. on p. 66).

[601] *Global Alliance for Genomics and Health: Human Genome Variation Map (HGVM) Pilot Project*. 2016 (cit. on p. 66).

[602] J. C. Venter. "Environmental Genome Shotgun Sequencing of the Sargasso Sea". In: *Science* 304.5667 (2004), pp. 66–74. DOI: 10.1126/science.1093857 (cit. on p. 73).

[603] E. Karsenti et al. "A Holistic Approach to Marine Eco-Systems Biology". In: *PLoS Biology* 9.10 (2011), e1001177. DOI: 10.1371/journal.pbio.1001177 (cit. on p. 73).

[604] T. M. Vogel et al. "TerraGenome: a consortium for the sequencing of a soil metagenome". In: *Nature Reviews Microbiology* 7.4 (2009), pp. 252–252. DOI: 10.1038/nrmicro2119 (cit. on p. 73).

[605] J. Peterson et al. "The NIH Human Microbiome Project". In: *Genome Research* 19.12 (2009), pp. 2317–2323. DOI: 10.1101/gr.096651.109 (cit. on p. 73).

[606] Anon. "Metagenomics versus Moore's law". In: *Nature Methods* 6.9 (2009), pp. 623–623. DOI: 10.1038/nmeth0909-623 (cit. on p. 73).

[607] X. Zhang et al. "Reading the Underlying Information From Massive Metagenomic Sequencing Data". In: *Proceedings of the IEEE* (2016), pp. 1–15. DOI: 10.1109/JPROC.2016.2604406 (cit. on pp. 73, 75).

[608] D. H. Huson et al. "Integrative analysis of environmental sequences using MEGAN4". In: *Genome Research* 21.9 (2011), pp. 1552–1560. DOI: 10.1101/gr.120618.111 (cit. on pp. 73, 78).

[609] A. Brady and S. Salzberg. "PhymmBL expanded: confidence scores, custom databases, parallelization and more". In: *Nature Methods* 8.5 (2011), pp. 367–367. DOI: 10.1038/nmeth0511-367 (cit. on pp. 74, 78).

[610] S. Lindgreen, K. L. Adair, and P. P. Gardner. "An evaluation of the accuracy and speed of metagenome analysis tools". In: *Scientific Reports* 6 (2016), p. 19233. DOI: 10.1038/srep19233 (cit. on pp. 74, 75, 99).

[611] H. Teeling and F. O. Glockner. "Current opportunities and challenges in microbial metagenome analysis–a bioinformatic perspective". In: *Briefings in Bioinformatics* 13.6 (2012), pp. 728–742. DOI: 10.1093/bib/bbs039 (cit. on p. 74).

[612] S. Vinga. "Editorial: Alignment-free methods in computational biology". In: *Briefings in Bioinformatics* 15.3 (2014), pp. 341–342. DOI: 10.1093/bib/bbu005 (cit. on pp. 74, 82).

[613] K. Vervier et al. "Large-scale machine learning for metagenomics sequence classification". In: *Bioinformatics* 32.7 (2016), pp. 1023–1032. DOI: 10.1093/bioinformatics/btv683 (cit. on pp. 75, 77).

[614] S. K. Ames et al. "Scalable metagenomic taxonomy classification using a reference genome database". In: *Bioinformatics* 29.18 (2013), pp. 2253–2260. DOI: 10.1093/bioinformatics/btt389 (cit. on pp. 74–76, 80, 85, 96).

[615] R. J. Randle-Boggis et al. "Evaluating techniques for metagenome annotation using simulated sequence data". In: *FEMS Microbiology Ecology* 92.7 (2016), fiw095. DOI: 10.1093/femsec/fiw095 (cit. on p. 75).

[616] D. O. Ricke, A. Shcherbina, and N. Chiu. "Evaluating performance of metagenomic characterization algorithms using in silico datasets generated with FASTQSim". In: *bioRxiv* (2016), p. 046532. DOI: 10.1101/046532. URL: http://biorxiv.org/content/early/2016/03/31/046532.abstract (cit. on p. 75).

[617] M. A. Peabody et al. "Evaluation of shotgun metagenomics sequence classification methods using in silico and in vitro simulated communities". In: *BMC Bioinformatics* 16.1 (2015), p. 363. DOI: 10.1186/s12859-015-0788-5 (cit. on p. 75).

[618] H. Vinje et al. "Comparing K-mer based methods for improved classification of 16S sequences". In: *BMC Bioinformatics* 16.1 (2015), p. 205. DOI: 10.1186/s12859-015-0647-4 (cit. on p. 75).

[619] Pavlopoulos et al. "Metagenomics: Tools and Insights for Analyzing Next-Generation Sequencing Data Derived from Biodiversity Studies". In: *Bioinformatics and Biology Insights* (2015), p. 75. DOI: 10.4137/BBI.S12462 (cit. on p. 75).

[620] J. Jacob. *Pipelines for pathogen identification*. Tech. rep. 2016. URL: https://goo.gl/IFCLdm (cit. on p. 75).

[621] S. N. Gardner et al. "Searching more genomic sequence with less memory for fast and accurate metagenomic profiling". In: *bioRxiv preprints* (2016). DOI: 10.1101/036681. URL: http://biorxiv.org/lookup/doi/10.1101/036681 (cit. on p. 76).

[622] D. Kim et al. "Centrifuge: rapid and sensitive classification of metagenomic sequences". In: *Genome Research* 26.12 (2016), pp. 1721–1729. DOI: 10.1101/gr.210641.116 (cit. on pp. 76, 80, 112).

[623] P. Menzel, K. L. Ng, and A. Krogh. "Fast and sensitive taxonomic classification for metagenomics with Kaiju". In: *Nature Communications* 7 (2016), p. 11257. DOI: 10.1038/ncomms11257 (cit. on pp. 77, 80).

[624] R. Ounit et al. "CLARK: fast and accurate classification of metagenomic and genomic sequences using discriminative k-mers". In: *BMC Genomics* 16 (2015). DOI: 10.1186/s12864-015-1419-2 (cit. on pp. 77, 80).

[625] S. S. Minot, N. Krumm, and N. B. Greenfield. "One Codex: A Sensitive and Accurate Data Platform for Genomic Microbial Identification". In: *bioRxiv preprints* (2015), p. 027607. DOI: 10.1101/027607. URL: http://biorxiv.org/lookup/doi/10.1101/027607 (cit. on pp. 77, 80).

[626] J. Kawulok and S. Deorowicz. "CoMeta: Classification of Metagenomes Using k-mers". In: *PLOS ONE* 10.4 (2015), e0121453. DOI: 10.1371/journal.pone.0121453 (cit. on pp. 77, 80, 106).

[627] A. Y. Lee, C. S. Lee, and R. N. Van Gelder. "Scalable metagenomics alignment research tool (SMART): a scalable, rapid, and complete search heuristic for the classification of metagenomic sequences from complex sequence populations". In: *BMC Bioinformatics* 17.1 (2016), p. 292. DOI: 10.1186/s12859-016-1159-6 (cit. on pp. 77, 80, 112).

[628] N. Segata et al. "Metagenomic microbial community profiling using unique clade-specific marker genes". In: *Nature Methods* 9.8 (2012), pp. 811–814. DOI: 10.1038/nmeth.2066 (cit. on p. 77).

[629] D. T. Truong et al. "MetaPhlAn2 for enhanced metagenomic taxonomic profiling". In: *Nature Methods* 12.10 (2015), pp. 902–903. DOI: 10.1038/nmeth.3589 (cit. on p. 77).

[630] D. H. Huson et al. "MEGAN analysis of metagenomic data". In: *Genome Research* 17.3 (2007), pp. 377–386. DOI: 10.1101/gr.5969107 (cit. on p. 78).

[631] B. Liu et al. "MetaPhyler: Taxonomic profiling for metagenomic sequences". In: *2010 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE, 2010, pp. 95–100. DOI: 10.1109/BIBM.2010.5706544 (cit. on p. 78).

[632] Y. Wang et al. "MetaCluster-TA: taxonomic annotation for metagenomic data based on assembly-assisted binning". In: *BMC Genomics* 15.Suppl 1 (2014), S12. DOI: 10.1186/1471-2164-15-S1-S12 (cit. on p. 78).

[633] A. A. Metwally et al. "WEVOTE: Weighted Voting Taxonomic Identification Method of Microbial Sequences". In: *bioRxiv preprints* (2016), pp. 1–22. DOI: 10.1101/054205. URL: http://biorxiv.org/lookup/doi/10.1101/054205 (cit. on p. 78).

[634] I. Borozan, S. Watt, and V. Ferretti. "Integrating alignment-based and alignment-free sequence similarity measures for biological sequence classification". In: *Bioinformatics* 31.9 (2015), pp. 1396–1404. DOI: 10.1093/bioinformatics/btv006 (cit. on p. 78).

[635] I. Borozan and V. Ferretti. "CSSSCL: a python package that uses combined sequence similarity scores for accurate taxonomic classification of long and short sequence reads." In: *Bioinformatics* 32.3 (2016), pp. 453–5. DOI: 10.1093/bioinformatics/btv587 (cit. on p. 78).

[636] A. Brady and S. L. Salzberg. "Phymm and PhymmBL: metagenomic phylogenetic classification with interpolated Markov models". In: *Nature Methods* 6.9 (2009), pp. 673–676. DOI: 10.1038/nmeth.1358 (cit. on p. 78).

[637] S. L. Salzberg et al. "Microbial gene identification using interpolated Markov models." In: *Nucleic acids research* 26.2 (1998), pp. 544–8. DOI: 10.1093/nar/26.2.544 (cit. on p. 78).

[638] T. A. K. Freitas et al. “Accurate read-based metagenome characterization using a hierarchical suite of unique signatures”. In: *Nucleic Acids Research* (2015), pp. 1–14. DOI: 10.1093/nar/gkv180 (cit. on p. 78).

[639] J. Lu et al. “Bracken: Estimating species abundance in metagenomics data”. In: *bioRxiv preprints* (2016), pp. 1–14. DOI: 10.1101/051813. URL: http://biorxiv.org/lookup/doi/10.1101/051813 (cit. on p. 78).

[640] D. Koslicki and D. Falush. *MetaPalette: A K-mer painting approach for metagenomic taxonomic profiling and quantification of novel strain variation*. Tech. rep. 2016. DOI: 10.1128/mSystems.00020-16 (cit. on p. 78).

[641] D. H. Huson et al. “MEGAN Community Edition - Interactive Exploration and Analysis of Large-Scale Microbiome Sequencing Data”. In: *PLOS Computational Biology* 12.6 (2016), e1004957. DOI: 10.1371/journal.pcbi.1004957 (cit. on p. 79).

[642] S. C. Bayliss et al. “Bayesian identification of bacterial strains from sequencing data”. In: *Microbial Genomics* 2.8 (2016), pp. 1–16. DOI: 10.1099/mgen.0.000075 (cit. on p. 79).

[643] T. H. Dadi, B. Y. Renard, and H. Lothar. “SLIMM : Species Level Identification of Microorganisms from Metagenomes”. In: *PeerJ Preprints* (2016), pp. 1–9. DOI: 10.7287/peerj.preprints.2378v1 (cit. on p. 79).

[644] D. Koslicki, S. Foucart, and G. Rosen. “WGSQuikr: Fast Whole-Genome Shotgun Metagenomic Classification”. In: *PLoS ONE* 9.3 (2014), e91784. DOI: 10.1371/journal.pone.0091784 (cit. on p. 79).

[645] A. Sobih, A. I. Tomescu, and V. Mäkinen. “MetaFlow: Metagenomic Profiling Based on Whole-Genome Coverage Analysis with Min-Cost Flows”. In: *Research in Computational Molecular Biology: 20th Annual Conference, RECOMB 2016, Santa Monica, CA, USA, April 17-21, 2016, Proceedings*. 2016, pp. 111–121. DOI: 10.1007/978-3-319-31957-5_8 (cit. on p. 79).

[646] G. Benoit et al. “Multiple comparative metagenomics using multiset k -mer counting”. In: *PeerJ Computer Science* 2 (2016), e94. DOI: 10.7717/peerj-cs.94 (cit. on p. 79).

[647] B. D. Ondov et al. “Mash: fast genome and metagenome distance estimation using MinHash”. In: *Genome Biology* 17.1 (2016), p. 132. DOI: 10.1186/s13059-016-0997-x (cit. on p. 79).

[648] V. I. Ulyantsev et al. “MetaFast: fast reference-free graph-based comparison of shotgun metagenomic data.” In: *Bioinformatics* 32.18 (2016), pp. 2760–7. DOI: 10.1093/bioinformatics/btw312 (cit. on p. 79).

[649] K. D. Murray et al. “kWIP: The k-mer Weighted Inner Product, a de novo estimator of genetic similarity”. In: *bioRxiv preprints* (2016), pp. 1–15. DOI: 10.1101/075481. URL: http://biorxiv.org/lookup/doi/10.1101/075481 (cit. on p. 80).

[650] L. Noé and G. Kucherov. “Improved hit criteria for DNA local alignment”. In: *BMC bioinformatics* 5.1 (2004), p. 149. DOI: 10.1186/1471-2105-5-149 (cit. on p. 82).

[651] G. Benson and D. Y. F. Mak. “Exact Distribution of a Spaced Seed Statistic for DNA Homology Detection”. In: *String Processing and Information Retrieval*. Vol. 5280 LNCS. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 282–293. DOI: 10.1007/978-3-540-89097-3_27 (cit. on p. 82).

[652] A. Ben-Hur et al. “Support vector machines and kernels for computational biology”. In: *PLoS Computational Biology* 4.10 (2008). DOI: 10.1371/journal.pcbi.1000173 (cit. on p. 83).

[653] E. Tortoli. “Phylogeny of the genus Mycobacterium: many doubts, few certainties.” In: *Infection, genetics and evolution : journal of molecular epidemiology and evolutionary genetics in infectious diseases* 12.4 (2012), pp. 827–31. DOI: 10.1016/j.meegid.2011.05.025 (cit. on p. 85).

[654] E. Helgason et al. “Bacillus anthracis, Bacillus cereus, and Bacillus thuringiensis—One Species on the Basis of Genetic Evidence”. In: *Applied and Environmental Microbiology* 66.6 (2000), pp. 2627–2630. DOI: 10.1128/AEM.66.6.2627-2630.2000 (cit. on p. 87).

[655] L. Alcaraz et al. “Understanding the evolutionary relationships and major traits of Bacillus through comparative genomics”. In: *BMC Genomics* 11.1 (2010), p. 332. DOI: 10.1186/1471-2164-11-332 (cit. on p. 87).

[656] A. F. Auch et al. “Digital DNA-DNA hybridization for microbial species delineation by means of genome-to-genome sequence comparison”. In: *Standards in Genomic Sciences* 2.1 (2010), pp. 117–134. DOI: 10.4056/sigs.531120 (cit. on p. 87).

[657] Y. Sun and J. Buhler. “Designing Multiple Simultaneous Seeds for DNA Similarity Search”. In: *Journal of Computational Biology* 12.6 (2005), pp. 847–861. DOI: 10.1089/cmb.2005.12.847 (cit. on p. 95).

[658] Louxin Zhang. “Superiority of Spaced Seeds for Homology Search”. In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 4.3 (2007), pp. 496–505. DOI: 10.1109/tcbb.2007.1013 (cit. on p. 97).

[659] R. Chikhi et al. “On the Representation of de Bruijn Graphs”. In: *Research in Computational Molecular Biology: 18th Annual International Conference, RECOMB 2014, Pittsburgh, PA, USA, April 2-5, 2014, Proceedings*. Vol. 8394 LNBI. 2014, pp. 35–55. DOI: 10.1007/978-3-319-05269-4_4 (cit. on p. 102).

[660] R. Chikhi et al. “On the Representation of De Bruijn Graphs”. In: *Journal of Computational Biology* 22.5 (2015), pp. 336–352. DOI: 10.1089/cmb.2014.0160 (cit. on p. 102).

[661] K. Salikhov. “Efficient data structures for storing and indexing DNA sequences”. To appear. PhD thesis. Université Paris-Est (cit. on pp. 104, 111).

[662] T. Tatusova et al. “RefSeq microbial genomes database: new representation and annotation strategy”. In: *Nucleic Acids Research* 42.D1 (2014), pp. D553–D559. DOI: 10.1093/nar/gkt1274 (cit. on pp. 107–109).

[663] T. Gagie, G. Manzini, and D. Valenzuela. “Compressed Spaced Suffix Arrays”. In: *arXiv preprints* (2014), pp. 1–9. URL: http://arxiv.org/abs/1312.3422 (cit. on p. 111).

[664] H. Wang et al. “BWTCP: A Parallel Method for Constructing BWT in Large Collection of Genomic Reads”. In: *High Performance Computing: 30th International Conference, ISC High Performance 2015, Frankfurt, Germany, July 12-16, 2015, Proceedings*. 2015, pp. 171–178. DOI: 10.1007/978-3-319-20119-1_13 (cit. on p. 111).

[665] H. Li. *Private communication*. 2016. URL: https://sourceforge.net/p/bio-bwa/mailman/message/35190726/ (cit. on p. 111).

[666] J. T. Simpson and R. Durbin. "Efficient de novo assembly of large genomes using compressed data structures." In: *Genome research* 22.3 (2012), pp. 549–56. DOI: 10.1101/gr.126953.111 (cit. on p. 112).

[667] *Combinatorics on words*. 2nd ed. Cambridge: Cambridge University Press, 1997. DOI: 10.1017/CBO9780511566097 (cit. on p. 123).

[668] D. Lind and B. Marcus. *An Introduction to Symbolic Dynamics and Coding*. Vol. 37. Cambridge: Cambridge University Press, 1995. DOI: 10.1017/CBO9780511626302 (cit. on p. 123).