Scuola di dottorato
Università degli Studi di Milano-Bicocca

Dipartimento di Informatica Sistemistica e Comunicazione – DISCo

PhD program in Computer Science – Cycle XXIX

Inferring Genomic Variants and their Evolution

Combinatorial Optimization for Haplotype Assembly and Quantification of Intra-Tumor Heterogeneity

PhD Thesis of

Simone Zaccaria

718549

Advisors: prof. Paola Bonizzoni (Univ. di Milano-Bicocca)
prof. Ben Raphael (Brown University)

Tutor: prof. Alberto Leporati
Coordinator: prof. Stefania Bandini

Academic Year 2015–2016

Contents

Chapter 1

Introduction

> An abstraction is one thing that represents several real things equally well.
>
> Edsger W. Dijkstra (1930-2002)

Computational biology is a field of computer science aiming to model biological processes and to design algorithms for answering the questions that arise from biological data. Recent technological advances have led to an unprecedented growth of biological data. In particular, sequencing technologies are currently able to provide a huge amount of small fragments of sequences of DNA, called *reads*. DNA, represented as a string on a four-letter alphabet $\Sigma = \{A, C, G, T\}$, encodes the genetic information that uniquely characterizes any human individual and this unique complement of information defines the individual's *genome* contained in every cell. The variation in the human population is due to *genomic variants* that distinguish the DNA of an individual from any other. Genomic variants correspond to differences either in single positions or involving larger portions of DNA. Studying the relationship between genomic variants and observable traits–*phenotypic traits*–has several applications to genome medicine, such as drug design, study of disease susceptibility, and analysis of evolutionary processes [1, 19, 64, 146, 152, 153]. Unfortunately, the current data do not offer a complete and exact view of an individual's genome. In fact, sequencing technologies consider millions of cells together and produce from DNA sequences of unknown origin many reads that contain errors and whose length is orders of magnitude lower than the length of the source (Figure 1.1) [22, 106, 140]. Moreover, to understand the relative position of the reads, they are typically mapped to a reference human genome through a process of alignment (Figure 1.1) [82, 94]. As such, one of the key challenges of computational

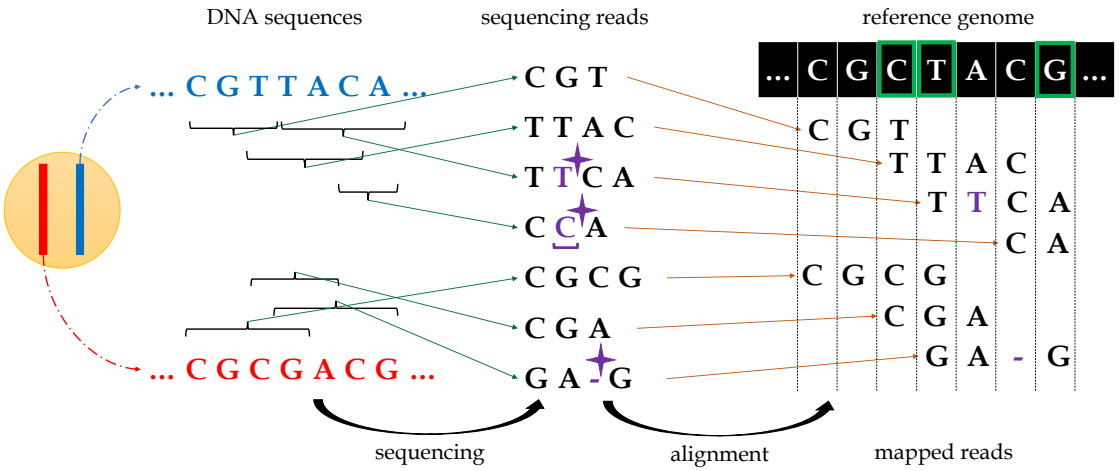

Figure 1.1: **Sequencing and mapping reads.** In the example, a cell is represented in yellow and contains two distinct sequences of DNA, one red and one blue. The process of sequencing produces a collection of sequencing reads from both the DNA sequences. Each read is a short fragment of the source and may be affected by errors, reported in purple with purple stars, that can either "substitute" characters or "insert" false characters or "delete" true characters. To know the relative position, the reads are mapped to a reference genome through a process of alignment. Single-point genomic variants are identified from the mapping of the reads. Therefore, SNP positions, represented as green boxes, are identified as positions affected by single-point mutations across a population of individuals.

biology is the inference of the genomic variants in the DNA of an individual from mapped sequencing reads [17, 18, 84, 122, 153].

The genomic variants of individuals within and across species are the result of Darwinian evolution [30, 144]. This process has run its course for billions of years and beneficial genetic changes, leading to better adaptability to the environment, have been passed on from parents to their offspring through the *germ line*, the reproductive cells of an organism. Genomic variants result from *mutations* that affect the DNA sequence of certain cells. We distinguish two types of mutations. *Germinal mutations* occur in the germ line, resulting in *germinal variants* of the genome. *Single-Nucleotide Polymorphisms (SNPs)* are the most common form of germinal variants and they are classified as point mutations since they affect single positions of a DNA sequence (Figure 1.2). SNPs can be easily identified from mapped sequencing reads by determining the presence of different values, or *alleles*, in the same SNP positions among the reads (Figure 1.1) [112]. Conversely, *somatic mutations* occur in a single somatic cell of an organism and result in *somatic variants* of the genome. One class of important somatic mutations are *Copy-Number Aberrations (CNAs)* that affect the number of copies of genomic segments by amplifying or deleting large genomic regions (Figure 1.2). In fact, CNAs are ubiquitous in can-

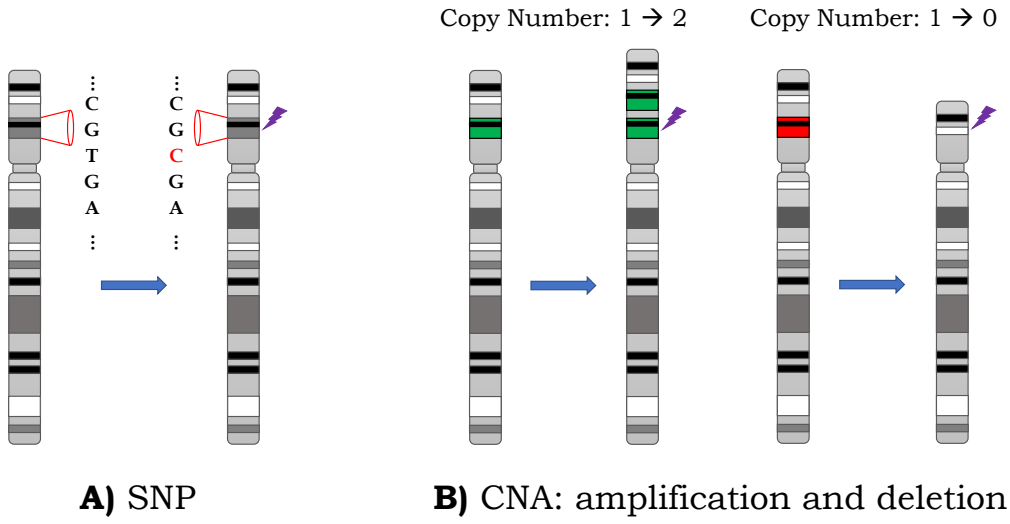

Figure 1.2: **Single-Nucleotide Polymorphisms (SNPs) and Copy-Number Aberrations (CNAs).** (A) SNPs are the result of single-point mutations that affect a single element in the DNA sequence. The DNA can be represented as a string on a four-letter alphabet $\Sigma = \{A, C, G, T\}$, as such a character T is mutated into a C in the example. (B) CNAs are the result of mutations that amplify or delete large genomic region. In the first example the green genomic segment is duplicated, consequently even the black nested segment is duplicated. Their copy number increases by 1. The second example shows the opposite case where the same segments are deleted and the copy number decreases by 1.

cer [26]. Moreover, the copy number of a specific genomic segment can be readily inferred by considering the number of sequencing reads–*sequencing read depth*–mapped to that segment: intuitively, the larger the number of reads mapped to the genomic segment, the higher the corresponding copy number, and vice-versa for lower number of reads (Figure 1.3) [21, 113, 148].

Germinal variants are inherited by individuals because the cells in the germ line are passed on from the parents to their offspring [144]. Thus, once they occur in an individual they are present in all the cells of its children. Unlike germinal variants, somatic cells by definition are not transmitted to the progeny and thus somatic variants are not inherited (Figure 1.4). However, a mutant cell can be the progenitor of a population of mutant cells, all of which have descended from the original cell and thus share the same complement of variants in the progenitor in addition to the somatic mutations they may acquire themselves (Figure 1.4). Cancer results from such an evolutionary process where somatic mutations accumulate in different cells and some of these mutations grant a selective advantage leading to elevated proliferation rates [114]. Therefore, a *clone* defines a subpopulation of cells sharing a unique complement of somatic variants introduced by the same somatic

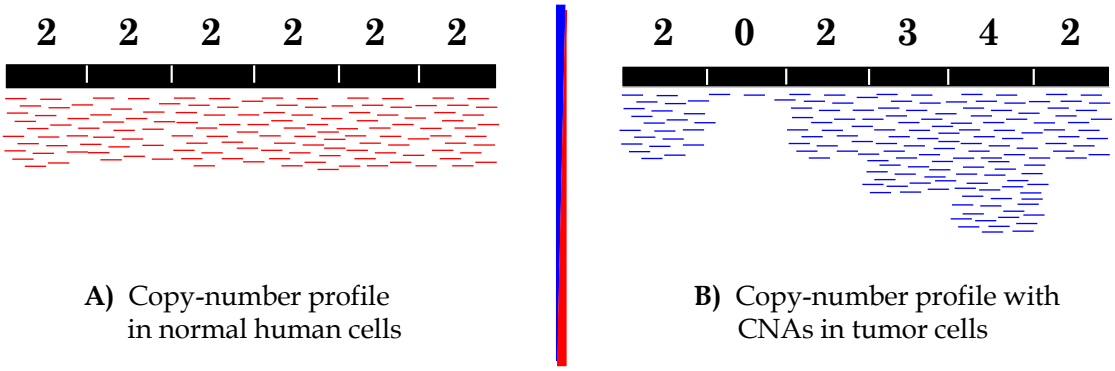

Figure 1.3: **Inferring CNAs from mapped sequencing reads.** A copy-number profile is used to express the integer copy number of each genomic segment. We can infer the copy number of each genomic segment from the number of sequencing reads mapped in the same segment because the number of mapped reads is proportional the its copy number. (A) In normal human cells, we expect there are two copies of each genomic segment, one inherited from the mother and one inherited from the father. (B) CNAs are identified by shifts of copy numbers from the normal state.

mutations. As traditionally done for the evolution of germinal mutations within a population of individuals or across species, the *evolutionary history* of clones can be represented as a *phylogenetic tree* whose nodes correspond to clones and whose edges are labeled by the occured somatic mutations (Figure 1.4) [53]. Identifying the genomic variants that characterize the distinct tumor clones and their evolution can assist in both diagnosis and prognosis of cancer [55, 149]. Unfortunately, the number of these evolutionary trees is extremely huge already with few nodes [51]. Thus, another key challenge of computational biology is the inference of a phylogenetic tree from the observed genomic variants of distinct individuals or tumor clones [27, 36, 127, 129].

Many of the biological questions in computational biology can be formulated as *combinatorial optimization problems*. A problem formulation formally models the aspects relevant to the application that we want to consider and the criterion we adopt to choose an answer to the starting question. In this thesis, we focus on the two previously mentioned challenges of modern applications in computational biology. More specifically, we consider problems concerning the inference of genomic variants and their evolution from sequencing reads of normal and cancer genomes. We adopt an approach based on combinatorial optimization for dealing with these questions: we formulate the biological question as an optimization problem, we study the computational complexity of the proposed problem, we design an algorithm to solve the problem, and we finally validate the algorithm and interpret its solutions on simulated and biological instances. The thesis mainly proposes

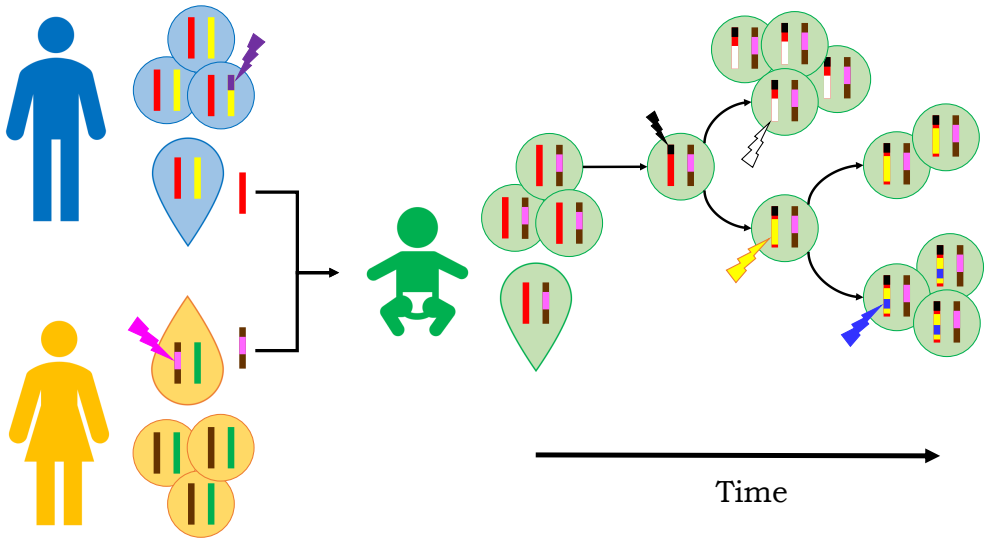

Figure 1.4: **Germinal and somatic mutations and their evolution.** Germinal variants are the result of mutations (pink thunder) occurring in the germ line, the reproductive cells of an individual. Cells of the germ line are represented as droplets, whereas somatic cells are circular. Any human individual inherits half of the genome from the father and the other half from the mother, represented by two lines uniquely identified by a color. Germinal variants may be passed on to the child from the parents, as the pink variant in the example. Somatic variants are instead the result of mutations (black, white, yellow, and blue thunders) in the somatic cells that are not passed on. However, during the lifetime of an individual, somatic mutations can accumulate. A cell acquiring the black somatic mutation is be the progenitor of subpopulations of cells that acquire additional somatic mutations. This evolutionary process is represented as a phylogenetic tree in the example.

novel problem formulations exploiting characteristics specific to each application and designs algorithms that perform well in practice due to a careful study of the combinatorial structure of the related problems.

The contributions of this thesis are subdivided into two parts. The first part deals with problems in the context of *haplotyping* and focuses on the most common genomic variants in normal human cells, which are SNPs. The second part deals with problems in the context of *intra-tumor heterogeneity* and focuses on the somatic CNAs that typically have a central role in cancer. Both the parts of the thesis involve the inference of genomic variants from sequencing reads. In fact, the first part aims to infer the SNPs co-oocuring on the same sequence of DNA, since a human individual inherits different DNA sequences from both the parents. While, the second part aims to infer the CNAs of distinct clones in the same tumor. However, the applications in each context comprise specific characteristics that we exploit in the problem formulations and in the design of related algorithms. For example, the ap-

plications in the first part consider the *long reads* produced by "future-generation" sequencing technologies for improving the resulting accuracy. However, long reads contain a high rate of errors that are uniformly distributed, unlike the traditional sequencing reads. While, the applications in the second part improve the inference of CNAs of distinct tumor clones by jointly inferring their evolution, because the clones from a single tumor share the same evolutionary history. In the following Section 1.1 and Section 1.2, we present the details of the contributions in both the parts of the thesis, respectively. Finally, Section 1.3 describes the structure of the thesis.

1.1 Haplotype Assembly

The genome of an individual is subdivided into *chromosomes*. The number of chromosomes and the number of copies of each chromosome depend on the species. More specifically, the human genome is composed of 23 pairs of homologous chromosomes: 22 pairs of autosomes and a pair of sexual chromosomes. Each copy of a chromosome in a pair is called a *haplotype* and corresponds to a sequence of DNA. One copy, the maternal haplotype, is inherited from the mother and the other copy, the paternal haplotype, is inherited from the father. As such, each haplotype may consist of different genomic variants and, more specifically, of different SNPs. Reconstructing the two haplotypes and identifying the co-occurring SNPs is crucial for characterizing the genome of an individual. The process is known as *phasing* or *haplotyping* and the provided information may be of fundamental importance for many applications, such as analyzing the relationships between genetic variation and gene function, or between genetic variation and disease susceptibility [18, 41]. These applications include agricultural research, medical diagnostics, and drug design [15, 17, 124]. The aim of *haplotype assembly* is to reconstruct the two haplotypes from sequencing reads. Figure 1.5 depicts the scheme of this approach where sequencing reads are mapped to a reference genome, the mapped reads are bipartitioned into two sets, and the reads on each part are assembled to reconstruct each haplotype. The first part of the thesis concerns haplotype assembly and involves both theoretical and methodological contributions.

Minimum Error Correction (MEC) is a prominent combinatorial approach for haplotype assembly that proposes, under a principle of parsimony, to correct the minimum number of errors in the sequencing reads [97]. Typically, the haplotypes and, hence, the reads are represented as binary vectors encoding either the presence or absence of a specific allele for each SNP position. As such, the input collection of reads is modeled as a matrix, called *fragment matrix*, whose rows correspond to reads and whose columns correspond to SNPs. The goal of MEC is hence to find the minimum number of flips of the binary values such that the reads can be un-

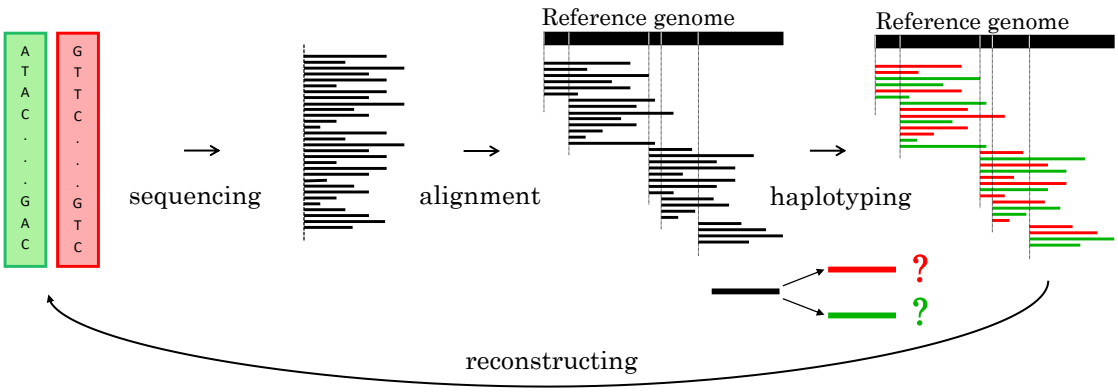

Figure 1.5: **Haplotype Assembly.** The genome of a human individual comprises two copies of each chromosome. For the sake of simplicity, we consider a single chromosome. Each copy is a *haplotype*: the red paternal haplotype and the green maternal haplotype. Sequencing technologies produce a collection of reads from both the haplotypes and the source of each reads is unknown (black short lines). Typically, reads are aligned to a reference genome. Mapped reads covering SNPs that have different alleles on the two haplotypes can be used to distinguish where they come from. Thus, the haplotyping process aims to bipartition the reads into two sets, red and green. However, the presence of errors in the reads confound the presence of SNPs and make the haplotyping a challenging task. At the end, we reconstruct the two haplotypes by assembling all the reads in each part of the bipartition.

ambiguously partitioned into two sets for each chromosome, each one identifying a haplotype. The main challenge is to distinguish between the true SNPs and the errors in single positions of the reads that appear as SNPs.

Unfortunately, MEC is computationally hard to solve, but some approximation-based or fixed-parameter approaches have been shown capable of obtaining accurate results on real data [41, 73, 88, 121]. Despite the significant amount of work present in the literature for the diploid case, some important questions related to the fixed-parameter tractability and approximability of MEC are still open. Two significant open problems are whether there exists a constant-factor approximation algorithm for MEC and whether MEC is in FPT when parameterized by parameters of classical or practical interest, such as the total number of corrections or the length of the reads. In the first part of the thesis, we deal with these questions and other related questions concerning MEC and its traditional variants that impose restrictions to the rows of fragment matrix.

The genome of certain species–especially plants, fish, and yeasts–is *polyploid*, that is, it is composed of more than two haplotypes, and the analysis of such genomes may improve our knowledge of their specific variants, as well as of the mechanisms of eukaryotic evolution [3, 13]. The design of algorithms for dealing with polyploid genomes has received only little attention in the literature

and an equivalent formulation of MEC has not been thoughtfully investigated yet [3, 13, 31]. Here, we extend the problem formulation of MEC to the polyploid case and we analyze its computational complexity, parameterized tractability, and approximability.

Haplotype assembly highly benefits from the advent of "future-generation" sequencing technologies and their capability to produce *long reads* [20, 132]. Long reads may significantly improve the accuracy of haplotype assembly due to their potential of covering many SNPs. However, they consist of novel characteristics different from the ones of traditional reads, such as an elevated error rate with a uniform distribution. Thus, methods for haplotype assembly from long reads need to take these new characteristics into account. Unfortunately, existing methods are not able to do this in a fully satisfactory way, either because they fail to fully exploit the novel characteristics of these long reads or because they are based on restrictive assumptions, such as the "all-heterozygous assumption" which forces to reconstruct complementary haplotypes [40, 87, 121].

By exploiting the uniform distribution of sequencing errors, we introduce a new formulation of MEC, called *k-constrained MEC (k-cMEC)*, that bounds the maximum number of corrections on each column of the fragment matrix. We design a dynamic-programming algorithm, called HapCol, for solving k-cMEC to perform haplotype assembly from long-reads. On a standard benchmark of real data [41], we show that HapCol is competitive with state-of-the-art methods, improving the accuracy and the number of phased positions. Furthermore, experiments on realistically-simulated datasets reveal that our method requires significantly less computing resources, especially memory. Thanks to its computational efficiency, HapCol can overcome the limitations of previous approaches, allowing to deal with larger datasets that may improve the accuracy of the results and without the traditional restrictive assumptions.

1.2 Quantification of Intra-Tumor Heterogeneity

Cancer results from an evolutionary process where somatic mutations accumulate in a population of cells during the lifetime of an individual. The clonal theory of cancer posits that all tumor cells result from this evolutionary process that started at a common founder cell [114]. This founder cell acquired its first somatic mutations that yielded a selective advantage, and its descendants acquired additional somatic mutations. Therefore, a single tumor consists of distinct clones comprising cells that share a unique complement of somatic mutations, and we refer to this phenomenon as *intra-tumor heterogeneity*. The *composition* of a single tumor is hence described by the number of clones, the genomic variants characterizing each clone, and the proportions of cells belonging to distinct clones. The *quantification*

of intra-tumor heterogeneity obtained by reconstructing the tumor composition has been shown to be important in cancer treatment [149]. Unfortunately, intra-tumor heterogeneity complicates the identification of somatic variants because in the traditional *bulk-sequencing* technologies millions of heterogeneous cells are sequenced together. However, this heterogeneity also provides a signal for inferring the tumor composition as well as the evolution of somatic mutations during cancer [63]. Thus, a number of methods have been developed to infer the *evolutionary history* of a tumor from sequencing reads of one or more samples [63, 67, 104, 113, 143].

CNAs are particularly useful for inferring tumor composition and evolution of clones because they are ubiquitous in solid tumors [26]. CNAs can be detected as copy-number deviations of genomic segments from their normal state. In fact, in normal human cells we expect each genomic segment to have a copy number 2: one copy is inherited from the father and the other from the mother. As such, for each clone we model the integer copy numbers of genomic segments as *copy-number profiles* that are usually represented as vectors of integers indicating the copy number of each segment. These profiles are measured using the read depth of sequencing reads from each sample, since the number of sequencing reads mapped in each segment is proportional to its copy number (Figure 1.3) [21, 113, 148]. Unfortunately, most sequencing cancer studies perform bulk sequencing of tumor samples and, consequently, sequencing reads are obtained from heterogeneous mixtures of cells belonging to distinct clones. Since the source of the reads is unknown, we cannot distinguish the reads belonging to different clones and the copy-number profiles are inferred considering all the reads. Consequently, fractional copy numbers are usually obtained instead of integer copy numbers, such that each fraction corresponds to the average of the different copy numbers of the same segment in distinct clones, weighted by their proportions (Figure 1.6). Therefore, to reconstruct CNAs, we need to infer the copy-number profile of each clone from fractional copy numbers.

The contributions of the second part of the thesis aim to improve the inference of CNAs by jointly inferring from multiple samples of the same tumor the CNAs of distinct clones and their evolution. To do this, we consider two problems where the first focuses only on reconstructing the evolution of CNAs, and the second jointly infers CNAs and their evolution. More specifically, we formally introduce the first *Copy-Number Tree (CNT)* problem, which aims to infer the evolutionary history of the clones in terms of CNAs from given integer copy-number profiles. In fact, the problem assumes that the profile of each clone has been retrieved from homogeneous samples containing a single clone. Furthermore, this problem is based on a recent model of CNAs capturing their effects on multiple segments [136] and we design the first exact algorithm for solving it. Next, we extend the previous approach to heterogeneous samples. We introduce the *Copy-Number Tree Mixture Deconvolution (CNTMD)* problem, which aims to jointly infer CNAs of distinct clones and their

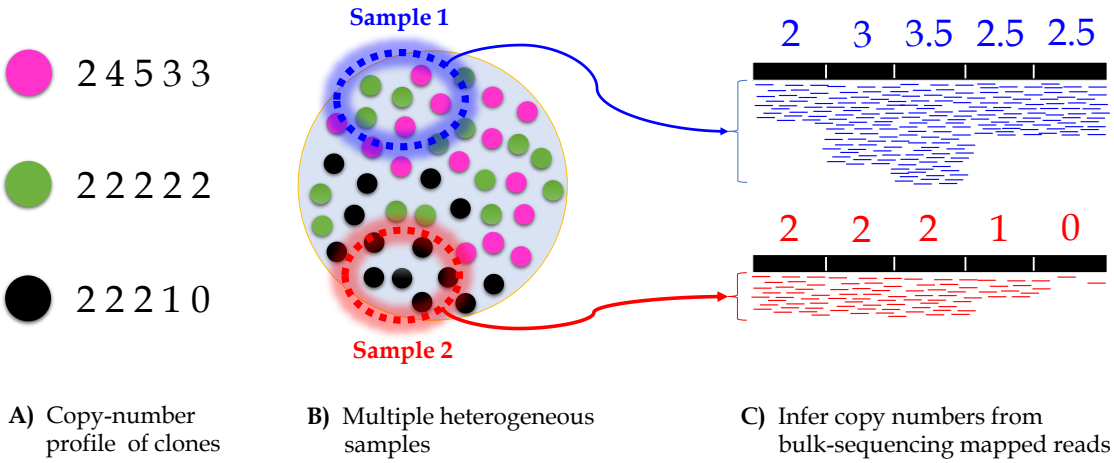

Figure 1.6: **Inferring fractional and integer copy numbers from multiple heterogeneous samples.** (A) The copy-number profiles are reported over five genomic segments for three clones, two tumor clones in black and magenta and the normal clone in green. (B) Two samples are sequenced from bulks of cells. Sample 1 is heterogeneous and comprises half of the cells from the magenta-tumor clone and the other half from the normal clone. Instead, Sample 2 is homogeneous and comprises only cells belonging to the black-tumor clone. (C) The sequencing reads of both the samples are mapped to a reference genome for inferring their copy numbers. Integer copy numbers are inferred from Sample 2 because only one clone is present. Instead, fractional copy numbers are inferred from Sample 1 due to its heterogeneity. Hence, each fraction corresponds to the average of the different copy numbers of the same segment in distinct clones, weighted by their proportions. For example, the fraction 3.5 of third segment results from the sum of 0.5 * 5 for the segment in the magenta-tumor clone and 0.5 * 2 for the same segment in the normal clone, since both the clones are present with a proportion of 50%.

evolution from one or more heterogeneous samples. This problem represents an alternative to traditional approaches that consider only one sample and infer the copy numbers of the clones and their evolution with two independent steps [24, 104, 136]. Figure 1.7 represents the differences of the two problems depending on the properties of the input samples. In the following, we present the details of these two problems.

Modeling CNAs is not straightforward because they can overlap, and thus positions in the genome cannot be treated independently. A number of models have been introduced to study CNA evolution, and these models can be classified into two categories. The first is inspired by traditional phylogenetic models and considers *single events* such that each of those independently affects the copy number of a single segment [24, 104]. However, these models do not account for dependency between adjacent segments in the genome. The second [136] considers the effects

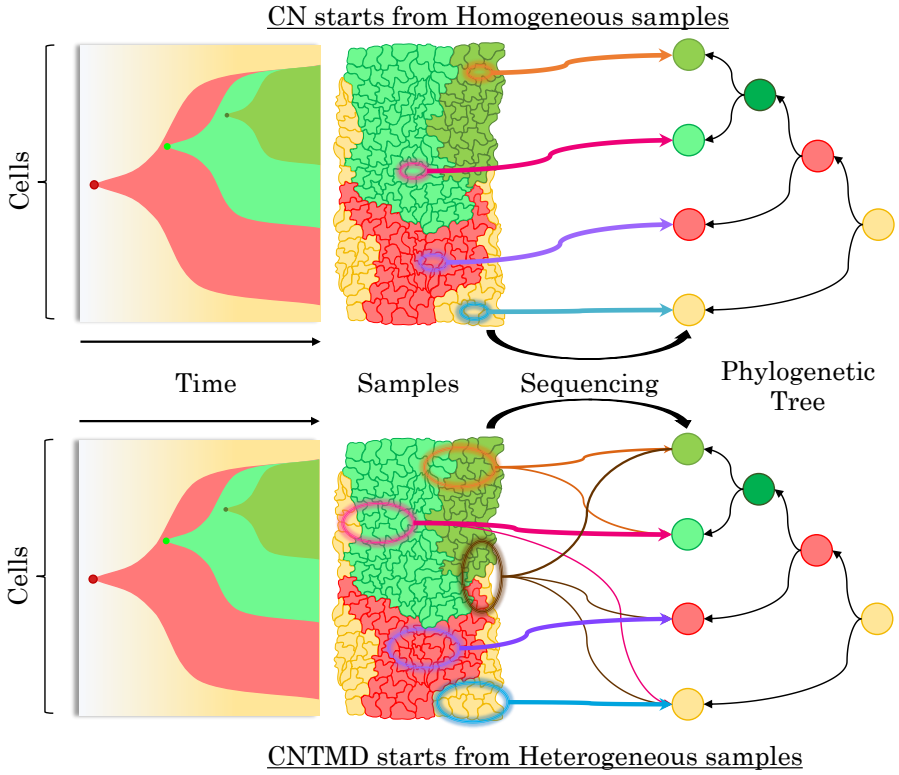

Figure 1.7: **The different approaches of Copy-Number Tree (CNT) and Copy-Number Tree Mixture Deconvolution (CNTMD) problems.** A tumor acquires somatic mutations over time and thus consists of heterogeneous subpopulations of cells, or clones. The tumor clones are colored in red, dark and light green, whereas the normal clone is colored in yellow. In the top part of the figure, CNT assumes to have four homogeneous samples such that each of these contains a single clone. Therefore, each clone is identified from sequencing a single sample and labels each leaf of the phylogenetic tree that CNT seeks to reconstruct. Conversely, in the bottom part of the figure, CNTMD starts from samples that are heterogeneous mixture of the distinct clones. Therefore, CNTMD jointly seeks to infer four tumor clones and their proportions from the sequencing of five starting samples and to reconstruct the phylogenetic tree whose leaves are labeled by these clones.

of CNAs on multiple segments as *interval events* that amplify or delete copies of contiguous segments. The evolutionary distance between two clones is defined by the minimum number of interval events that change the copy-number profile of one into the profile of the other. Using this distance measure, heuristics based on neighbor joining have been applied to reconstruct phylogenetic trees [136]. Here, we formally introduce the CNT problem that, under the same principle of parsimony, seeks a phylogenetic tree explaining with the minimum number of interval events the copy-number evolution of the clones given for the leaves. We show that

the problem is computationally hard, and we design an exact algorithm based on a ILP formulation that turns out to scale to instances of practical size.

CNT assumes to know the integer copy-number profiles of each single clone. Instead, fractional copy numbers are typically inferred from heterogeneous samples. A number of methods have been developed to infer tumor composition from fractional copy numbers by considering each sample independently [21, 54, 71, 113, 116, 148]. However, one can obtain more information by jointly considering more samples from the same tumor [63], as successfully done for single-nucleotide mutations [46, 99] or non-integer copy numbers [134]. The goal is to model the fractional copy numbers of different samples as the product of the integer copy numbers of distinct clones and their proportions (Figure 1.6). However, multiple factorizations can be found, especially without imposing a structure on the inferred CNAs for the resulting profiles. Therefore, the inference of distinct clones may benefit from the joint inference of their evolution. We introduce the Copy-Number Tree Mixture Deconvolution (CNTMD) problem, which given sequencing reads from multiple heterogeneous samples of a tumor aims to find the most parsimonious phylogenetic tree such that each sample corresponds to a mixture of the extant clones. More specifically, CNTMD factorizes the observed-fractional copy numbers into the integer copy numbers of distinct clones and their proportions, and chooses the factorization with the evolutionary history comprising the minimum number of interval events. We design a coordinate-descent algorithm that outperforms existing approaches on simulated data and provides a higher-resolution view of a prostate cancer dataset [67] than published analyses.

1.3 Outline

The thesis is structured as follows. In Chapter 2 we present the details of the main kind of sequencing technologies and the characteristics of the corresponding reads. Next, we review some of the basic notions of phylogenetic analysis that are preliminary to the contributions of Part II. Lastly, in the same chapter we present the basic concepts of combinatorial optimization and the related scheme that we adopt for the contributions in both the parts of this thesis.

In Part I, we present the contributions of the thesis about haplotype assembly. In Chapter 3 we put the contributions into context, we present the known results concerning the computational complexity, parameterized tractability, and approximability of MEC and its traditional variants. In addition, we review the current state-of-the-art methods for dealing with long reads. The basic models and formulations of MEC as well as the novel problem formulations for long reads and polyploid genomes are formally introduced in Chapter 4. In Chapter 5 we present the theoretical contributions concerning the parameterized tractability and approximabi-

lity of MEC and its variants. While, in Chapter 6 we design an exact algorithm, namely HapCol, for dealing with long reads and their novel characteristics. As such, in Chapter 7 we show the results of the comparison between HapCol and current state-of-the-art methods applied both on real and simulated data. We conclude with a discussion about the contributions of this part in Chapter 8.

In Part II, we present the contributions of the thesis about the quantification of intra-tumor heterogeneity. In Chapter 9 we review the background related to alternative models for the evolution of CNAs and methods that infer the CNAs of distinct clones from a single heterogeneous sample. The CNT and CNTMD problem formulations are introduced in Chapter 10 based on the model of interval events and copy-number tree. Furthermore, we show that CNT is NP-hard in Chapter 11, and, consequently, we suspect CNTMD to be NP-hard as well. Thus, in Chapter 12 we firstly design a ILP formulation for solving CNT and next we incorporate an extended version of this ILP formulation into a coordinate-descend algorithm for solving a distance-based variant of CNTMD. Chapter 13 describes the results obtained by running these two algorithms on simulated instances of practical size. In addition, we run the algorithm for CNTMD on a prostate-cancer dataset analyzing the results and comparing to previous published analyses. Refinements of the model and improvements to the algorithm are finally discussed in Chapter 14.

Finally, Chapter 15 presents the final considerations of this thesis. In particular, we firstly describe the main directions for future work concerning the contributions in both the parts of this thesis. Next, we present some closing remarks about the significance of the combinatorial-optimization scheme that we apply in this work and that guides all the contributions.

Chapter 2

Preliminaries

This chapter is devoted to the description of some concepts preliminary to the contributions through the rest of the thesis. First, we present the basic kind of sequencing technologies and the properties of the sequencing reads produced by these. Next, we review the basic notions of phylogenetic analysis that are preparatory to the contributions introduced in Part II. Lastly, we introduce the formal definitions and main concepts of combinatorial optimization. Moreover, we describe the details of the combinatorial-optimization scheme that we adopted to guide all the contributions of this work.

2.1 Sequencing Technologies and Reads

DNA can be represented as a string on a four-letter alphabet $\Sigma = \{A, C, G, T\}$, where each character corresponds to a different nucleotide base pair. The total length of DNA in human individuals is over 3 billions of characters. Sequencing is the process aiming to extract fragments of DNA, that are called *reads*. Hence, reads can be represented as substrings of the DNA string. Reads are orders of magnitude shorter than the length of DNA and they may contain errors. However, the average length of the reads, as well as the error rate and the kind of errors, significantly vary depending on the category of sequencing technologies that are used.

Next-Generation Sequencing (NGS) technologies are able to produce *short reads* that are composed of few hundreds of characters, around 50-300 characters [106, 140]. However, Paired-End technology allows NGS to produce pairs of short reads that are separated by a gap called *insert size* of fixed length, typically slightly lower than twice the read's length [102]. This improves the information provided by NGS reads because the size between the two reads in the same pair is known and they belong to the same DNA sequence. The error rate for reads produced by NGS technologies is usually pretty low, typically lower than 1%. The errors

are usually non-uniformly distributed and most of them correspond to *substitutions* that result in wrong characters of the reads.

"Future-generation" sequencing technologies, such as single molecule real-time technologies like PacBio RS II (http://www.pacificbiosciences.com/products/) and Oxford Nanopore flow cell technologies like MinION (https://www.nanoporetech.com/), produce *long reads*. These correspond to single fragments of DNA, but they are much longer. The length of these reads varies between 10 000 and 50 000 characters. However, the surprising length of these reads comes at a price. In fact the error rate is much higher and is around 10-15% [20, 132]. Even the kind of errors is different than the ones of NGS. The substitutions are usually lower than 5%, whereas the remaining errors are mainly composed of *indels*, comprising insertions of false characters with a rate up to 15% and deletions of characters up to 2%. Lastly, one of the main characteristics of long reads is the uniform distribution of the sequencing errors.

To obtain a sufficient amount of reads that are an informative representation of good quality of the DNA sequence, every sequencing technology needs a certain amount of genetic material from the cells. Two approaches are typically adopted by sequencing technologies. The basic approach of traditional technologies is *bulk sequencing* that consists of sequencing a bulk of cells. A bulk corresponds to a sample from a human individual and comprises millions of different cells [102]. A typical and important assumption is that the reads are obtained uniformly from all the involved DNA sequences. Consequently, we expect that casual somatic variants characterizing only a very small proportion of the cells in the bulk do not have a significant impact on the sequencing reads. Therefore, bulk sequencing is usually applied to analyze the genome of normal human cells [22, 102]. Conversely, in cancer we expect to find a significant amount of cells belonging to distinct tumor clones, because some of the accumulated mutations grant a selective advantage leading to elevated proliferation rates. The sequencing reads of each sample are consequently mixed since they are obtained from many cells of different clones. However, due to the previous assumption for which the reads are uniformly extracted, the proportions of the reads showing a specific genomic variant reflect the proportions of the clones comprising that variant. Therefore, factorization or *deconvolution* approaches, as the one investigated in Part II, can be applied to reconstruct the tumor composition using such data. In fact, many cancer-sequencing studies successfully used bulk sequencing of multiple samples [46, 63, 67, 71, 103].

A second recent approach is *single-cell sequencing* [42, 110, 111]. The key idea of this approach is to isolate a single cell and to perform whole-genome amplifications to obtain enough genetic material for the sequencing. Thus, Single-Cell Sequencing can offer a high resolution of genomic variants since all the reads belong to the DNA sequences of a single cell. This approach can have a significant impact in the context

of cancer because the clones can be retrieved by identifying the genomic variants characterizing each cell without the challenging task of the deconvolution [43, 108, 109, 151]. Unfortunately, Single-Cell Sequencing has still a prohibitive cost for whole genome analysis of thousands of cells and is affected by technological issues resulting in poor data quality [157]. For example, one of the main issues of Single-Cell Sequencing are the errors during the starting amplification process. In fact, some of the regions are not amplified during this step and this may result in the loss of genomic segments that ambiguously appear as the result of CNAs.

2.2 Phylogenetic Analysis

In this section, we present the basics of phylogenetic analysis by comparing the model traditionally used for describing the evolution of SNPs [49, 91, 92, 145] to the model for intra-tumor heterogeneity introduced in Part II for describing the evolution of CNAs. *Phylogenetics* studies the evolutionary history of biological processes or population of individuals among or within species [53, 70]. The *taxa* are the main units of a phylogenetic analysis. A taxon is described in terms of a set of *characters* corresponding to observable attributes or traits. Each character can have two or more *states*, such that each state determines the variation of a character observed in a specific taxon. As such, any taxon is uniquely identified by a complement of states for the defined characters.

Traditionally, when considering the evolution of SNPs, a taxon corresponds to an individual from a population. Each character corresponds to a SNP position, that is a position in the genome where the individuals of the population may contain different alleles. Hence, the characters have only two states indicating either the presence or absence of a SNP allele, and an individual is uniquely characterized by the states of the SNP positions. Conversely, in our model for intra-tumor heterogeneity, the taxa correspond to clones, that are subpopuplations of cells sharing a unique complement of genomic variants as result of occurred somatic mutations. In Part II, we focus on CNAs, therefore there is a character associated to each genomic segment and its state corresponds to the copy number of that segment in the genome of a clone. Unlike the binary characters of the previous case, these characters have more states. Furthermore, to identify the segments that are contiguous, we define a relation of adjacency among them.

Mutational events define the evolutionary relationship between taxa. We model the effects of mutational events on the characters to describe the evolutionary process from a taxon to its *descendants*. The events change the states of the characters, and these effects can be represented as a function of *transition* defining the change of state for every character induced by an event. For the evolution of SNPs, the events correspond to single-point mutations. Therefore, an event acts indepen-

dently on each SNP position by changing its value, that is a flip of the states since they are binary. Conversely, CNAs in the case of intra-tumor heterogeneity act on multiple contiguous segments and their effects depend on the kind of mutational event. In fact, we define two kind of events, called *interval events*: amplifications increase by one copy the state of the affected segments, whereas deletions delete one copy.

We model the *evolutionary history* of certain taxa as a *phylogenetic tree* that is a directed tree whose nodes correspond to taxa [53, 70]. A phylogenetic tree is defined by its *topology*, comprising the set of nodes and the set of edges, and by a *labeling* of its nodes and edges. As such, the nodes are labeled by the states of the corresponding taxa. While, an edge from a parental node to its child defines the evolution from the corresponding taxon to its descent. The edge is consequently labeled by the events that describe the evolution transforming the states of the parent to the states of the child. Generally, the leaves of the tree correspond to the *extant taxa*, whereas the internal nodes correspond to the *ancestors*. We define a node as an ancestor of a group of leaves if there is a path in the tree from this node to each of the leaves, and it corresponds to *most common ancestor* of those leaves if there is no path from this node to another of their ancestor. Therefore, the root of the tree is the most common ancestor of all the extant taxa that we are considering.

The *inference of the evolutionary history* of a set of taxa is the process that aims to infer a phylogenetic tree such that its leaves are labeled by the states of these taxa. Clearly, there exists many such trees. Moreover, the number of phylogenetic trees grows extremely fast increasing the number of leaves and it is huge already with a small number of leaves, since it is a double factorial [51]. As such, we define a criterion for choosing an evolutionary tree for the given taxa. For example, the *principle of parsimony*, which prefers the simplest scenario comprising a minimum number of events, is a criterion often used in biology and followed by the contributions in both the parts of this thesis [50, 56]. Other criteria such as “maximum likelihood” are generally used [52, 66].

Typically, a function is defined to measure the evolutionary distance between two taxa depending on the related events. As done traditionally for the evolution of SNPs as well as in this work for CNAs, the function can simply measures the minimum number of events that transform the states of a taxon to the states of its descent. This function is consequently used to introduce a weight of the edges and the total cost of a tree corresponds to the sum of these weights over all the edges. Therefore, under a principle of parsimony, the challenge is to infer a phylogenetic tree of minimum cost. Furthermore, restrictions to the topology of the considered phylogenetic trees are often imposed to limit the number of possibilites and avoiding trees that are considered equivalent or symmetric. For example, the phylogenetic

trees are often restricted to be full binary trees, such that each internal node has either zero or two children.

When considering the evolution of SNPs under a principle of parsimony, traditional approaches seeks for the phylogenetic tree of minimum cost from a set of individuals whose SNPs have been identified through the analysis of mapped sequencing reads [49, 91, 92, 145]. Conversely, we aim in Part II to find a phylogenetic tree, called *copy-number tree*, comprising the minimum number of interval events from a set of tumor clones whose copy numbers are identified from the number of sequencing reads mapped in genomic segments.

2.3 Combinatorial Optimization

In this section, I describe the approach based on combinatorial optimization that guides all the contributions in both the parts of this thesis. The main focus of this work is on the designing of algorithms for solving *problems* in computational biology. As such, I firstly present the definitions of problems and algorithms. I describe the formal framework applied to study the performance of the algorithms for a problem. Next, I describe the main techniques used to deal with problems considered computationally hard. Lastly, the solving scheme is presented.

2.3.1 Problems, Algorithms, and Computational Complexity

Many of the questions in computational biology can be formulated as *combinatorial optimization problems*. In computer science, an optimization problem is defined by giving a set of *input parameters*, a set of *unknowns*, a set of *constraints*, and an *objective function*. Both the input parameters and the unknowns correspond to free variables over specific domains. However, the values of the parameters are given and an *instance* of the problem is defined by assigning a value to each parameter. Conversely, the unknowns define a set of objects for each instance. The constraints correspond to conditions on the values of the unknowns, and the objective function is a special constraint asking to either minimize or maximize a function on the unknowns. Therefore, a *feasible solution* is an object defined by the unknowns that satisfies the constraints, and a feasible solution is an *optimal solution* if the object correspondingly minimizes or maximizes the value of the objective function over all the defined objects. Given an instance x of a problem, the value of the objective function of any optimal solution is called *optimum* and is denoted by $OPT(x)$.

An *algorithm* f is a general step-by-step procedure that provides a feasible solution $f(x)$ with a value $c(x)$ of the objective function for any instance x of a problem. More formally, an algorithm corresponds to any sequence of operations expressible by a Turing's Machine [29, 60, 86]. We say that an algorithm solves an optimization

problem, that is an *exact algorithm*, if $c(x)$ is equal to $OPT(x)$ for any instance x of the problem. In general, we are interested in finding the "most efficient" algorithm for solving a specific problem. The efficiency requirements of an algorithm are conveniently expressed as functions on variables that describe the *size* of any instance. For example, the variable n is indicating the size of an instance x. The variables of size are intended to represent the amount of data that are necessary to encode every parameter of the instance. We clearly expect that the algorithm's requirements vary with the size of the input. As such, the *running time* $t(n,x)$ is a function that measures the number of steps performed by the algorithm on an instance x of size n, and its *time complexity* is expressed as the maximum running time over all the instances of the problem of size n. Typically, the big-O notation is used to define the time complexity as an upper bound of the running time [7, 29, 90]. For example, the notation $g(n) = O(n^2)$ indicates that $g(n) \leq cn^2$ for a function $g(n)$ and any constant $c > 0$. Space is another important measure for the algorithm's requirements and gauges the memory used by an algorithm to store any data structure used during the computation. *Space complexity* is defined symmetrically to time complexity and we refer the reader to [86] and to [29] for more details.

Computational Complexity Theory defines the *computational complexity* of optimization problems which is a classification of problems depending on the time complexity of the related solving algorithms [60, 81]. The theory is based on *decision problems* where the objective function is replaced by a *question* whose answer is either "yes" or "no" for any feasible solution. As such, each instance x of the problem can be either a *yes*-instance or a *no*-instance depending on the existence of a feasible solution with a yes-answer. However, optimization and decision problems are strongly related since standard techniques transform optimization problems into decision problems without changing their computational complexity [60]. One of the basic computational classes is P which contains the problems solvable by algorithms with a polynomial time complexity, that are called "efficient algorithms". If the problem does not admit such an algorithm, a *brute-force* approach can be adopted: we enumerate all the feasible solutions, whose number is typically exponential, and we verify whether each of these corresponds to an optimal solution, or to a yes-answer in the case of decision problems. As such, another basic computational class is NP which contains the problems that are *verifiable* in polynomial time, that is there exists an algorithm of polynomial time complexity to determine whether a feasible solution either answers yes for decision problems or is optimal for optimization problems. The relation between P and NP is fundamental for the Theory of Computational Complexity, but, unfortunately, is still unknown and corresponds to one of the "big open questions" of our century.

The concept of *reduction* is introduced to study the relation between two decision problems Π_A, Π_B and, consequently, between the corresponding optimization prob-

lems. A reduction $\Pi_A \rightarrow \Pi_B$ is a function π, called "transformation", computable in polynomial time which transforms every instance x of Π_A into a corresponding instance $\pi(x)$ of Π_B such that x is a yes-instance if and only if $\pi(x)$ is a yes-instance. A reduction preserves the yes-answer or the "optimality" of the solutions. Therefore, if $\Pi_A \rightarrow \Pi_B$, an algorithm solving Π_B can be used to solve Π_A through the corresponding transformation. Moreover, we consequently say that Π_B is "at least as difficult as" Π_A and the computational class of Π_B must be contained in the computational class of Π_A. Problems that are at least as difficult as any other problem in NP are called NP-hard. More formally, a problem Π_h is NP-hard if $\Pi \rightarrow \Pi_h$ for each problem Π in NP. Equivalently, a problem Π_h is NP-hard if another problem Π'_h known to be NP-hard is reducible to Π_h [60].

We clearly know P $\subseteq$ NP since a problem solvable in polynomial time is even verifiable in polynomial time, but it is unknown whether P = NP or P $\neq$ NP. However, there is a class of problems in NP that are called NP-*complete* leading computer scientists to believe that P $\neq$ NP. A problem Π is NP-complete if Π is in NP and is NP-hard. Hence, NP-complete problems are the most difficult in NP. For more details on the Theory of Computational Complexity and on additional computational classes, we refer the reader to [60].

2.3.2 Facing Computationally Hardness

Under the assumption that P $\neq$ NP, an efficient algorithm for NP-hard problems does not exists and these problems are considered computationally hard. However, an algorithm "practically tractable" may still exists, where the tractability criterion clearly depends on the context of the application. There are several traditional techniques aiming to design such alternative algorithms for dealing with NP-hard problems and they are typically based on two main ideas: dropping the constraint of optimality on the final solution or designing non-polynomial algorithms that can be still considered *tractable*. We now introduce three of the most common techniques.

The first technique aims to design *approximation algorithms* that are algorithms with a polynomial time complexity and that have a "provable solution quality" [6, 119]. More specifically, an $\alpha-$approximation algorithm f is an algorithm with an *approximation factor* $\alpha \geq 1$ that provides, for every instance x, a feasible solution $f(x)$ with objective value $c(x)$ such that $OPT(x) \leq c(x) \leq \alpha OPT(x)$ for a problem of minimization and $\frac{1}{\alpha}OPT(x) \leq c(x) \leq OPT(x)$ for a problem of maximization. Note that α can be either constant for constant-approximation algorithms or a function $\alpha(n)$ on the size n of x. Therefore, approximation algorithms provide solutions that are not guaranteed to be optimal, but their objective value $c(x)$ is guaranteed to be within certain bounds. *Approximability Theory* studies the existence of approximation algorithms for each problem. Obviously, we desire approximation algorithms whose factor α is as close as possible to 1, that is the value for

which the resulting solutions are also optimal. Therefore, any problem admitting an approximation algorithm with a constant factor is in the class APX. For more details on additional classes, such as APX-hard and PTAS, and for the definitions of approximation-preserving reductions, such as PTAS-reductions and L-reductions, we refer the reader to [6].

The second technique aims to design *fixed-parameter tractable* algorithms that are exact algorithms whose time complexity is polynomial when some of the size variables are fixed as constants [39]. More formally, let n, k be the variables that describe the size of any instance x. We say that an exact algorithm f is fixed-parameter tractable when parameterized by the input parameters of size k if the time complexity of f is of the form $O(t(k) \cdot n^c)$ such that $t(k)$ depends only on k and c is constant. The size variable k is consequently the only in the non-polynomial part of the time complexity. Therefore, fixed-parameter tractable algorithms may be "practically efficient" when parameterized by parameters whose values are relatively small on instances of practical interest for a specific application. *Tractability Theory* studies the existence of fixed-parameter tractable algorithms, and FPT is the class containing the problems that admit such algorithms when parameterized on some input parameters. For additional details, we refer the reader to [39].

The last technique aims to exploit the very effective algorithms, called *solvers*, that have been designed in literature for well-known NP-hard problems. In fact, several solvers have been developed and improved over the years for solving some of the basic NP-hard problems that have many applications in different areas [5, 14, 33, 37, 69, 75, 77, 96, 107]. These methods typically result to be very effective in practice because they use several procedures and heuristics reducing the size of many instances and solve the remaining part as fast as possible. Two of the basic problems for which these methods have been intensely studied are the Satisfiability problem (SAT) and the Integer Linear Programming (ILP) [60]. SAT is the first problem that has been proven to be NP-hard: given a boolean logical formula, it aims to find a true assignments to the variables (of minimum size in its optimization variant) such that the formula evaluates to true [28, 93]. While, ILP is a mathematical optimization problem where the variables are defined over numerical domain, some of them are restricted to be integer, the constraints are expressed as linear inequalities, and the objective function is a linear function. Therefore, a ILP formulation corresponds to a system of linear equations subjected to a function on the variables either to minimize or maximize [118]. Since these basic problems are NP-hard and our problems are often in NP, we can reduce our problem to them and we can use the solvers to obtain a solution. Therefore, the main challenge of this technique is to design a reduction in a compact and minimal way such that the solver can exploit the properties and constraints of the original problem.

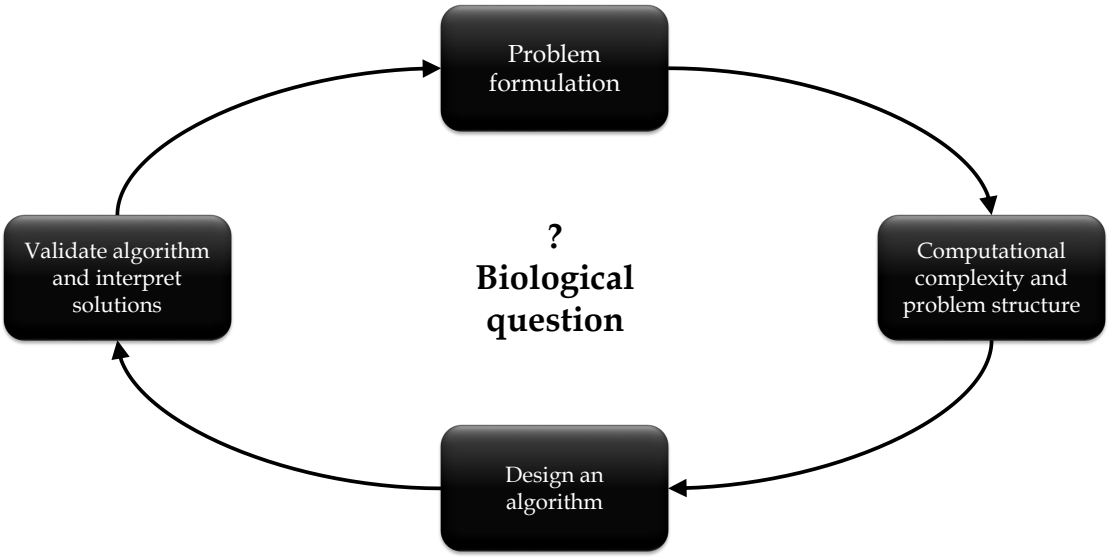

Figure 2.1: **Scheme of combinatorial optimization.** The figure depicts the steps of the scheme based on combinatorial optimization that guides all the contributions on each part of this thesis for answering biological questions. The scheme is composed of four steps. In the first step, a problem formulation is proposed aiming to model the crucial aspects of the application that we want to consider and the criterion for choosing an optimal solution. The second step studies of the computational complexity of the problem for understanding the kind of algorithm that we can design. Moreover, in the same step the structure of the problem is analyzed for understanding the combinatorial properties characterizing the optimal solutions. In the third step, we design a algorithm for the problem. When the problem is computationally hard, specific techniques for dealing with it can be adopted. The last step aims to apply the algorithm on benchmark instances to validate it and on biological data to analyze the solutions for obtaining an answer to the biological question.

2.3.3 A Combinatorial-Optimization Scheme for Computational Biology

Starting from a biological question in a specific application, I adopt a combinatorial-optimization scheme composed of four steps: (1) problem formulation, (2) analyzing computational complexity and combinatorial structure, (3) designing an algorithm, and (4) validating the algorithm and interpreting the solutions. Figure 2.1 depicts the steps of this scheme.

1. The first step is to phrase the biological question as a combinatorial optimization problem by modeling in mathematical terms the elements that we want to consider in the context of the application. Therefore, we encode the set of feasible solutions defining the corresponding constraints on the parame-

ters and unknowns of the model and an objective function representing the criterion we adopt to select an optimal solution. For example, the *principle of parsimony* is one of the most common criteria in biology and prefers solutions defining the simplest scenario characterized by the minimum number of events. The problem formulation varies depending on which elements of the context are considered and how these are encoded. There is a trade-off between including enough aspects of the question to arrive at a meaningful abstraction without getting lost in all the tiny details. Moreover, several variants of the same formulation can be defined by encoding the properties of the objects with different constraints. The challenge is to find the assumptions and the constraints that completely characterize the crucial aspects of the application and that define a problem for which we can design a "tractable" algorithm finding a solution with the best performance.

2. The second step is to study the computational complexity of the resulting problem formulation and to reveal the structure of optimal solutions. The computational complexity of the problem defines the kind of algorithms that we can design for dealing with the problem. While, the *structure* of the problem comprises several properties characterizing the optimal solutions. Useful structures of the problem to exploit are for example the property of sub-optimality for which the optimal solutions can be broken up in smaller sub-problems to solve recursively or necessary and sufficient conditions that help to identify when a feasible solution is optimal. If the problem admits any polynomial-time algorithm, we know that it is in P. Searching for a polynomial-time algorithm is a challenging task, however studying the structure of the problem may reveal properties to exploit for obtaining a polynomial time algorithm. Alternatively, we can attempt to reduce our problem to another problem known to be in P and to use its efficient algorithms. If a polynomial algorithm cannot be find, the problem may be NP-hard. This is a quite common situation, since many of the significant problems in computational biology turn out to be NP-hard. Again, studying the structure of the problem may show similarities to other known NP-hard problems that may be reduced to our problem revealing its computational hardness.

3. Using the knowledge about the computational complexity and structure of the problem from the previous step, we now aim to design an algorithm for our problem formulation. When the problem is in P, we found an efficient algorithm. Therefore, we try to find an efficient alternative algorithm with a lower time complexity or to improve the efficiency of the current algorithm. There are several techniques aiming to improve the running time of an algorithm. For example, recursive algorithms can be improved using the *dynamic*

programming technique for obtaining an iterative algorithm that solves each sub-problem only once and stores their partial solutions. Otherwise, if we reduced the problem to another one in P, we can modify an algorithm for the known problem by exploiting the structure of our problem. When the problem is NP-hard, we need to adopt one of the strategies previously described for dealing with its computational hardness. We may design an approximation algorithm, a fixed-parameter tractable algorithm, or a reduction to a problem with effective solvers. Similar techniques to the previous can be used to improve the running time of approximation and fixed-parameter tractable algorithms. While, when we reduce our problem to a known NP-hard problem, we aim to design the most compact and minimal reduction by exploiting the constraints that encode the structure of the problem. Moreover, we can use one of the solvers that have been deeply engineered and tested over the years, such as CPLEX [75] or GUROBI [69] for solving ILP formulations.

4. The final step is to validate the algorithm by running its implementation on simulated instances and to interpret the solutions obtained by running it on biological instances. Since the ground truth is known, simulated instances are useful for evaluating the performance and quality of the results returned by the algorithm. Therefore, the performance of the algorithm can be evaluated by measures that capture the aspects relevant to the application. Moreover, a comparison between the algorithm and current state-of-the-art methods is crucial for verifying the improvements of the algorithm or the effects of the problem formulation that have been introduced. While, when we run the algorithm on biological instances, we obtain solutions to interpret for answering the original biological question. Information visualization techniques may help in the interpretation of the solutions. For certain biological problems, there are benchmark instances coming together with the true solutions that have been retrieved and biologically verified with additional information. As done for simulated instances, these benchmarks can be used to assess the quality of the obtained solutions and possibly compare the proposed algorithm with state-of-the-art methods. Otherwise, when benchmark instances are not available, we can verify the presence of common patterns with information available in literature or computed with other techniques. In addition, we can re-evaluate the solutions in terms of different measures that capture different aspects of the biological question.

Part I

Haplotype Assembly

Chapter 3

Background

Diploid organisms such as humans contain two sets of chromosomes, one from each parent. Reconstructing the two distinct copies of each chromosome, called *haplotypes*, is crucial for characterizing the genome of an individual. The process is known as *phasing* or *haplotyping* and the provided information may be of fundamental importance for many applications, such as analyzing the relationships between genetic variation and gene function, or between genetic variation and disease susceptibility [18, 41]. These applications include agricultural research, medical diagnostics, and drug design [15, 17, 124]. In diploid species, haplotyping requires assigning the variants to the two parental copies of each chromosome, which exhibit differences in terms of *Single Nucleotide Polymorphisms* (SNPs). Since a large scale direct experimental reconstruction of the haplotypes from the collected samples is not yet cost-effective [88], a computational approach–called *haplotype assembly*–that considers a set of reads, each one sequenced from a chromosome copy, has been proposed. Reads (also called *fragments*) have to be assigned to the unknown haplotypes, using a reference genome in a preliminary mapping phase, if available. This involves dealing in some way with sequencing and mapping errors and leads to a computational task that is generally modeled as an optimization problem [89, 97].

The presence of sequencing and mapping errors makes the haplotype assembly problem a challenging task. Typically, the haplotypes and, hence, the reads can be modeled as binary vectors representing either the presence of the same allele of the reference genome or a variant. Moreover, the fragments obtained from sequencing may not cover some positions of the haplotypes. These uncovered positions are called *holes*, whereas a sequence of holes within a fragment is called *gap*. In the literature, different combinatorial formulations of the problem have been proposed [2, 38, 89, 97]. Among them, *Minimum Error Correction* (MEC) [97] has been proved particularly successful in the reconstruction of accurate haplotypes for diploid species [23, 73, 125]. It aims at correcting the input data with the minimum number of corrections to the SNP values, such that the resulting reads can be un-

ambiguously partitioned into two sets, each one identifying a haplotype. However, MEC is a computationally hard problem. Indeed, MEC is APX-hard even if the fragments have at least one gap [25] and remains NP-hard even if the fragments do not contain gaps (*Gapless MEC*) [25]. wMEC [65] is the weighted variant of the problem, where each possible correction is associated with a weight that represents the confidence degree assigned to that SNP value at the corresponding position. This confidence degree is a combination of the probability that an error occurred during sequencing (phred-based error probability) for that base call, and of the confidence of the read mapping to that genome position. The usage of such weights has been experimentally validated as a powerful way to improve accuracy [158].

To deal with the computational hardness of MEC, several preliminary methods based on the frameworks of parameterized tractability and approximability have been proposed. However, some important questions related to these approaches are still open. In addition, there are two crucial directions that have not been fully investigated yet. The first direction concerns the proposal of problem formulations and the design of algorithms that exploit the characteristics of the data produced by recent technologies, since haplotype assembly benefits from technological developments in genome sequencing. In fact, the advent of next generation sequencing (NGS) technologies provided a cost-effective way of assembling the genome of diploid organisms. However, in order to assemble accurate haplotypes, it is necessary to have reads that are long enough to span several different heterozygous positions [41]. This kind of data are becoming increasingly available with the advent of "future-generation". However, existing methods are not able to deal with such data in a fully satisfactory way, either because accuracy or performances degrade as read length and sequencing coverage increase, or because they are based on restrictive assumptions.

The second direction concerns the study of those species–especially plants, fishes, and yeasts–whose genome is *polyploid*, that is, it is composed of more than two copies for each chromosome. The analysis of such genomes may improve our knowledge of their specific variants, as well as of the mechanisms of eukaryotic evolution [3, 13]. Still, the development of haplotype assembly methods for polyploid genomes has received only little attention in the literature [3, 13, 31]. In fact, the mathematical foundations and a formulation of the MEC problem for polyploid genomes have not been thoughtfully investigated yet.

In Section 3.1 we present the background and the contributions of this thesis concerning the tractability and approximability of MEC and its variants for diploid and polyploid genomes. In addition, Section 3.1.1 includes a detailed descriptions of the state-of-the-art methods introduce to deal with MEC. Similarly, the background and the contributions of this thesis concerning methods to haplotype assembly from long reads are subsequently presented in Section 3.2.

Table 3.1: Current knowledge of computational complexity, approximability, and fixed-parameter tractability for MEC and its variants, Gapless MEC and Binary MEC. Notice that the expression "all-het" states that the corresponding result holds only under the all-heterozygous assumption, while UCG is the Unique Games Conjecture by Khot [83]. The results on parameterized complexity hold for MEC, hence the negative result holds only for MEC while the positive results also hold for its restrictions.

	Computational complexity	*Approximability*	*Parameterized complexity*
MEC	NP-hard [97]	APX-hard [25] $\notin$ APX under UGC $O(\log nm)$ approx. (Sect. 5.1)	$\notin$ XP on c_p and c_f (only MEC, Sect. 5.1) FPT by ℓ (all-het) [73]
Gapless MEC	NP-hard [25]	?	FPT by *cov* [120]
Binary MEC	?	PTAS [80, 117] Simple direct 2-approx (Sect. 5.3)	FPT by ℓ (Sect. 5.4.2) FPT by h (Sect. 5.1)

n number of fragments; m number of SNPs/columns;
ℓ maximum fragment length; *cov* maximum coverage;
h minimum number of corrections; c_p/c_f maximum number of non-hole elements on each column/fragment;

3.1 On the Tractability and Approximability of MEC

The parameterized complexity framework proved to be useful for coping with the computational intractability of MEC on diploid genomes, as it did for several well-known hard combinatorial problems [39]. In particular, MEC is in FPT when parameterized by the *coverage* [120, 121], that is, the maximum number of fragments that cover a SNP position. Moreover, MEC is in FPT also when parameterized by the length of the fragments [73], but this is known only under the *all-heterozygous assumption*, which forces to reconstruct complementary haplotypes. In fact, this

assumption allows the dynamic programming algorithm of He et al. [73] to focus on the reconstruction of a single haplotype and, hence, to limit the possible combinations for each SNP position. Despite the significant amount of work present in the literature for the diploid case, some important questions related to the fixed-parameter tractability and approximability of MEC are still open. Two significant open problems are whether there exists a constant-factor approximation algorithm for MEC and whether MEC is in FPT when parameterized by parameters of classical or practical interest, such as the total number of corrections or the length of the fragments. Indeed, removing the dependency on the all-heterozygous assumption from the algorithm by He et al. [73] does not appear straightforward and, hence, fixed-parameter tractability of MEC when parameterized by the fragment length is still an open problem.

The restriction of MEC where the fragments do not contain holes (*Binary MEC*) is particularly interesting from a mathematical point of view, and is the variant of the well-known *Hamming k-Median Clustering Problem* [25, 85] when $k = 2$. This clustering problem asks for k representative "consensus" (also called "median") strings with the goal of minimizing the Hamming distance between each input string and its closest consensus string. Hamming 2-Median Clustering is well studied from the approximation viewpoint, and at least two Polynomial Time Approximation Schemes (PTAS) have already been proposed [80, 117]. Instead, the computational complexity of Binary MEC is still unknown.

In Chapter 5, we present advances in the characterization of the fixed-parameter tractability and the approximability of MEC problem in the general, gapless, and binary cases that we describe in Chapter 4. We first show that MEC is not in APX, *i.e.*, it is not approximable within constant factor. In addition, we show that MEC is not in XP when parameterized by the number of non-hole elements on SNP positions and fragments. Since these parameters are upper bounds for the maximum number of corrections on each SNP position and on each fragment, it follows that there is no algorithm for MEC exponential in the maximum number of corrections on each SNP position and on each fragment. These parameters are of particular interest, since recent sequencing technologies produce datasets with a low error rate and/or with a uniform distribution of sequencing errors, hence the expected maximum number of corrections to apply on each column/fragment is lower than the coverage/fragment length. However, this result basically rules out the existence of fixed-parameter algorithms on (natural) parameters strictly smaller than coverage and fragment length. Moreover, we show that a reduction previously known [57] can be adapted to prove that MEC is approximable within factor $O(\log nm)$ (where n is the number of fragments and m is the number of SNPs) and that MEC is in FPT when parameterized by the total number of corrections. By inspecting novel combinatorial properties of gapless instances, we also show that Gapless MEC is in

Table 3.2: Current knowledge of computational complexity, approximability, and fixed-parameter tractability for the newly introduced k-ploid MEC and its variants, k-ploid Gapless MEC and k-ploid Binary MEC. The results on parameterized complexity hold for k-ploid MEC, hence the negative result holds only for k-ploid MEC while the positive results also hold for its restrictions.

	Computational complexity	*Approximability*	*Parameterized complexity*
k-ploid MEC	NP-hard when $k = 2$ (Sect. 5.4)	$\notin$ APX when $k = 2$ under UGC (Sect. 5.4)	$\notin$ XP on c_p, c_f, and k (only k-ploid MEC, Sect. 5.4)
k-ploid Gapless MEC	NP-hard when $k = 2$ (Sect. 5.4)	?	FPT by cov and k (Sect. 5.4.1)
k-ploid Binary MEC	NP-hard [25]	PTAS [80, 117]	FPT by ℓ and k (Sect. 5.4.2)

ℓ maximum fragment length; cov maximum coverage;
c_p/c_f maximum number of non-hole elements on each column/fragment;

FPT when parameterized by the length of the fragments and that Binary MEC can be approximated within factor 2. Although Binary MEC is known to admit a PTAS, the 2-approximation algorithm we give is more practical and intuitive than the previous approximation results. Table 3.1 summarizes all the known results prior to this work and highlights the novel contributions we present here, establishing the new state of the art for MEC, Gapless MEC, and Binary MEC problems.

In Chapter 4 we also extend the formulation of the MEC problem to the polyploid case by introducing the *k-ploid Minimum Error Correction (k-ploid MEC)* problem. In addition to the previous theoretical results concerning MEC, we subsequently analyze in Chapter 5 the aspects regarding its computational complexity, parameterized tractability, and approximability. Notice that SNP positions usually assume at most two values also in the polyploid case and, as a consequence, the haplotypes and the fragments can be still represented as binary vectors. In particular, since fixed-parameter tractable algorithms for parameters of practical interest revealed to be successful for dealing with diploid genomes, we show that k-ploid MEC is in FPT when parameterized by the coverage and the number of haplotypes and when parameterized by the fragment length and the number of haplotypes. The latter result clearly applies also to the diploid case, but the algorithm that constructively proves this result has a worse time complexity than the one we specifically pro-

pose for the diploid case (which is instead based on a theoretical result that does not extend to the polyploid case). Table 3.2 reports current knowledge of computational complexity, approximability, and fixed-parameter tractability for the newly introduced k-ploid MEC and its variants, k-ploid Gapless MEC and k-ploid Binary MEC.

3.1.1 Related Works

In this section, we describe the details of the methods that have been traditionally proposed to deal with MEC. Moreover, we describe the different approaches that have been applied to reconstruct the haplotypes in the case of polyploid genomes.

The MEC problem was introduced in [97], where it was shown to be NP-hard on arbitrary instances. Cilibrasi et al. [25] refined the computational complexity analysis by showing that MEC is NP-hard even on instances where fragments do not have gaps (Gapless MEC) and that it is APX-hard on instances where fragments have at most one gap (1-gap MEC). These restrictions are motivated by the characteristics of the prevailing sequencing technologies of that time. Moreover, they also showed that MEC on instances without holes (Binary MEC) is a special form of the Hamming 2-Median Clustering problem and, hence, it admits a Polynomial Time Approximation Scheme (PTAS) in a randomized [117] and deterministic [80] form. Interestingly, the existence of a PTAS for Gapless MEC (or its APX-hardness) and the NP-hardness of Binary MEC are still open questions (albeit Binary MEC with an arbitrary number of haplotypes is NP-hard [25]).

Several heuristic approaches have been proposed to cope with the computational intractability of MEC (see, for example, those surveyed by Geraci [62] and Duitama et al. [41]). Many of these are based on graph-theoretical formulations of the problem. For example, HapCUT [8] proceeds by iteratively computing max-cuts in a graph where each vertex represents a fragment and each edge denotes the presence of conflicts between fragments, while HapCompass [2] models the problem as the *Minimum Weighted Edge Removal (MWER)* problem on a particular graph-based representation of the fragments, called *compass graph*, and heuristically solves it with a strategy based on cycle basis local optimization.

Also exact approaches have been successfully proposed. One of the first exact approaches was proposed by Wang et al. [150] who presented an exact algorithm based on the branch-and-bound method. However, this approach was not always suitable for instances of realistic size, and a genetic algorithm was applied as heuristic for those cases. Since fragments produced by Next-Generation Sequencing technologies usually span only a few SNP positions, He et al. [73] proposed a dynamic programming algorithm whose time complexity is exponential in the fragment length. This algorithm, from a theoretical point of view, establishes that MEC is in FPT when parameterized by the fragment length (but only under the all-heterozygous

assumption). However, it is also important to deal with long fragments, as they usually improve the accuracy of the solution. As a consequence, the authors also presented a reduction of MEC to MaxSAT in order to use well-known and effective MaxSAT solvers for dealing with realistic instances composed of long fragments. For the same reason, Chen et al. [23] proposed an approach based on Integer Linear Programming (ILP) coupled with a procedure for decomposing the input into small independent blocks in order to improve performances. Notably, this approach does not necessarily rely on the all-heterozygous assumption. Despite the decomposition procedure allows to greatly simplify the input matrix for unweighted instances, the ILP formulation requires a large (quadratic) number of variables and, hence, the approach failed in solving certain blocks of the input, called *hard blocks*, resorting to an heuristic for solving them.

Patterson et al. [120, 121] proposed an FPT algorithm for MEC when parameterized by the coverage. Using coverage as parameter is motivated by the fact that it is not expected to grow as fast as fragment length in a realistic dataset with the advent of future-generation sequencing technologies. However, limiting the coverage poses a practical limit on datasets that can be managed, even if a recent parallel version [4] allowed to obtain a constant factor improvement on coverages that can be handled. To better model the characteristics of data produced by future-generation sequencing technologies, Pirola et al. [125] recently proposed a novel variant of the MEC problem, namely the *k-constrained MEC* problem, where the maximum number of corrections for each column is bounded by a given constant k, and showed that this variant is in FPT when parameterized by k and coverage by introducing a dynamic-programming algorithm called HapCol [126]. This result does not conflict with the parameterized intractability we present in Section 5.1 when the parameter is the maximum number of corrections on each column, since HapCol is exponential in both k and coverage. Furthermore, despite an algorithm exponential only in the coverage already existed, HapCol represents a significant practical advancement since the combined use of the two parameters allows to greatly reduce time and space requirements on real data by better modeling the characteristics of such data.

To the best of our knowledge, a theoretical analysis of the computational complexity of the MEC problem on polyploid genomes has never been performed before this work. However, some approaches for dealing with haplotype assembly in polyploid genomes have already been introduced. Aguiar and Istrail [3] extended HapCompass in order to reconstruct multiple copies of the chromosomes, while Das and Vikalo [31] formulated the problem as a semi-definite program and devised a fast approximate algorithm for finding a low-rank solution for them. Recently, Berger et al. [13] proposed a maximum-likelihood estimation framework for polyploid hap-

lotype assembly which is able to manage high ploidity while maintaining acceptable performance.

A computational problem related to polyploid haplotype assembly is the sequence multiassembly problem, which is the problem of reconstructing a set of k sequences from their aligned fragments where k is unknown. However, the two problems differ in some key points. Indeed, in the multiassembly problem, the fragments are often assumed to be error-free (or previously error-corrected) and the aim is to minimize the cardinality k of such a set. This computational problem models, for example, the tasks of estimating viral quasispecies [48] or transcriptome assembly [9, 142, 147] and is often based on the combinatorial problem of Minimum Path Cover of a directed acyclic graph representing the overlaps between the fragments. Interestingly, the problem can be solved in polynomial time if the input fragments are gapless [59] or if there are only contiguous subpath constraints [9, 131], but becomes NP-hard if the fragments have at least one gap [11, 12, 131].

3.2 Methods for Long Reads

Haplotype assembly benefits from technological developments in genome sequencing. In fact, the advent of next generation sequencing (NGS) technologies provided a cost-effective way of assembling the genome of diploid organisms. However, in order to assemble accurate haplotypes, it is necessary to have reads that are long enough to span several different heterozygous positions [41]. This kind of data are becoming increasingly available with the advent of "future-generation" sequencing technologies such as single molecule real-time technologies like PacBio RS II (http://www.pacificbiosciences.com/products/) and Oxford Nanopore flow cell technologies like MinION (https://www.nanoporetech.com/). These technologies, thanks to their ability of producing single end reads longer than 10 000 bases, eliminate the need of paired-end data and have already been used for tasks like genome finishing and haplotype assembly [141]. Besides read length, the future-generation sequencing technologies produce fragments with novel features, such as the uniform distribution of sequencing errors, that are not properly addressed (or exploited) in most of the existing methods that, instead, are tailored to the characteristics of traditional NGS technologies.

Recently, MEC and wMEC approaches have been used in the context of long reads, confirming that long fragments allow to assemble haplotypes more accurately than traditional short reads [2, 41, 120, 121]. Since MEC is NP-hard [25], exact solutions have exponential complexity. Different approaches tackling the computational hardness of the problem have been proposed in literature. Integer linear programming techniques have been recently used [23], but the approach failed to optimally solve some "difficult blocks". There were also proposed Fixed-Parameter

Tractable (FPT) algorithms that take time exponential in the number of variants per read [16, 73, 74] and, hence, are well-suited for short reads, but become unfeasible for long reads. For this kind of data, heuristic approaches have been proposed to respond to the lack of exact solutions [8, 41]. Most of the proposed heuristics, such as RefHap [40], make use of the traditional *all-heterozygous* assumption, that forces the heterozigosity of all the phased positions. These heuristics have good performances but do not offer guarantees on the optimality of the returned solution [41]. Two recent papers [87, 120] aim at processing future-generation long reads by introducing algorithms exponential in the sequencing coverage, a parameter which is not expected to grow as fast as read length with the advent of future-generation technologies. The first algorithm, called ProbHap [87], is a probabilistic dynamic programming algorithm that optimizes a likelihood function generalizing the objective function of MEC. Albeit ProbHap is significantly slower than the previous heuristics, it obtained a noticeable improvement in accuracy. The second approach, called WhatsHap [120], is the first exact algorithm for wMEC that is able to process long reads. It was shown to be able to obtain a good accuracy on simulated data of long reads at coverages up to 20x and to outperforms all the previous exact approaches. However, it cannot handle coverages higher than 20x and its performance evidently decreases when approaching that limit.

In addition to the contributions presented in the previous Section 3.1, we exploit a characteristic of future-generation technologies, namely the uniform distribution of sequencing errors, for introducing in Chapter 4 a new variant of wMEC, called k-constrained MEC (k-cMEC). In Chapter 6, we design an exact fixed-parameter tractable algorithm for solving k-cMEC whose parameters are (i) the maximum number k of corrections that are allowed on each SNP position and (ii) the coverage. The new algorithm, called HapCol, is based on a characterization of feasible solutions from the problem structure studied in Chapter 4 and its time complexity is $O(cov^{k+1}Lm)$ (albeit it is possible to prove a stricter bound), where cov is the maximum coverage, L is the read length, and m is the number of SNP positions. HapCol is able to work without the all-heterozygous assumption.

In Chapter 7, we experimentally compare accuracy and performance of HapCol on real and realistically simulated datasets with three state-of-the-art approaches for haplotype assembly – RefHap, ProbHap, and WhatsHap. On a real standard benchmark of long reads [41], we executed each tool under the all-heterozygous assumption, since this dataset has low coverage (~3x on average) and since the covered positions are heterozygous with high confidence. HapCol turns out to be competitive with the considered methods, improving the accuracy and the number of phased positions. We also assessed accuracy and performance of HapCol on a large collection of realistically-simulated datasets reflecting the characteristics of "future-generation" sequencing technologies that are currently (or soon) available (coverage

up to 25x, read length from 10 000 to 50 000 bases, substitution error rate up to 5%, and indel rate equal to 10%) [20, 76, 132]. When considering higher coverages, interesting applications such as SNP calling or heterozygous SNPs validation become feasible and reliable [112]. Since these applications require that haplotypes are reconstructed without the all-heterozygous assumption, on the simulated datasets we only considered the tools that do not rely on this assumption – WhatsHap and HapCol. Results on the simulated datasets with coverage 15–20x show that HapCol, while being as accurate as WhatsHap (they achieve an average error of ~2%), is faster and significantly more memory efficient (~2 times faster and ~28 times less memory). The efficiency of HapCol allows to further improve accuracy. Indeed, the experimental results show that HapCol is able to process datasets with coverage 25x on standard workstations/small servers (whereas WhatsHap exhausted all the available memory, 256GB) and that, since the number of ambiguous/uncalled positions decreases, the haplotypes reconstructed by HapCol at coverage 25x are ~9% more accurate than those reconstructed at coverage 20x.

Chapter 4

Problem Formulation

Some of the results in this chapter are published in:

P. Bonizzoni, R. Dondi, G. W. Klau, Y. Pirola, N. Pisanti, and S. Zaccaria. On the minimum error correction problem for haplotype assembly in diploid and polyploid genomes. *Journal of Computational Biology*, 23(9):718–736, 2016.

In this chapter, we present the new problem formulations considered in Part I and their problem structure. Firstly, in Section 4.1 we present the basic model of reads and haplotypes, as well as the problem formulations of MEC and its traditional variants. Next, we study the structure of these problems and the combinatorial properties of their optimal solutions in Section 4.2. We conclude by describing the new problem formulations that are introduced in this part of the thesis. More specifically, in Section 4.3 we introduce the k-cMEC problem by exploiting the uniform distribution of sequencing errors that characterize long reads. While, in Section 4.4 we extend the MEC formulation to polyploid genomes and we describe some basic results about its tractability that are inherited from the known results of MEC reported in Chapter 3.

4.1 Basic Model

In this section, we introduce some basic notions and the formal definition of the MEC problem. In the rest of the work, we indicate, as usual, the value of a vector s at position t as $s[t]$.

Let s be a vector. As usual, we denote the value of s at position t by $s[t]$. A *haplotype* is a vector h of length m belonging to $\{0,1\}^m$. Let h_1, h_2 be the two haplotypes of an individual. A position j is called *heterozygous* if $h_1[j] \neq h_2[j]$, otherwise (*i.e.*, if $h_1[j] = h_2[j]$) j is called *homozygous*. A *fragment* is a vector f of

length m belonging to $\{0,1,-\}^m$. In a fragment f, a position $f[j] = -$ is called a *hole*. A *gap* in a fragment f is a maximal sub-vector of f of holes, preceded and followed by a non-hole element (that is, there exist two positions j_1 and j_2 with $j_1+1 < j_2$ such that $f_i[j_1], f_i[j_2] \neq -$ and $f_i[t] = -$ for all t with $j_1 < t < j_2$). Moreover, the length ℓ_i of a fragment f_i is defined as the number of elements contained in f between the leftmost and rightmost non-hole elements (included).

A *fragment matrix* is a matrix $\mathscr{M}$ composed of n rows and m columns such that each entry contains a value in $\{0,1,-\}$. Each row of $\mathscr{M}$ corresponds to fragment, or read, and hence is a vector belonging to $\{0,1,-\}^m$. Symmetrically, each column of $\mathscr{M}$ corresponds to a SNP position and is a vector belonging to $\{0,1,-\}^n$. We denote by f_i the i-th row of $\mathscr{M}$ and by p_j the j-th column of $\mathscr{M}$. As a consequence, the entry of $\mathscr{M}$ at the i-th row and j-th column is denoted by $f_i[j]$ or $p_j[i]$. We define as ℓ or L the maximum length over all the fragments in $\mathscr{M}$. A fragment matrix is *gapless* if no fragment contains a gap.

Given two row vectors s_1 and s_2 belonging to $\{0,1,-\}^m$, s_1 and s_2 are in *conflict* when there exists a position j, with $1 \leq j \leq m$, such that $s_1[j] \neq s_2[j]$ and $s_1[j], s_2[j] \neq -$, otherwise s_1 and s_2 are in *agreement*. A collection $\mathscr{F}$ of fragments is in *agreement* if any pair of fragments f_1, f_2 in $\mathscr{F}$ is in agreement. A fragment matrix $\mathscr{M}$ is *conflict free* if and only if there exist two haplotypes h_1, h_2 such that each row of $\mathscr{M}$ is in agreement with one of h_1 and h_2. In an equivalent way, a fragment matrix $\mathscr{M}$ is conflict free if and only if there exists a *bipartition* (P_1, P_2) of the fragments in $\mathscr{M}$ such that P_1 and P_2 are in agreement.

When a fragment matrix $\mathscr{M}$ is conflict free, all the fragments in each part of the bipartition can be merged in order to reconstruct a haplotype, intended as a fragment without holes. Unfortunately, a fragment matrix $\mathscr{M}$ is not always conflict free. The Minimum Error Correction problem deals precisely with this issue by asking for a minimum set of *corrections* that make a fragment matrix conflict free, where a correction of a given fragment f_i at position j, with $f_i[j] \neq -$, is the flip of the value $f_i[j]$, replacing a 0 with a 1, or a 1 with a 0.

Problem 4.1 *Minimum Error Correction (MEC) problem*
***Input**: a fragment matrix $\mathscr{M}$ of n rows and m columns.*
***Output**: a conflict free matrix $\mathscr{M}'$ obtained from $\mathscr{M}$ with the minimum number of corrections.*

Gapless MEC is the restriction of MEC where the input fragment matrix $\mathscr{M}$ is gapless, while *Binary MEC* is the restriction of (Gapless) MEC where the matrix $\mathscr{M}$ does not contain holes (that is, when $\mathscr{M}$ is a binary matrix). In the weighted variant wMEC of MEC, there is a weight $w(p_j[i])$ associated with each non-hole entry $p_j[i]$ of the input matrix $\mathscr{M}$ that represents the cost of correcting that entry. In this case, the goal is to minimize the total weight instead of the number of corrections.

A column of a matrix is called *homozygous* if it contains values in $\{0,-\}$ or in $\{1,-\}$, otherwise it is called *heterozygous*. We say that a column p_j *covers* a row f_i if $p_j[i] \in \{0,1\}$ or there exist l,r with $l < j < r$ such that $p_l[i], p_r[i] \in \{0,1\}$ (*i.e.*, $p_j[i]$ is a hole belonging to a gap) and we define the *active fragments* of p_j as the set $\mathscr{A}_{\mathscr{F}}(p_j)$ of all the fragments covered by p_j. We denote by $\mathscr{A}_{\mathscr{F}}(p_{j_1}, p_{j_2})$ the intersection $\mathscr{A}_{\mathscr{F}}(p_{j_1}) \cap \mathscr{A}_{\mathscr{F}}(p_{j_2})$ for two columns p_{j_1} and p_{j_2}. Therefore, the *coverage* cov_j of a column p_j is equal to $|\mathscr{A}_{\mathscr{F}}(p_j)|$ and we define as cov the maximum coverage over all the columns of $\mathscr{M}$. Furthermore, we indicate as $\widehat{\mathscr{A}_{\mathscr{F}}}(p_j)$ the *non-hole active fragments* in $\mathscr{A}_{\mathscr{F}}(p_j)$. As such, the *non-hole coverage* $\widehat{cov}_j$ of a column p_j is correspondingly equal to $|\widehat{\mathscr{A}_{\mathscr{F}}}(p_j)|$ and $\widehat{cov}$ is the maximum non-hole coverage over all the columns of $\mathscr{M}$. However, when considering the variants Gapless MEC and Binary MEC, we know that $\widehat{cov} = cov$ and, consequently, $\widehat{\mathscr{A}_{\mathscr{F}}}(p_j) = \mathscr{A}_{\mathscr{F}}(p_j)$. For the sake of simplicity, in these cases we just refer to them as cov and $\mathscr{A}_{\mathscr{F}}(p_j)$.

Given a conflict free fragment matrix $\mathscr{M}$, any heterozygous column p_j encodes a bipartition of the fragments covered by p_j indicating which one belongs to one haplotype and which one belongs to other. Instead, any homozygous column p_j gives no information on how the covered fragments have to be partitioned, and it is "in accordance" with any other bipartition or heterozygous column.

Definition 4.1 *Two columns p_{j_1}, p_{j_2} of a fragment matrix $\mathscr{M}$ are in* accordance *if (1) at least one of p_{j_1}, p_{j_2} is homozygous or (2) p_{j_1}, p_{j_2} are both heterozygous and on $\mathscr{A}_{\mathscr{F}}(p_{j_1}, p_{j_2})$ they are identical or complementary.*

The *correction distance* between two columns p_{j_1}, p_{j_2} evaluates the minimum number of corrections needed to transform p_{j_1} and p_{j_2} into heterozygous columns in accordance and it is defined as $d(p_{j_1}, p_{j_2}) = \min\{|E|, |\overline{E}|\}$, where

$$E = \{i : p_{j_1}[i] \neq p_{j_2}[i] \wedge p_{j_1}[i] \neq - \wedge p_{j_2}[i] \neq -\}$$

and

$$\overline{E} = \{i : p_{j_1}[i] = p_{j_2}[i] \wedge p_{j_1}[i] \neq - \wedge p_{j_2}[i] \neq -\}.$$

Given a column p_j of a fragment matrix $\mathscr{M}$, we define the *homozygous distance* $H(p_j)$ as the number of times the minor allele (i.e., the least frequent value of the column) appears in p_j if it is not greater than an integer k, or infinity otherwise. More formally, $H(p_j)$ is equal to $d(p_j, \underline{0})$ if $d(p_j, \underline{0}) \leq k$ (notice that $d(p_j, \underline{1}) = d(p_j, \underline{0})$, where $\underline{0}$ and $\underline{1}$ are the columns composed only of zeros and ones, resp.), or to ∞ otherwise. Homozygous columns cannot induce a conflict due to the fact that the corresponding positions in the two reconstructed haplotypes can be homozygous with no influence on the other positions. For this reason, we can remove every homozygous column from any input fragment matrix $\mathscr{M}$ without changing the optimal

solution and we can assume that $\mathscr{M}$ is only composed of heterozygous columns [1]. However, notice that a heterozygous column p_j in the input can be transformed into a homozygous column p'_j in the output. As a consequence, the optimal solution $\mathscr{M}'$ can potentially contain homozygous columns. Furthermore, given a conflict free matrix $\mathscr{M}'$, notice that the two resulting haplotypes h_1, h_2 can be easily computed from the bipartition of the fragments induced by the columns of $\mathscr{M}'$.

Any instance of MEC can be transformed to an instance of wMEC whose input matrices are gapless. In fact, each gap in any fragment of the input matrix $\mathscr{M}$ can be modeled as zero-weight entries equal to 0 or 1. For this reason, even though we propose in Chapter 5 and Chapter 6 approaches that considers gapless fragment matrices the approach can be easily extended to deal with any general fragment matrix $\mathscr{M}$.

4.2 Problem Structure

As stated in the following lemma, pairwise column accordance on gapless matrices is a necessary and sufficient condition for being conflict free. The property of these matrices is fundamental for several algorithm that we design in Chapter 5 and Chapter 6.

Lemma 4.1 *Let $\mathscr{M}$ be a* gapless *fragment matrix. Then, $\mathscr{M}$ is conflict free if and only if each pair of columns is in accordance.*

Proof By definition, if $\mathscr{M}$ is conflict free, each pair of columns is in accordance. For this reason, we just prove by induction on the number m of columns in $\mathscr{M}$ that if each pair of columns is in accordance, then $\mathscr{M}$ is conflict free.

If $m = 1$, the lemma obviously holds.

Assume by induction that the lemma holds for the first $m-1$ columns in $\mathscr{M}$, we need to prove that the lemma still holds for all the m columns. The submatrix on the first $m-1$ columns is conflict free by induction and, for this reason, a bipartition (P_1, P_2) of the corresponding fragments exists. We assume that p_m is heterozygous, since the lemma clearly holds when p_m is homozygous. Moreover, we define p_h the rightmost heterozygous column on the first $m-1$ columns and we ignore the homozygous columns between p_h and p_m because they cannot induce conflicts and are in accordance with any other heterozygous column. By assumption, p_h and p_m are in accordance. Hence, p_h and p_m define the same bipartition on the fragments

[1] This assumption should not be confused to what in the literature is called *all heterozygous assumption* which refers to a requirement for the output. The latter asks for haplotypes which are complementary in all positions, while the assumption we make here is on the input and is motivated by the fact that it is without loss of generality for this problem statement.

in $\mathscr{A}_{\mathscr{F}}(p_h, p_m)$. Since $\mathscr{M}$ is gapless, there is no column p_y in $\{p_1, \ldots, p_{h-1}\}$ such that $\mathscr{A}_{\mathscr{F}}(p_y, p_m) \setminus \mathscr{A}_{\mathscr{F}}(p_h) \neq \emptyset$, hence $\mathscr{A}_{\mathscr{F}}(p_m) \setminus \mathscr{A}_{\mathscr{F}}(p_h) \not\subseteq \mathscr{A}_{\mathscr{F}}(p_y)$ for $1 \leq y \leq h-1$. It follows that there exists a bipartition $(P_1 \cup P'_1, P_2 \cup P'_2)$ for every fragment active on all the m columns, where (P'_1, P'_2) is the bipartition induced by p_m on the fragments in $\mathscr{A}_{\mathscr{F}}(p_m) \setminus \mathscr{A}_{\mathscr{F}}(p_{m-1})$. As a consequence the submatrix on the first m columns is conflict free. □

Since such a property is independent by the order of the columns, this result also applies to fragment matrices that can be transformed to gapless matrices by rearranging their columns (*gapless-reducible* fragment matrices). Testing if a fragment matrix $\mathscr{M}$ is gapless-reducible can be performed in polynomial time by testing if the binary matrix $\mathscr{B}(\mathscr{M})$ obtained from $\mathscr{M}$ by substituting each non-hole element with a one and each hole with a zero has the *consecutive ones property* (C1P) [105]. Therefore, we immediately obtain the following result.

Corollary 4.2 *Let $\mathscr{M}$ be gapless-reducible fragment matrix. Then, $\mathscr{M}$ is conflict free if and only if each pair of columns is in accordance.*

Gapless-reducibility, beside being of theoretical interest, can also be relevant in practice, as it is potentially able to transform an "almost gapless" fragment matrix (*i.e.*, a fragment matrix with only a few short gaps due, for example, to indel sequencing errors) to a gapless matrix and, hence, to apply algorithms designed for gapless instances which, in general, could be more efficient than those designed for general instances.

Notice that the accordance relation among heterozygous columns is transitive since it basically requires that pairs of columns are equal or complementary. Therefore, since homozygous columns cannot induce conflicts, we have the following result.

Corollary 4.3 *Let $\mathscr{M}$ be a* gapless *fragment matrix. Then, $\mathscr{M}$ is conflict free if and only if each pair of consecutive columns in the matrix obtained from $\mathscr{M}$ by removing its homozygous columns is in accordance.*

The property defined in Lemma 4.1 is particularly important when designing exact algorithms for Gapless MEC, as it allows to test only for pairwise column accordance in order to ensure that the matrix is conflict free. For example, the fixed-parameter algorithm for Gapless MEC that we present in Chapter 5 is based on this property. Furthermore, notice that if we relax the requirement that $\mathscr{M}$ is gapless (or gapless-reducible), then the property does not hold. Consider, for example, the fragment matrix $\mathscr{M}$ composed of three fragments $f_1 = 01-$, $f_2 = -01$, and $f_3 = 1-0$. The three columns are pairwise in accordance, but the matrix is not conflict free (and, in fact, f_3 contains a gap).

Given two columns p_{j_1}, p_{j_2} of a fragment matrix $\mathscr{M}$, we define their (generalized) Hamming distance $d_H(p_{j_1}, p_{j_2})$ as $|\{i \mid \{p_{j_1}[i], p_{j_2}[i]\} = \{0,1\}\}|$. As said in the previous section, their correction distance $d(p_{j_1}, p_{j_2})$ equivalently corresponds to the minimum between $d_H(p_{j_1}, p_{j_2})$ and $d_H(\overline{p_{j_1}}, p_{j_2})$ (where $\overline{p}$ is the complement of p on non-hole entries). Notice that the correction distance is non-negative and symmetric, but does not satisfy the triangle inequality, hence, despite the name, is not a metric. Moreover, recall that the homozygous distance $H(p_j)$ is the minimum between the number of 0's and 1's contained in p_j. Intuitively, the correction distance is the cost of making a column equal or complementary to another column, while the homozygous distance is the cost of making a column homozygous. As such, a solution of MEC over a fragment matrix $\mathscr{M}$ is a bipartition of its fragments, that can be encoded as a binary vector O. It is easy to see that the cost of that solution is:

$$cost_{\mathscr{M}}(O) = \sum_{j=1}^{m} \min(d(O, p_j), H(p_j)). \tag{4.1}$$

4.3 The k-constrained MEC

In this section, we introduce a variant of the MEC problem, called *k-constrained MEC (k-cMEC)* and motivated by the uniform distribution of sequencing errors of future-generation technologies, where the number of errors (hence, corrections) per column is bounded by an integer k. Given an input fragment matrix $\mathscr{M}$, a conflict free fragment matrix $\mathscr{M}'$ obtained from $\mathscr{M}$ with h corrections is defined as a *k-corrected matrix* for $\mathscr{M}$ if for each column p_j in $\mathscr{M}$ and for the corresponding column p'_j in $\mathscr{M}'$ we have $d(p_j, p'_j) \leq k$. According to this definition we introduce the following variant of MEC:

Problem 4.2 *k-constrained Minimum Error Correction (k-cMEC)*
Input*: a fragment matrix $\mathscr{M}$ and an integer k.*
Output*: a k-corrected matrix $\mathscr{M}'$ for $\mathscr{M}$ obtained with the minimum number of corrections.*

Given a k-corrected matrix $\mathscr{M}'$ for a fragment matrix $\mathscr{M}$, we can see each heterozygous column p'_j in $\mathscr{M}'$ as the correction of the corresponding column p_j in $\mathscr{M}$. Hence, considering a column p_j, we define a *k-correction* B_j for p_j as a vector in $\{0,1,-\}^n$ with $\mathscr{A}_{\mathscr{F}}(B_j) = \mathscr{A}_{\mathscr{F}}(p_j)$ such that $d(p_j, B_j) \leq k$ and B_j is heterozygous. According to this definition a k-correction B_j describes a feasible way to transform p_j into the heterozygous column p'_j when $d(p'_j, B_j) = 0$. Therefore, we define the space of these corrections as β_j, such that β_j is the set containing all the possible k-corrections B_j for the column p_j. Notice that $\underline{0}$ and $\underline{1}$ can be imagined as the corrections for any homozygous column in $\mathscr{M}'$.

The weighted variant of this problem can be easily defined in the same way as wMEC for MEC. The goal of the weighted version is to compute a k-corrected matrix $\mathscr{M}'$ obtained from $\mathscr{M}$ with minimum total weight.

Consider a fragment matrix $\mathscr{M}$. There always exists a feasible solution for the MEC problem on input $\mathscr{M}$, while a feasible solution for the k-cMEC problem, for a fixed k, on input $\mathscr{M}$ may not exist. This implies that a feasible solution for the MEC problem on input $\mathscr{M}$ may not be a feasible solution for the k-cMEC problem. Hence, an optimal solution for the k-cMEC problem is not necessarily an optimal solution for the MEC problem.

4.4 Polyploid Genomes

In this section, we introduce a formulation of the MEC problem applied to polyploid genomes. In particular, we assume that the number k of chromosome copies is known *a priori*, and, for this reason, we consider the k-ploid variant of the problem, that is, the *k-ploid Minimum Error Correction (k-ploid MEC)* problem.

The main concepts and definitions for this problem are the same of those introduced in the previous Section 4.1 for the (diploid) MEC problem. The most important difference lies in the definition of the output. In fact, the goal of the novel formulation is to reconstruct k haplotypes, where k is the (given) number of chromosome copies that compose the polyploid genome of the species under study. As a consequence, we need to generalize the concept of conflict free fragment matrix. In particular, let $\mathscr{M}$ be a fragment matrix, we say that $\mathscr{M}$ is *k-conflict free* if and only if there exists a k-partition $\mathscr{F} = (\mathscr{F}_1, \ldots, \mathscr{F}_k)$ of its fragments such that each part $\mathscr{F}_i$ is in agreement. At last, we formally define the k-ploid MEC problem:

Problem 4.3 *k-ploid Minimum Error Correction (k-ploid MEC) problem*
***Input**: an integer k and a fragment matrix $\mathscr{M}$ of n rows and m columns.*
***Output**: a k-conflict free matrix $\mathscr{M}'$ obtained from $\mathscr{M}$ with the minimum number of corrections.*

As for the MEC problem, Gapless k-ploid MEC is the restriction of k-ploid MEC where the input fragment matrix $\mathscr{M}$ is gapless, while Binary k-ploid MEC is the restriction of (Gapless) k-ploid MEC where the matrix $\mathscr{M}$ does not contain holes (that is, when $\mathscr{M}$ is a binary matrix). Furthermore, notice that all the fragments in each part of the k-partition of a k-conflict free matrix can be merged in order to reconstruct a haplotype, in the same way as it can be done for the bipartition of a conflict free matrix in the diploid case.

Clearly, k-ploid MEC is a generalization of (diploid) MEC. As a consequence, k-ploid MEC inherits some hardness results from MEC. Since we know that MEC

is APX-hard [25], that MEC is not in APX under the Unique Games Conjecture (Chapter 5), that gapless MEC is NP-hard [25], and that MEC is not in XP when parameterized by the number of non-hole elements on rows and columns (Chapter 5), the following theorem clearly holds.

Theorem 4.4 *k-ploid MEC is* APX*-hard, k-ploid MEC is not in* APX *under the Unique Games Conjecture, gapless k-ploid MEC is* NP*-hard, and k-ploid MEC is not in* XP *when parameterized by the number k of haplotypes and the number of non-hole elements on rows and columns.*

In addition, Cilibrasi et al. [25] showed that if the number of haplotypes to be reconstructed is specified as part of the input and if the input mfragment matrix does not have holes (such as in the case of k-ploid Binary MEC), the k-ploid Binary MEC problem becomes NP-hard. However, the authors were not able to say whether there exists a constant-factor approximation algorithm for the problem.

In Chapter 5, we propose two algorithms that allow to prove that k-ploid MEC is in FPT when parameterized by coverage and number of haplotypes and when parameterized by fragment length and number of haplotypes. The choice of parameters is reasonable, since, depending on the characteristics of sequencing technologies, coverage or fragment length are usually limited by small constants. Moreover, species that naturally have more than 8 haplotypes are quite rare among higher-order organisms.

Chapter 5

On the Tractability and Approximability

Some of the results in this chapter are published in:

P. Bonizzoni, R. Dondi, G. W. Klau, Y. Pirola, N. Pisanti, and S. Zaccaria. On the fixed parameter tractability and approximability of the minimum error correction problem. In *26th Annual Symposium on Combinatorial Pattern Matching (CPM)*, volume 9133 of *LNCS*, pages 100–113, 2015.

P. Bonizzoni, R. Dondi, G. W. Klau, Y. Pirola, N. Pisanti, and S. Zaccaria. On the minimum error correction problem for haplotype assembly in diploid and polyploid genomes. *Journal of Computational Biology*, 23(9):718–736, 2016.

This chapter presents the main theoretical contributions of Part I concerning MEC, its traditional variants, and the new formulation k-ploid MEC extending MEC to polyploid genomes. Firstly, in Section 5.1 we show that the Edge Bipartization problem is reducible to MEC. Therefore, we introduce several results concerning the approximability and the parameterized tractability of MEC. Next, in Section 5.2 we show that a variant of practical interest of MEC, Gapless MEC, is in FPT when parameterized by the fragment length. In the subsequent Section 5.3 we design a simple and direct 2−approximation algorithm for Binary MEC. Lastly, in Section 5.4 we design two dynamic-programming algorithms to show that k-ploid MEC is in FPT when parameterized by the coverage and the number of haplotypes and when parameterized by the fragment length and the number of haplotypes.

5.1 Approximation and Parameterized Complexity of MEC

In this section, we show that MEC is not in APX, that is MEC cannot be approximated within constant factor. We achieve this result by introducing an L-reduction [6] from the Edge Bipartization problem to MEC.

The Edge Bipartization problem is defined as follows.

Problem 5.1 *Edge Bipartization (EB) problem [60]*
***Input**: an undirected graph $G = (V, E)$.*
***Output**: $E' \subseteq E$ of minimum size such that $G' = (V, E \setminus E')$ is bipartite.*

Now, we present the details of the reduction. Given an undirected graph $G = (V, E)$, we build the associated fragment matrix $\mathscr{M}(G)$ (with $|V|$ rows and $|E|$ columns) by setting, at each column p_j associated with edge $e_j = \{u, v\} \in E$, $f_u[j] = 0$, $f_v[j] = 1$, and $f_z[j] = -$ for $z \neq u, v$. Notice that, by construction, there exists a conflict in $\mathscr{M}(G)$ between fragments f_u and f_v if and only if $\{u, v\} \in E$.

Lemma 5.1 *Let $G = (V, E)$ be an undirected graph and $\mathscr{M}(G)$ be the associated fragment matrix. Given a solution E' of EB over G, we can compute in polynomial time a solution of MEC over $\mathscr{M}(G)$ with $|E'|$ corrections. Symmetrically, given a solution of MEC over $\mathscr{M}(G)$ with h corrections, we can compute in polynomial time a solution E' of EB over G of size at most h.*

Proof ($\Rightarrow$) Let E' be a set of edges such that $(V_1 \uplus V_2, E \setminus E')$ is bipartite, where V_1 and V_2 are the parts of the bipartition. Build a matrix $\mathscr{M}'(G)$ from $\mathscr{M}(G)$ by flipping, for each $e_j = \{u, v\} \in E'$, the entry $f_u[j]$. Clearly, $\mathscr{M}'(G)$ is obtained from $\mathscr{M}(G)$ with $|E'|$ corrections and it does not contain conflicts induced by edges in E'. Let $(\mathscr{F}_1, \mathscr{F}_2)$ be the bipartition of fragments of $\mathscr{M}'(G)$ such that $\mathscr{F}_i := \{f_u \mid v_u \in V_i\}$ (for $i \in \{1, 2\}$). Each $\mathscr{F}_i$ is in agreement because it does not contain a pair of fragments associated with the endpoints of an edge of $E \setminus E'$. Hence, $\mathscr{M}'(G)$ is conflict free.

($\Leftarrow$) Let $\mathscr{M}'(G)$ be a conflict free matrix obtained from $\mathscr{M}(G)$ with h corrections and let C' be the subset of columns of $\mathscr{M}'(G)$ that contain exactly a correction. In fact, notice that a single correction is sufficient to transform a column into a homozygous column in accordance with any other column. Consider the set $E' := \{e_j \in E \mid p_j \in C'\}$. Clearly, $|E'| \leq h$. Since $\mathscr{M}'(G)$ is conflict free, there exists a bipartition $(\mathscr{F}_1, \mathscr{F}_2)$ of the fragments such that both $\mathscr{F}_1, \mathscr{F}_2$ are in agreement. Build sets V_1, V_2 such that $V_i := \{v_u \mid f_u \in \mathscr{F}_i\}$ (with $i \in \{1, 2\}$). We claim that $(V_1 \uplus V_2, E \setminus E')$ is bipartite. Suppose to the contrary that there exists an edge $e_j = \{u, v\} \in E \setminus E'$ such that $u, v \in V_i$, $i \in \{1, 2\}$. Since $f_u[j] = f_v[j]$ in $\mathscr{M}'(G)$, this implies that exactly one of

$f_u[j]$ and $f_v[j]$ has been corrected (since $f_u[j] \neq f_v[j]$ in $\mathcal{M}(G)$). As a consequence, we have that $e_j \in E'$, contradicting the assumption. □

Khot [83] proved that, under the Unique Games Conjecture, EB is not in APX. Since Lemma 5.1 proves that MEC is L-reducible to EB, we have the following result.

Theorem 5.2 *Under the Unique Games Conjecture [83], MEC is not in* APX.

Edge Bipartization is NP-hard even if the graph is cubic, that is if each vertex has degree three [154]. If the graph is cubic, then the fragment matrix built by our reduction contains two non-hole elements on each column and three non-hole elements on each fragment. As a consequence, any optimal solution clearly places at most two corrections on each column (actually, at most a correction is placed on each column of any optimal solution, since it is enough for transforming the column into a homozygous column) and at most three corrections on each fragment. Hence, from this observation and from the NP-hardness of EB on cubic graphs, we obtain the following result.

Theorem 5.3 *MEC is not in* XP[1] *when parameterized by the number of non-hole elements on columns (SNP positions) and on rows (fragments).*

As a consequence of Theorem 5.3, there is no algorithm for MEC of time complexity $O(n^{f(c_p,c_f)})$ where c_p, c_f are the maximum number of non-hole entries in each column and row, respectively. Furthermore, since the number of non-hole elements on each column and each row are upper bounds for the maximum number of corrections on each column and each fragment, it follows that MEC is not in XP (hence, also not in FPT) when parameterized by these parameters.

The inapproximability result given in Theorem 5.2 nicely complements an approximation (and fixed-parameter tractable) result that can be inferred by a reduction presented in [57], where MEC is reduced to the Maximum Bipartite Induced Subgraph problem (MBIS). Given a vertex-weighted graph G, MBIS asks for a maximum weight subset of vertices of G that induces a bipartite graph. The reduction defines a graph, called *fragment graph*, whose set of nodes is the union of two sets: a set of nodes, called *fragment nodes*, one for each fragment, and a set of nodes, called *entry nodes*, one for each entry of the matrix. In order to avoid the removal of fragments nodes, they are assigned a sufficiently large weight.

The reduction can be easily reworked in order to prove approximation and fixed-parameter tractability results for MEC. More precisely, MEC is now reduced to

[1]We recall that XP is the class of parameterized problems that admit an algorithm of time complexity $n^{f(k)}$ for some computable function f and parameter k.

	p_1	p_2	p_3	p_4
f_1	1	0	-	1
f_2	-	1	0	0
f_3	0	-	1	1

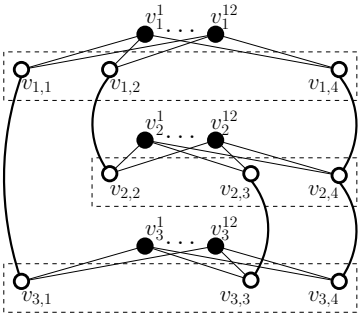

Figure 5.1: A 3×4 fragment matrix (left) and the associated *fragment graph* (right). Fragment-nodes are in black, while entry-nodes are in white.

the *Graph Bipartization* (GB) problem, a problem related to MBIS. Given an unweighted graph G, GB asks for the minimum number of vertex removals so that the resulting graph is bipartite. The reduction given in [57] can be modified by defining a new version of the fragment graph (see Fig. 5.1), where each (weighted) fragment node is substituted with a sufficiently large set of fragment nodes. From the construction of the fragment graph, it follows that a fragment matrix $\mathscr{M}$ is conflict free if and only if the corresponding fragment graph is bipartite and that a solution of MEC with k corrections corresponds to a solution of GB that removes k vertices.

Since GB can be approximated within factor $O(\log|V|)$ [61] and is in FPT when parameterized by the number of removed vertices [68, 130], we have that:

Theorem 5.4 *(1) MEC can be approximated in polynomial time within factor* $O(\log nm)$ *where* n *is the number of rows and* m *is the number of columns.*
(2) MEC is in FPT *when parameterized by the total number of corrections.*

5.2 Gapless MEC is in FPT when Parameterized by Fragment Length

In this section, we introduce a fixed-parameter tractable algorithm for Gapless MEC when parameterized by the maximum length ℓ of the fragments. The algorithm is based on a dynamic programming approach and aims at finding a specific tripartition for the columns of a gapless fragment matrix $\mathscr{M}$. In this section, we assume w.l.o.g. that $\mathscr{M}$ is a gapless fragment matrix and the fragments of $\mathscr{M}$ are sorted by starting position.

Lemma 4.1 provides a sufficient and necessary condition for the reconstruction of a solution for Gapless MEC, that is a conflict free fragment matrix. For this reason, the gapless condition is required by this algorithm. In fact, if the fragment matrix contains gaps, the accordance of the columns is not sufficient to ensure that

there are no conflicts. Therefore, we firstly show a result that directly derives from Lemma 4.1. In particular, the following proposition stresses the relationship between a bipartition of the fragments and a tripartition of the columns in a gapless fragment matrix $\mathscr{M}$ that is conflict free.

Lemma 5.5 *Given a gapless fragment matrix $\mathscr{M}$, the following assertions are equivalent:*

1. *$\mathscr{M}$ is conflict free.*
2. *There exists a bipartition $(\mathscr{F}_1, \mathscr{F}_2)$ of the fragments, where both $\mathscr{F}_1$ and $\mathscr{F}_2$ are in agreement.*
3. *There exists a tripartition $T = (L, H, R)$ of the columns such that each column in H is homozygous, each column in $L \cup R$ is heterozygous, $d_H(p_{j_1}, p_{j_2}) = 0$ for all the columns $p_{j_1}, p_{j_2} \in L$ ($p_{j_1}, p_{j_2} \in R$, resp.) and $d_H(\overline{p_{j_1}}, p_{j_2}) = 0$ for each column $p_{j_1} \in L$ and each column $p_{j_2} \in R$.*

Proof The equivalence between (1) and (2) holds by definition. Therefore, we only show that (1) and (3) are equivalent.

($\Rightarrow$) If $\mathscr{M}$ is conflict free, each pair of columns p_{j_1}, p_{j_2} is in accordance by Lemma 4.1. By definition, either at least one column is homozygous or $d(p_{j_1}, p_{j_2}) = 0$. It directly follows that a tripartition $T = (L, H, R)$ can be built such that each column in H is homozygous, each column in $L \cup R$ is heterozygous, $d_H(p_{j_1}, p_{j_2}) = 0$ for all the columns $p_{j_1}, p_{j_2} \in L$ ($p_{j_1}, p_{j_2} \in R$, resp.) and $d_H(\overline{p_{j_1}}, p_{j_2}) = 0$ for each column $p_{j_1} \in L$ and each column $p_{j_2} \in R$.

($\Leftarrow$) Let p_{j_1}, p_{j_2} be two columns. If at least one column belongs to H, then p_{j_1} and p_{j_2} are in accordance by definition. Otherwise, when p_{j_1} and p_{j_2} are both heterozygous, then $d(p_{j_1}, p_{j_2}) = 0$. Indeed, if they belong to the same part (L or R), then $d_H(p_{j_1}, p_{j_2}) = 0$, whereas if they belong to different parts then $d_H(\overline{p_{j_1}}, p_{j_2}) = 0$. Hence, p_{j_1} are p_{j_2} in accordance. □

Based on Lemma 5.5, we introduce an algorithm for Gapless MEC that builds a tripartition of the columns of $\mathscr{M}$ in order to find a conflict free matrix $\mathscr{M}'$ obtained from $\mathscr{M}$ with the minimum number of corrections. Notice that in the rest of this section we implicitly refer only to tripartitions built as reported in the third assertion of Lemma 5.5.

The algorithm iteratively proceeds row-wise and, at each step, computes a tripartition for the columns considered so far. In particular, the key observation that allows to bound the exponential complexity of the algorithm to the parameter ℓ is that we can build any tripartition for all the columns in $\mathscr{M}$ by adding only a subset of columns, called *active columns*, for each row. This subset contains the columns covering the current fragment and the columns covering both previous and successive

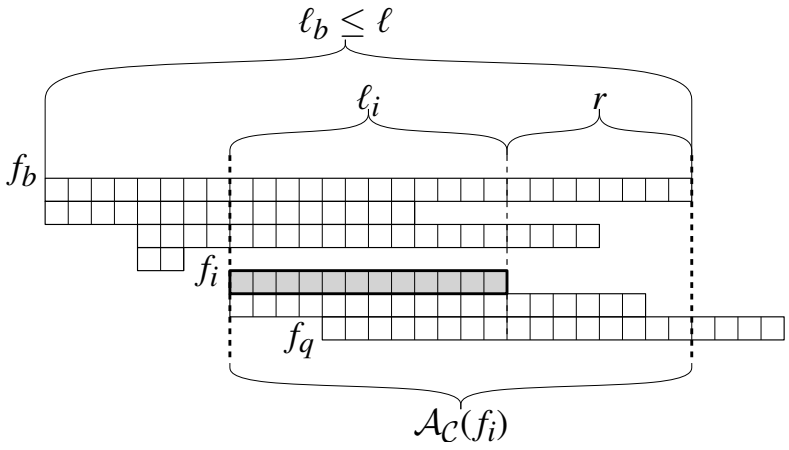

Figure 5.2: The set $\mathscr{A}_{\mathscr{C}}(f_i)$ of active columns for a fragment f_i. There are r columns p_j to the right of f_i such that there exist two fragments f_b, f_q with $b < i < q$ and $p_j[b], p_j[q] \neq -$. Since fragments are sorted by starting position, we have that $|\mathscr{A}_{\mathscr{C}}(f_i)| = \ell_i + r \leq \ell_b \leq \ell$.

fragments. Indeed, we need to remember the tripartition established by previous fragments for columns that are covered by successive fragments. More formally, we define the set *active columns* for a fragment f_i as:

$$\mathscr{A}_{\mathscr{C}}(f_i) = \{p_j \mid (p_j[i] \neq -) \vee (\exists x, y \text{ with } x < i < y \mid p_j[x], p_j[y] \neq -)\}$$

Fig. 5.2 represents the active columns $\mathscr{A}_{\mathscr{C}}(f_i)$ of a fragment f_i. The cardinality of $\mathscr{A}_{\mathscr{C}}(f_i)$ is bounded by ℓ. In fact, considering a row f_i, notice that $\ell_i \leq \ell$ and no column p_k, to the left of f_i, is in $\mathscr{A}_{\mathscr{C}}(f_i)$. Recall that fragments are sorted by starting position and assume that r is the number of columns p_j to the right of f_i, such that there are f_b, f_q with $b < i < q$ and $p_j[b], p_j[q] \neq -$. Since the r columns must be contained in $\mathscr{A}_{\mathscr{C}}(f_b)$ for a fragment f_b with a starting position preceding the one of f_i, it holds that $\ell_i + r \leq \ell_b \leq \ell$. It clearly follows that $|\mathscr{A}_{\mathscr{C}}(f_i)| = \ell_i + r \leq \ell$.

Considering two rows f_{i_1} and f_{i_2}, with $i_1 < i_2$, a tripartition for all the columns in $\mathscr{A}_{\mathscr{C}}(f_{i_1}) \cup \mathscr{A}_{\mathscr{C}}(f_{i_2})$ can be computed by combining a tripartition T_1 for $\mathscr{A}_{\mathscr{C}}(f_{i_1})$ and a tripartition T_2 for $\mathscr{A}_{\mathscr{C}}(f_{i_2})$, only if T_1 and T_2 are "in accordance", that is, they are partitioning the shared columns in the same way. For this reason, we say that a tripartition $T_2 = (L_2, H_2, R_2)$ for $\mathscr{A}_{\mathscr{C}}(f_{i_2})$ *extends* another tripartition $T_1 = (L_1, H_1, R_1)$ for $\mathscr{A}_{\mathscr{C}}(f_{i_1})$ if and only if $L_1 \cap \mathscr{A}_{\mathscr{C}}(f_{i_2}) \subseteq L_2$, $H_1 \cap \mathscr{A}_{\mathscr{C}}(f_{i_2}) \subseteq H_2$, and $R_1 \cap \mathscr{A}_{\mathscr{C}}(f_{i_2}) \subseteq R_2$.

At each step i, the algorithm computes a tripartition T for $\mathscr{A}_{\mathscr{C}}(f_i)$ extending a tripartition T' for $\mathscr{A}_{\mathscr{C}}(f_{i-1})$. Since $\mathscr{A}_{\mathscr{C}}(f_{i-1})$ also contains all the columns p_j with $p_j[i-1] = -$ such that there exists $y < i-1$ with $p_j[y] \neq -$ and $p_j[i] \neq -$, it follows

that T even extends any tripartition computed at the previous steps extended by T'. As a consequence, we prove the following implication.

Lemma 5.6 *If there exists a conflict free matrix $\mathscr{M}''$ obtained from $\mathscr{M}$ on the first $i-1$ rows that induces a tripartition T' for the columns in $\mathscr{A}_{\mathscr{C}}(f_{i-1})$, and if T is a tripartition for the columns in $\mathscr{A}_{\mathscr{C}}(f_i)$ extending T', then there exists a conflict free matrix $\mathscr{M}'$ obtained from $\mathscr{M}$ on the first i rows that induces the tripartition T for the columns in $\mathscr{A}_{\mathscr{C}}(f_i)$.*

Proof By definition, $p_j[i] \neq -$ and $p_j[y] = -$ for each column $p_j \in \mathscr{A}_{\mathscr{C}}(f_i) \setminus \mathscr{A}_{\mathscr{C}}(f_{i-1})$ and for each $y < i$. By assumption T extends T', hence build $\mathscr{M}'$ such that the columns covered by the first $i-1$ rows are tripartitioned as in $\mathscr{M}''$ and the remaining columns only covered by f_i are tripartitioned according to T. By construction, $\mathscr{M}'$ induces the tripartition T for $\mathscr{A}_{\mathscr{C}}(f_i)$. Since $\mathscr{M}''$ is conflict free, it follows that $\mathscr{M}'$ is conflict free by Lemma 5.5. □

At each step i and for each tripartition $T = (L,H,R)$ for $\mathscr{A}_{\mathscr{C}}(f_i)$, the algorithm chooses the tripartition T' extended by T for $\mathscr{A}_{\mathscr{C}}(f_{i-1})$ that induces the minimum cost (*recursive step*) and computes the minimum number of corrections to add on the current fragment f_i in order to tripartition all the columns in $\mathscr{A}_{\mathscr{C}}(f_i)$ according to T (*local contribution*). In particular, the algorithm considers the minimum number of corrections on f_i such that $p_j[i] = 1$ or $p_j[i] = 0$ for all p_j in L and, on the contrary, $p_j[i] = 0$ or $p_j[i] = 1$ for all p_j in R. At the same time, the minimum number of corrections on the fragment f_i is computed for each column p_j in H such that p_j on the first i rows can be optimally transformed into a homozygous column. Therefore, we define $D[i,T]$ as the minimum number of corrections to obtain a conflict free matrix $\mathscr{M}'$ from $\mathscr{M}$ on the first i rows that induces a tripartition T for $\mathscr{A}_{\mathscr{C}}(f_i)$. The algorithm proceeds row-wise computing the value $D[i,T]$ for each fragment f_i and for each tripartition T for $\mathscr{A}_{\mathscr{C}}(f_i)$ by the following recursive equation:

$$D[i,T] = \Delta(i,T) + \min_{T' \text{ extended by } T} D[i-1,T'] \tag{5.1}$$

where T' is a tripartition for $\mathscr{A}_{\mathscr{C}}(f_{i-1})$. In the recursion, we consider only the tripartitions T' extended by T, since the shared columns have to be partitioned in the same way. In conclusion, the local contribution is defined as:

$$\Delta(i,T) = O(i,H) + \min \begin{cases} E^0(i,L) + E^1(i,R) \\ E^1(i,L) + E^0(i,R) \end{cases} \quad \textit{where } T = (L,H,R) \tag{5.2}$$

such that $E^x(i,F)$ is the cost of correcting the columns in F for fragment f_i to value x, that is $E^x(i,F) = |\{j \mid j \in F \wedge p_j[i] \notin \{x,-\}\}|$, and $O(i,H)$ is the minimum number of

corrections to apply on fragment f_i such that the columns in H, considered on the first i rows, can be turned into homozygous columns with minimum cost. Denote by $\#^x_{i,j}$ the number of values equal to x in $\{p_j[1],\dots,p_j[i]\}$. The minimum between $\#^0_{i,j}$ and $\#^1_{i,j}$ states the minimum number of corrections necessary to turn a column p_j on the first i rows into a homozygous column. Since $O(i,H)$ refers only to the corrections on fragment f_i, we can compute $O(i,H)$ as:

$$O(i,H) = \sum_{j \in H} \begin{cases} 1 & p_j[i] = 0 \text{ and } \#^0_{i,j} \leq \#^1_{i,j} \\ 1 & p_j[i] = 1 \text{ and } \#^1_{i,j} \leq \#^0_{i,j} \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

Given a set of columns F, it is easy to see that $\sum_{i \in \{1,\dots,n\}} O(i,F) = \sum_{p_j \in F} H(p_j)$.

The base case of the recurrence is $D[1,T] = \Delta(1,T)$ for each tripartition T for $\mathscr{A}_{\mathscr{C}}(f_1)$. The algorithm returns the optimum corresponding to $\min_T D[n,T]$ where T is a tripartition for $\mathscr{A}_{\mathscr{C}}(f_n)$. Furthermore, an optimal tripartition for all the columns can be computed by backtracking.

The algorithm computes all the values $D[i,T]$ for each tripartition T of the columns in $\mathscr{A}_{\mathscr{C}}(f_i)$ and for each i in $\{1,\dots,n\}$. It follows that there are $O(3^\ell \cdot n)$ entries and, therefore, the space complexity is equal to $O(3^\ell \cdot n)$. Given a tripartition T, we need $O(3^\ell)$ time to enumerate all the tripartitions T' extended by T because we have to tripartition all the columns in $|\mathscr{A}_{\mathscr{C}}(f_{i-1}) \setminus \mathscr{A}_{\mathscr{C}}(f_i)|$ with $\mathscr{A}_{\mathscr{C}}(f_{i-1}) \leq \ell$ and, consequently, $|\mathscr{A}_{\mathscr{C}}(f_{i-1}) \setminus \mathscr{A}_{\mathscr{C}}(f_i)| \leq \ell$. Since $\Delta(i,T)$ can be computed in $O(\ell)$ time, each entry $D[i,T]$ can be computed in $O(3^\ell \cdot \ell)$. It follows that the total running time of the algorithm is $O(3^{2\ell} \cdot \ell \cdot n)$. Notice that storing partial information during the computation (using an approach similar to the one presented in [120]) we can decrease the complexity to $O(3^\ell \cdot \ell \cdot n)$.

We now show the correctness of the algorithm.

Lemma 5.7 *Consider a gapless fragment matrix $\mathscr{M}$.*

1. *If $D[i,T] = h$, then there exists a conflict free matrix $\mathscr{M}'$ obtained from $\mathscr{M}$ on the first i rows with h corrections that induces a tripartition T for the columns in $\mathscr{A}_{\mathscr{C}}(f_i)$.*

2. *If $\mathscr{M}'$ is a conflict free matrix obtained from $\mathscr{M}$ on the first i rows with h corrections that induces a tripartition T for the columns in $\mathscr{A}_{\mathscr{C}}(f_i)$, $D[i,T] \leq h$.*

Proof We prove the lemma by induction on i. Both the statements obviously hold for $i = 1$. Assume that lemma holds for $i-1$, we show that both the statements hold for i.

(1) By Eq. (5.1), there exists a tripartition T' for $\mathscr{A}_{\mathscr{C}}(f_{i-1})$ such that T extends T' and $D[i,T] = h = \Delta(i,T) + D[i-1,T']$. Assuming $D[i-1,T'] = h'$, by induction

there exists a conflict free matrix $\mathscr{M}''$ obtained from $\mathscr{M}$ on the first $i-1$ rows with h' corrections that induces a tripartition T' for $\mathscr{A}_{\mathscr{C}}(f_{i-1})$. By Proposition 5.6, there exists a conflict free matrix $\mathscr{M}'$ obtained from $\mathscr{M}$ on the first i rows that induces a tripartition T for $\mathscr{A}_{\mathscr{C}}(f_i)$. Since T extends T', by construction we can add $\Delta(i,T)$ corrections on fragment f_i in order to build $\mathscr{M}'$ starting from $\mathscr{M}''$. It follows that $\mathscr{M}'$ is obtained from $\mathscr{M}$ with $\Delta(i,T)+h'=h$ corrections.

(2) Assume that $\mathscr{M}''$ is the submatrix of $\mathscr{M}'$ obtained from $\mathscr{M}$ on the first $i-1$ rows with h' corrections that induces a tripartition T' for $\mathscr{A}_{\mathscr{C}}(f_{i-1})$. Clearly, T' is extended by T due to the fact that $\mathscr{M}''$ is equal to $\mathscr{M}'$ on the first $i-1$ rows. Since $\mathscr{M}'$ contains $\Delta(i,T)$ corrections on the row f_i by construction, it follows that $h=\Delta(i,T)+h'$. Moreover, we know that $D[i-1,T'] \leq h'$ by induction and by Eq. (5.1) that $D[i,T]=\Delta(i,T)+\min_{T'' \text{ extended by } T} D[i-1,T'']$. Hence, since $\min_{T'' \text{ extended by } T} D[i-1,T''] \leq D[i-1,T']$, we conclude that $D[i,T] \leq \Delta(i,T)+h'$ and, consequently, $D[i,T] \leq h$. □

From the correctness of the algorithm, it directly follows that:

Theorem 5.8 *Gapless MEC (without the all-heterozygous assumption) is in* FPT *when parameterized by the length of the fragments and it can be solved in* $O(3^{\ell} \cdot \ell \cdot n)$ *time.*

5.3 A 2-approximation Algorithm for Binary MEC

In this section we present a 2-approximation algorithm for Binary MEC, that is the restriction of MEC where the fragment matrix has values in $\{0,1\}$ only, and hence does not contain holes. The approximation algorithm is based on the observation that heterozygous columns in binary matrices naturally encode bipartitions of the fragments and that, by Lemma 4.1, if the columns of a gapless fragment matrix are pairwise in accordance then the matrix is conflict free. In particular, Algorithm 1 builds a feasible solution $\texttt{SOL}[t]$ for each t in $\{1,\ldots,m\}$ assuming that p_t is the closest column to an (unknown) optimal bipartition O of the fragments. Each solution $\texttt{SOL}[t]$ corrects columns $p_{j'}$ with cost $H(p_{j'}) \leq d(p_t,p_{j'})$ into homozygous columns (equal to $\underline{1}$ or $\underline{0}$ depending on the best choice), whereas it corrects the remaining columns $p_{j''}$ with cost $d(p_t,p_{j''}) < H(p_{j''})$ into heterozygous columns equal (or complementary, depending on the best choice) to p_t. It is easy to see that $\texttt{SOL}[t]$ for each t in $\{1,\ldots,m\}$ is a feasible solution (by Lemma 4.1) and that its cost is exactly $cost_{\mathscr{M}}(p_t)$, as reported in Eq. 4.1.

Algorithm 1 is a 2-approximation algorithm for Binary MEC.

Algorithm 1 A 2-approximation algorithm for Binary MEC

Require: A $n \times m$ binary matrix $\mathscr{M}$
  for $t = 1$ **to** m **do** ▷ Assume that p_t is the column "closest" to O
    for $j = 1$ **to** m **do**
      if $H(p_j) \leq d(p_t, p_j)$ **then**
        Set p_j homozygous in $\mathtt{SOL}[t]$
      else
        Set p_j equal/complementary to p_t in $\mathtt{SOL}[t]$
  return $\arg\min_{\mathtt{SOL}[t]} cost_{\mathscr{M}}(p_t)$

Lemma 5.9 *Given a fragment matrix $\mathscr{M}$ without holes, if OPT is the optimum for Binary MEC on input $\mathscr{M}$, then Algorithm 1 returns in $O(m^2 n)$ time a feasible solution with cost OPT' such that $OPT' \leq 2 \cdot OPT$.*

Proof Assume that p_O is the column of $\mathscr{M}$ closest to an optimal bipartition O, that is $d(O, p_O) \leq d(O, p_j)$ for each j in $\{1, \ldots, m\}$ and assume that $d_H(O, p_O) \leq d_H(\overline{O}, p_O)$ (if $d_H(\overline{O}, p_O) < d_H(O, p_O)$ we can substitute O with $\overline{O}$ since they encode the same bipartition). Clearly, one such a column exists and $d_H(O, p_O) \leq d(O, p_j)$ for each j in $\{1, \ldots, m\}$. We show that, under this assumption, $d(p_O, p_j) \leq 2d(O, p_j)$. By the triangle inequality, $d_H(p_O, p_j) \leq d_H(p_O, O) + d_H(O, p_j)$. Hence, since $d_H(p_O, O) \leq d(O, p_j) \leq d_H(O, p_j)$, we have $d_H(p_O, p_j) \leq 2d_H(O, p_j)$. Similarly, we can prove that $d_H(p_O, \overline{p_j}) \leq 2d_H(O, \overline{p_j})$. As a consequence we have that $d(p_O, p_j) \leq 2d_H(O, p_j)$ and that $d(p_O, p_j) \leq 2d_H(O, \overline{p_j})$, which then imply $d(p_O, p_j) \leq 2d(O, p_j)$. Clearly, since $d(p_O, p_j) \leq 2d(O, p_j)$, we also have that $\min(d(p_O, p_j), H(p_j)) \leq 2\min(d(O, p_j), H(p_j))$.

Since Algorithm 1 iteratively assumes that each column p_j is the closest column to the unknown optimal bipartition O, we have that the cost of the returned solution is $OPT' \leq cost_{\mathscr{M}}(p_O) \leq 2\sum_{j=1}^{m} \min(d(O, p_j), H(p_j)) = 2OPT$. Since each iteration t of the algorithm computes $\mathtt{SOL}[t]$ in $O(mn)$ time, the total running time is clearly equal to $O(m^2 n)$. □

Algorithm 1 runs in $O(m^2 n)$ time and, due to its simplicity, it is a more direct and practical approach than the PTAS algorithms known in the literature [80, 117].

5.4 Parameterized Tractability of k-ploid MEC

In the following sections, we propose two algorithms to prove that k-ploid MEC is in FPT when parameterized by coverage and number of haplotypes (Section 5.4.1) and when parameterized by fragment length and number of haplotypes (Section 5.4.2). The choice of parameters is reasonable, since, depending on the characteristics of

sequencing technologies, coverage or fragment length are usually limited by small constants. Moreover, species that naturally have more than 8 haplotypes are quite rare among higher-order organisms.

5.4.1 k-ploid MEC is in FPT when Parameterized by Coverage and Number of Haplotypes

In this section, we introduce a fixed-parameter tractable algorithm for k-ploid MEC when parameterized by the coverage cov and the number k of haplotypes. The algorithm is based on a dynamic programming approach and aims at finding a k-partition of minimum cost for all the fragments. This algorithm extends on the k-ploid case the general idea used in [120] for solving the diploid MEC problem.

The algorithm iteratively proceeds column-wise and, at each step, computes a k-partition for the fragments covered by the columns considered so far. The key idea for the fixed-parameter tractability of the algorithm is to show that an optimal k-partition for all the fragments can be built through this iterative procedure by adding, at each step, only a subset of fragments whose cardinality is limited by the coverage.

Let $\mathcal{M}$ be a fragment matrix and p_j be one of its columns. Then, we denote with $\mathcal{F}^j = (\mathcal{F}^j_1, \ldots, \mathcal{F}^j_k)$ a k-partition for the fragments in $\mathcal{A}_{\mathcal{F}}(p_j)$. Let p_{j_1}, p_{j_2} be two columns of $\mathcal{M}$, then we say that $\mathcal{F}^{j_2}$ *extends* $\mathcal{F}^{j_1}$ if and only if, for each $e \in \{1, \ldots, k\}$, $\mathcal{F}^{j_1}_e \cap \mathcal{A}_{\mathcal{F}}(p_{j_2}) \subseteq \mathcal{F}^{j_2}_e$. If a k-partition $\mathcal{F}^{j_1}$ is extended by a k-partition $\mathcal{F}^{j_2}$, it follows that we can easily build a k-partition $\mathcal{F}^{j_1,j_2}$ for the fragments in $\mathcal{A}_{\mathcal{F}}(p_{j_1}, p_{j_2})$ such that $\mathcal{F}^{j_1,j_2}$ extends both $\mathcal{F}^{j_1}$ and $\mathcal{F}^{j_2}$. Therefore, it is easy to see that any k-partition for all the fragments covered by the first j columns can be built starting from a k-partition for all the fragments covered by the first $j-1$ columns that induces a k-partition $\mathcal{F}^{j-1}$ for $\mathcal{A}_{\mathcal{F}}(j-1)$ and a k-partition $\mathcal{F}^j$ for the fragments in $\mathcal{A}_{\mathcal{F}}(p_j)$ where F^j extends $\mathcal{F}^{j-1}$.

We define $D[j, \mathcal{F}^j]$ as the minimum number of corrections to obtain a k-conflict free matrix $\mathcal{M}'$ from $\mathcal{M}$ on the first j columns such that $\mathcal{M}'$ induces a k-partition $\mathcal{F}^j$ for the fragments in $\mathcal{A}_{\mathcal{F}}(p_j)$. The value $D[j, \mathcal{F}^j]$ for each column p_j and for each k-partition $\mathcal{F}^j$ for $\mathcal{A}_{\mathcal{F}}(p_j)$ can be computed by the following recursive equation:

$$D[j, \mathcal{F}^j] = \Delta(j, \mathcal{F}^j) + \min_{\mathcal{F}^{j-1} \text{ extended by } \mathcal{F}^j} D[j-1, \mathcal{F}^{j-1}] \tag{5.4}$$

where $\mathcal{F}^{j-1}$ is a k-partition for $\mathcal{A}_{\mathcal{F}}(p_{j-1})$ and $\Delta(j, \mathcal{F}^j)$ is the "local contribution" of the k-partition $\mathcal{F}_j$ on column p_j. Informally, $\Delta(j, \mathcal{F}^j)$ is the minimum number of corrections needed for making the fragments in $\mathcal{A}_{\mathcal{F}}(p_j)$ k-conflict free on column p_j according to the k-partition $\mathcal{F}^j$. Such a cost can be easily computed as $\Delta(j, \mathcal{F}^j) = \sum_{e=1}^{k} \min(\#^0(\mathcal{F}^j_e), \#^1(\mathcal{F}^j_e))$ where $\#^u(\mathcal{F}^j_e) = |\{i \mid p_j[i] = u\}|$, *i.e.* the number of fragments in $\mathcal{F}^j_e$ with value u in column p_j.

The base case of the recurrence is $D[1,\mathscr{F}^1] = \Delta(1,\mathscr{F}^1)$ for each k-partition $\mathscr{F}^1$ for the fragments in $\mathscr{A}_{\mathscr{F}}(p_1)$. The optimum is $\min_{\mathscr{F}^m} D[m,\mathscr{F}^m]$ and a corresponding optimal k-partition for all the fragments of the input fragment matrix can be computed by backtracking.

The algorithm computes the entries $D[j,\mathscr{F}^j]$ for each k-partition $\mathscr{F}^j$ of the fragments in $\mathscr{A}_{\mathscr{F}}(p_j)$ and for each j in $\{1,\ldots,m\}$. Since the number of k-partitions for cov elements is k^{cov}, it follows that there are $O(k^{cov}\, m)$ entries. Given a k-partition $\mathscr{F}^j$, we need $O^*(k^{cov})$ time[2] to enumerate all the k-partitions $\mathscr{F}^{j-1}$ extended by $\mathscr{F}^j$ because we have to partition all the fragments in the set $\mathscr{A}_{\mathscr{F}}(p_{j-1}) \setminus \mathscr{A}_{\mathscr{F}}(p_j)$ (whose cardinality is clearly, at most, cov). Since $\Delta(j,\mathscr{F}^j)$ can be computed in polynomial time, each entry $D[j,\mathscr{F}^j]$ can be computed in $O^*(k^{cov})$. It follows that the total running time of the algorithm is $O^*(k^{2cov})$. Notice that, by storing partial information during the computation (as suggested in Section 5.2), we can decrease the complexity to $O^*(k^{cov})$. We omitted a detailed analysis of the polynomial factors in the time complexity since it would require to explicitly describe how fragments and partitions are represented and manipulated but it would not be useful for our purpose of characterizing the parameterized complexity of the problem.

In conclusion, we prove the correctness of the algorithm.

Lemma 5.10 *Consider a fragment matrix $\mathscr{M}$.*

1. *If $D[j,\mathscr{F}^j] = g$, then there exists a k-conflict free matrix $\mathscr{M}'$ obtained from $\mathscr{M}$ on the first j columns with g corrections that induces a partition $\mathscr{F}^j$ for the fragments in $\mathscr{A}_{\mathscr{F}}(p_j)$.*

2. *If $\mathscr{M}'$ is a k-conflict free matrix obtained from $\mathscr{M}$ on the first j columns with g corrections that induces a partition $\mathscr{F}^j$ for the fragments in $\mathscr{A}_{\mathscr{F}}(p_j)$, then $D[j,\mathscr{F}^j] \leq g$.*

Proof We prove the lemma by induction on j. Both statements obviously hold for $j = 1$. Assume that lemma holds for $j-1$, we show that both statements also hold for j.

(1) By Eq. (5.4), there exists a k-partition $\mathscr{F}^{j-1}$ for fragments in $\mathscr{A}_{\mathscr{F}}(p_{j-1})$ such that $\mathscr{F}^{j-1}$ is extended by $\mathscr{F}^j$ and $D[j,\mathscr{F}^j] = \Delta(j,\mathscr{F}^j) + D[j-1,\mathscr{F}^{j-1}]$. By induction there exists a k-conflict free matrix $\mathscr{M}''$ obtained from $\mathscr{M}$ on the first $j-1$ columns with at least $D[j-1,\mathscr{F}^{j-1}]$ corrections that induces a partition $\mathscr{F}^{j-1}$ for the fragments in $\mathscr{A}_{\mathscr{F}}(p_{j-1})$. Since $\mathscr{F}^{j-1}$ is extended by $\mathscr{F}^j$, there exists a k-partition $\mathscr{F}^{j-1,j}$ that induces $\mathscr{F}^{j-1}$ when restricted to $\mathscr{A}_{\mathscr{F}}(p_{j-1})$ and that induces $\mathscr{F}^j$ when restricted to $\mathscr{A}_{\mathscr{F}}(p_j)$. As a consequence, we can build a fragment matrix $\mathscr{M}'$ which is equal to $\mathscr{M}''$ on the first $j-1$ columns and whose j-th column is the correction

[2] We recall that $O^*(k^{cov})$ denotes the class $O(k^{cov}\,\mathrm{poly}(nm))$, where nm is the size of the input.

of p_j such that each part of $\mathscr{F}^j$ is in accordance. Such a correction, as explained before, needs to flip at least $\Delta(j,\mathscr{F}^j)$ elements of p_j. Since $\mathscr{M}''$ is k-conflict free and since no fragment in $\mathscr{A}_{\mathscr{F}}(p_j)\setminus\mathscr{A}_{\mathscr{F}}(p_{j-1})$ is covered by one of the first $j-1$ columns, $\mathscr{M}'$ is k-conflict free. Moreover, the total number of corrections needed to obtain $\mathscr{M}'$ from the first j columns of $\mathscr{M}$ is clearly equal to $\Delta(j,\mathscr{F}^j)+D[j-1,\mathscr{F}^{j-1}]=D[j,\mathscr{F}^j]$ by construction.

(2) Assume that $\mathscr{M}''$ is the submatrix of $\mathscr{M}'$ obtained from $\mathscr{M}$ on the first $j-1$ columns with g' corrections that induces a partition $\mathscr{F}^{j-1}$ for the fragments in $\mathscr{A}_{\mathscr{F}}(p_{j-1})$. Obviously, since $\mathscr{M}''$ is equal to the first $j-1$ columns of $\mathscr{M}'$, $\mathscr{M}''$ is k-conflict free and $\mathscr{F}^j$ extends $\mathscr{F}^{j-1}$. Since $\Delta(j,\mathscr{F}^j)$ is the minimum number of corrections needed to transform column p_j into column p'_j of $\mathscr{M}'$ such that each part of $\mathscr{F}^j$ is in agreement, we have that $\Delta(j,\mathscr{F}^j)+g'\leq g$. By induction we know that $D[j-1,\mathscr{F}^{j-1}]\leq g'$, hence we have that $D[j,\mathscr{F}^j]\leq\Delta(j,\mathscr{F}^j)+D[j-1,\mathscr{F}^{j-1}]\leq g$. □

From the correctness of the algorithm, it directly follows that:

Theorem 5.11 *k-ploid MEC is in* FPT *when parameterized by coverage and number of haplotypes.*

5.4.2 k-ploid MEC is in FPT when Parameterized by Fragment Length and Number of Haplotypes

In this section, we introduce a fixed-parameter tractable algorithm for k-ploid MEC when parameterized by the fragment length ℓ and the number of haplotypes k. Recall that, for many applications, both parameters are bounded by small constants. For example, the widespread Illumina sequencing technologies produce reads spanning only one or a few SNP positions and most species do not have more than 4–8 haplotypes.

This algorithm is based on a different approach than that presented in Section 5.2 for diploid MEC because that algorithm heavily relies on the characterization of conflict free fragment matrices given by Lemma 5.5 that cannot be easily extended to the k-ploid case. Indeed, one of the key ingredients for proving the existence of the tripartition T in Lemma 5.5 (assertion 3) is that each heterozygous column implicitly encodes a partial solution (that is, a bipartition of the fragments covered by that column). As a consequence, the existence of a solution for the fragments covered by any pair of heterozygous columns (i.e., their accordance) can be easily checked by testing the equality or the complementarity between the two columns on their shared active fragments. This idea cannot be directly applied to the k-ploid case, since a single heterozygous column is clearly not sufficient to encode a k-partition (with k non-empty parts). Groups of h columns (for some fixed h)

could be a natural encoding of k-partitions, but, in this case, we cannot check their accordance only by testing the equality or complementarity between these groups (as we do in the diploid case). Hence, a characterization of k-conflict free fragment matrices based on the existence of a partition for the columns (analogous to the tripartition T of the diploid case) seems not straightforward. Notice that the algorithm we introduce in this section for k-ploid MEC can be clearly applied also to diploid MEC, but the one designed in Section 5.2 has a better time complexity.

The novel algorithm uses an approach similar to that of [73]. In particular, the algorithm, rather than k-partitioning the fragments, aims at the direct reconstruction of k haplotypes such that each fragment aligns to one of them with the minimum total amount of mismatches. Clearly, this is equivalent to computing a k-conflict free fragment matrix $\mathscr{M}'$ obtained from $\mathscr{M}$ with the minimum amount of corrections. Indeed, given k haplotypes, a k-conflict free matrix $\mathscr{M}'$ can be computed in polynomial time by correcting each fragment in a such of way that it perfectly aligns to its closest haplotype (in terms of Hamming distance on non-hole elements). Intuitively, the algorithm, for each column p_j, computes the minimum number of mismatches needed for aligning each fragment ending at column p_j or before to a set of k haplotypes. Such a minimum number of mismatches is computed by partitioning the fragments into two parts: (*i*) those ending exactly at column p_j and (*ii*) those ending strictly before (that is, on the left of) column p_j. The key observation for reducing the time complexity is that the set of fragments ending at column p_j (denoted with $\mathscr{E}(j)$) aligns to a subvector (or a "window") of each haplotype whose length is at most ℓ, where ℓ is the maximum length of a fragment.

In the following, a *haplotype window* is an ℓ-long vector over $\{0,1\}$ and, as usual, given a vector v, $v[i_1 : i_2]$ represents the subvector of v between positions i_1 and i_2 (included and 1-based). We say that a haplotype window $\hat{h}'$ *overlaps* with another haplotype window $\hat{h}$ if $\hat{h}'[2:\ell] = \hat{h}[1:\ell-1]$. We define $D[j,(\hat{h}_1,\ldots,\hat{h}_k)]$ as the minimum number of corrections needed by fragments in $\bigcup_{t=1}^{j}\mathscr{E}_t$ to reconstruct k haplotypes $(h_1,\ldots,h_k)$ such that, for each h_e, $h_e[j-\ell+1:j]$ is equal to $\hat{h}_e$. The algorithm proceeds column-wise computing the value $D[j,(\hat{h}_1,\ldots,\hat{h}_k)]$ for each column p_j and for each collection of k haplotype windows $(\hat{h}_1,\ldots,\hat{h}_k)$ as follows:

$$D[j,(\hat{h}_1,\ldots,\hat{h}_k)] = \Delta(j,(\hat{h}_1,\ldots,\hat{h}_k)) + \min D[j-1,(\hat{h}'_1,\ldots,\hat{h}'_k)] \quad \text{for each } \hat{h}'_e \text{ overlapping with } \hat{h}_e \text{ with } 1 \le e \le k \tag{5.5}$$

where $\Delta(j,(\hat{h}_1,\ldots,\hat{h}_k))$ is the cost needed to align the fragments in $\mathscr{E}(j)$ to the haplotype windows $\hat{h}_1,\ldots,\hat{h}_k$ and that can be easily computed as follows:

$$\Delta(j,(\hat{h}_1,\ldots,\hat{h}_k)) = \sum_{f_i \in \mathscr{E}(j)} \min_{\hat{h}_e} d_H(\hat{h}_e, f_i[j-\ell+1:j]) \tag{5.6}$$

where d_H is the Hamming distance between the two vectors. Without loss of generality and for the sake of simplicity, we assume that if $j-\ell+1 \leq 0$, then the expression $d_H(\hat{h}_e, f_i[j-\ell+1:j])$ is replaced with $d_H(\hat{h}_e[\ell-j+1:\ell], f_i[1:j])$ in Eq. 5.6.

The base case of the recurrence is $D[1,(\hat{h}_1,\ldots,\hat{h}_k)] = \Delta(1,(\hat{h}_1,\ldots,\hat{h}_k))$ for each collection of haplotype windows $(\hat{h}_1,\ldots,\hat{h}_k)$. Moreover, the algorithm returns the value $\min_{(\hat{h}_1,\ldots,\hat{h}_k)} D[m,(\hat{h}_1,\ldots,\hat{h}_k)]$ corresponding to the optimum, and a collection of k optimal haplotypes can be reconstructed by backtracking.

The algorithm computes all the values $D[j,(\hat{h}_1,\ldots,\hat{h}_k)]$ for each position j from 1 to m and for each collection of haplotype windows $(\hat{h}_1,\ldots,\hat{h}_k)$. Each haplotype window is an ℓ-long binary vector, hence the number of collections of k haplotype windows is $2^{k\ell}$. As a consequence, $O(m2^{k\ell})$ entries have to be stored for the backtracking phase. Furthermore, given a position j and a collection of haplotype windows $(\hat{h}_1,\ldots,\hat{h}_k)$, each entry $D[j,(\hat{h}_1,\ldots,\hat{h}_k)]$ can be computed by Eq. (5.5) in time $O^*(2^k)$. Indeed, the number of collections of k haplotype windows overlapping (element-wise) with $(\hat{h}_1,\ldots,\hat{h}_k)$ is 2^k and $\Delta(j,(\hat{h}_1,\ldots,\hat{h}_k))$ can be computed in polynomial time, which is needed to obtain the minimum Hamming distance between each fragment (there are at most *cov* fragments ending at each position) and a haplotype window (of length ℓ) of the collection. It follows that the total running time is $O^*(2^{k\ell+k})$ and, by storing partial information, it can be decreased to $O^*(2^{k\ell})$.

The following lemma shows the correcteness of the algorithm.

Lemma 5.12 *Consider a fragment matrix $\mathcal{M}$.*

1. *If $D[j,(\hat{h}_1,\ldots,\hat{h}_k)] = g$, then k haplotypes $(h_1,\ldots,h_k)$ can be reconstructed with g corrections from the fragments in $\bigcup_{t=1}^{j}\mathcal{E}_t$ such that, for each h_e, $h_e[j-\ell+1:j]$ is equal to $\hat{h}_e$.*

2. *If k haplotypes $(h_1,\ldots,h_k)$ are reconstructed from the fragments in $\bigcup_{t=1}^{j}\mathcal{E}_t$ with g corrections, then $D[j,(\hat{h}_1,\ldots,\hat{h}_k)] \leq g$ (with $\hat{h}_e = h_e[j-\ell+1:j]$, for each $e \in \{1,\ldots,k\}$).*

Proof We prove the lemma by induction on j. Both statements obviously hold for $j=1$. Assume that lemma holds for $j-1$, we show that both statements hold for j.

(1) By Eq. (5.5) there exists a collection of haplotype windows $(\hat{h}'_1,\ldots,\hat{h}'_k)$ overlapping (element-wise) with $(\hat{h}_1,\ldots,\hat{h}_k)$ such that $D[j,(\hat{h}_1,\ldots,\hat{h}_k)] = \Delta(j,(\hat{h}_1,\ldots,\hat{h}_k)) + D[j-1,(\hat{h}'_1,\ldots,\hat{h}'_k)]$. By induction, there exists k haplotypes $(h'_1,\ldots,h'_k)$ that can be reconstructed with $D[j-1,(\hat{h}'_1,\ldots,\hat{h}'_k)]$ corrections from the fragments in $\bigcup_{t=1}^{j-1}\mathcal{E}_t$ such that, for each h'_e, $h'_e[j-\ell:j-1]$ is equal to $\hat{h}'_e$. As a consequence, $h'_e[j-\ell+1:j-1]$ is equal to $\hat{h}_e[1:\ell-1]$ for each e in $\{1,\ldots,k\}$. Let $(h_1,\ldots,h_k)$ be the collection of j-long haplotypes such that $h_e[1:j-1] = h'_e$ and $h_e[j] = \hat{h}_e[\ell]$, for each e in $\{1,\ldots,k\}$. Each fragment in $\mathcal{E}(j)$ aligns to one of $(h_1,\ldots,h_k)$ with $\Delta(j,(\hat{h}_1,\ldots,\hat{h}_k))$ total mismatches while, by induction, the fragments in $\bigcup_{t=1}^{j-1}\mathcal{E}_t$ align with $D[j-1,(\hat{h}'_1,\ldots,\hat{h}'_k)]$

total mismatches. Hence, $(h_1,\ldots,h_k)$ is a solution of k-ploid MEC on the fragments $\bigcup_{t=1}^{j}\mathscr{E}_t$ with cost $g=\Delta(j,(\hat{h}_1,\ldots,\hat{h}_k))+D[j-1,(\hat{h}'_1,\ldots,\hat{h}'_k)]$.

(2) Let $(\hat{h}'_1,\ldots,\hat{h}'_k)$ and $(\hat{h}_1,\ldots,\hat{h}_k)$ be the collections of haplotype windows such that $\hat{h}'_e$ is equal to $h_e[j-\ell:j-1]$ and $\hat{h}_e$ is equal to $h_e[j-\ell+1:j]$ for each e in $\{1,\ldots,k\}$. We assume that g' is the number of corrections needed by fragments in $\bigcup_{t=1}^{j-1}\mathscr{E}_t$ to reconstruct the haplotypes $(h_1[1:j-1],\ldots,h_k[1:j-1])$. By induction, it follows that $D[j-1,(\hat{h}'_1,\ldots,\hat{h}'_k)]\leq g'$. By construction, $\Delta(j,(\hat{h}_1,\ldots,\hat{h}_k))$ is the minimum number of corrections that fragments in $\mathscr{E}(j)$ need for reconstructing the haplotype windows $(\hat{h}_1,\ldots,\hat{h}_k)$. Hence, $g\geq\Delta(j,(\hat{h}_1,\ldots,\hat{h}_k))+g'\geq\Delta(j,(\hat{h}_1,\ldots,\hat{h}_k))+D[j-1,(\hat{h}'_1,\ldots,\hat{h}'_k)]\geq D[j,(\hat{h}_1,\ldots,\hat{h}_k)]$. □

From the correctness of the algorithm, it directly follows that:

Theorem 5.13 *k-ploid MEC is in* FPT *when parameterized by fragment length and number of haplotypes.*

Chapter 6

Methods for Long Reads

Some of the results in this chapter are published in:

Y. Pirola[†], S. Zaccaria[†], R. Dondi, G. W. Klau, N. Pisanti, and P. Bonizzoni. HapCol: accurate and memory-efficient haplotype assembly from long reads. *Bioinformatics*, 32(11): 1610, 2015.

[†]joint first authorship

This chapter includes all the details of the algorithm, called HapCol, that we design for haplotype assembly from long reads. Firstly, we describe in Section 6.1 the dynamic programming algorithm of HapCol, we analyze its time and space complexity, and we design an approach for substantially improving its running time. Next, we design in Section 6.2 the backtracking procedure needed to reconstruct the resulting haplotypes from the optimum computed by the dynamic programming. Some details concerning the implementation of the algorithm and its input parameters are reported in Section 6.3. Lastly, we formally prove the correctness of the dynamic-programming algorithm in Section 6.4.

6.1 HapCol

In this section we present a fixed-parameter tractable algorithm for solving the k-cMEC problem, when parameterized by the maximum number k of corrections that are allowed in each column and by the coverage cov. The algorithm is based on an exact dynamic programming approach. After presenting the basic dynamic programming equation, we show that the approach can be easily adapted to manage gaps and, possibly, the all-heterozygous assumption.

Informally, the algorithm iteratively computes, for all j from 1 to m, a k-corrected matrix $\mathcal{M}'$ on the first j columns $p_1, \ldots, p_j$ of the input matrix $\mathcal{M}$ by con-

sidering all the possible corrections p'_j for the last column p_j such that $d(p_j, p'_j) \leq k$ and choosing the best option. The corrected column p'_j can be either homozygous or heterozygous. If it is homozygous, we pay a cost equal to the homozygous distance $H(p_j)$ of p_j but then p'_j is in accordance with any other column and no other check must be performed. If p'_j is heterozygous, then we consider all the possible k-corrections B_j for p_j in β_j and for each one two different cases may arise: (1) there exists a column p_q with $q < j$ that "shares" some fragments with p_j and that in the optimal solution p'_q is heterozygous (clearly, $q \geq j - L$); (2) all the previous columns that share some fragments with p_j are homozygous in the optimal solution $\mathscr{M}'$. In the first case, p'_q and p'_j must be either identical of complementary on the shared fragments (Lemma 4.1). It follows that we have to choose the best option among all the k-corrections B_q such that $d(B_q, B_j) = 0$ and we pay a cost equal to that of the correction B_j (i.e., $d(p_j, B_j)$) plus the cost of transforming the columns between p_q and p_j into homozygous columns (i.e., their homozygous distance). In the second case, all the columns to the left of p_j that share some fragment with p_j are homozygous in the optimal solution $\mathscr{M}'$ (and we pay a total cost equal to their homozygous distance). As a consequence, any k-correction B_j of p_j is in accordance with them and we pay a cost equal to $d(p_j, B_j)$.

More formally, let $\mathscr{M}$ be a fragment matrix and B_j be a k-correction for p_j, we define $D[j, B_j]$ as the minimum number of corrections needed to obtain a k-corrected matrix $\mathscr{M}'$ for $\mathscr{M}$ on columns $p_1, \ldots, p_j$ such that p'_j is heterozygous and $d(p'_j, B_j) = 0$. Moreover, we define $OPT[j]$ as the minimum number of corrections needed to obtain a k-corrected $\mathscr{M}'$ for $\mathscr{M}$ on columns $p_1, \ldots, p_j$. Finally, we define p_{L_j} (p_{R_j}, resp.) as the rightmost (leftmost, resp.) column to the left (right, resp.) of p_j such that $\mathscr{A}_{\mathscr{F}}(p_{L_j}, p_j) = \varnothing$ ($\mathscr{A}_{\mathscr{F}}(p_{R_j}, p_j) = \varnothing$, resp.); if it does not exist, p_{L_j} (p_{R_j}, resp.) corresponds to an empty column in position 0 ($m+1$, resp.). Note that $j - L_j \leq L$ and $R_j - j \leq L$.

Without loss of generality, we implicitly assume that there exists a dummy empty column p_0 in position 0 of the input $\mathscr{M}$. Thus, we can define $OPT[0] = 0$ and $D[0, \cdot] = 0$.

For $0 < j \leq m$, $D[j, B_j]$ and $OPT[j]$ can be computed as follows:

$$D[j, B_j] = \min \begin{cases} \min\limits_{\substack{q: L_j + 1 \leq q \leq j-1, \\ B_q : d(B_j, B_q) = 0}} D[q, B_q] + d(p_j, B_j) + \sum_{y=q+1}^{j-1} H(p_y) \\ OPT[L_j] + d(p_j, B_j) + \sum_{y=L_j+1}^{j-1} H(p_y) \end{cases} \tag{6.1}$$

$$OPT[j] = \min \begin{cases} OPT[j-1] + H(p_j) & // \ p_j \text{ is homozygous} \\ \min\limits_{\forall B_j} D[j, B_j] & // \ p_j \text{ is heterozygous} \end{cases} \tag{6.2}$$

The optimum cost is given by $OPT[m]$ and a corresponding optimal solution $\mathscr{M}'$ can be reconstructed by backtracking. Technical details about the backtracking procedure are in the following Section 6.2. In addition, the formal proof of correctness is presented in Section 6.4.

The algorithm can be easily adapted to the weighted version of k-cMEC. In this case, each non-hole element $p_i[h]$ of the input matrix $\mathscr{M}$ has a weight $w(p_i[h])$. Given a column p_j and any k-correction B_j in β_j, the key idea is to consider the weight $w(p_j,B_j)$ as the minimum sum of the weights in order to transform p_j in p'_j such that $d(p'_j,B_j)=0$ and to consider the weight $w_H(p_j)$ as the minimum sum of weights in order to transform p_j into a homozygous column. Hence, we want to minimize the sum of such weights by replacing $d(p_j,B_j)$ with $w(p_j,B_j)$ and $H(p_j)$ with $w_H(p_j)$ in the recursive equations.

Assume to consider a general fragment matrix $\mathscr{M}$ that may contain gaps. As explained before, any gap can be modeled as zero-weight elements. Since each of these elements can be equal to 0 or 1 with a cost of 0, we can adapt the algorithm such that for each column all the combinations of values for its gaps will be considered. It follows that any k-correction B_j for a column p_j is extended with any combination of values for its gaps and added to β_j.

Furthermore, the algorithm can be slightly modified in order to find a solution under the all-heterozygous assumption that forces to reconstruct two complementary haplotypes. In this case, the homozygous columns, both in the input and in the output, have to be considered as "special" heterozygous columns that place all the covered fragments in the same part of the fragments bipartition. Hence, we remove from the recursive equation the possibility to transform each column p_j into a homozygous column and we add to β_j the k-correction B_j that transforms p_j into a "special" heterozygous column.

We now show how the time complexity $O((\sum_{s=0}^{k}\binom{cov}{s})^2\cdot cov\cdot L\cdot m)$ of a naive implementation of the recursive equation can be reduced to $O(\sum_{s=0}^{k}\binom{cov}{s}\cdot cov\cdot L\cdot m)$ by a careful re-engineering of the algorithm.

First, given n elements, their (n,s)-combinations for all s between 0 and k are $\sum_{s=0}^{k}\binom{n}{s}$ and they correspond to all the k-corrections B_j in β_j. Such a set can be enumerated in lexicographic order in time $O(\sum_{s=0}^{k}\binom{cov}{s})$ [86, see Alg. T, chapter 7.2.1.3] and, using the same ideas, it is possible to compute the i-th element (for any arbitrary i) of this order in time $O(k)$.

In a direct implementation of the recurrence equations, there exists $O(m\cdot\sum_{s=0}^{k}\binom{cov}{s})$ entries $D[j,B_j]$ and m entries $OPT[j]$. Computing each entry $D[j,B_j]$ requires checking at most one entry $OPT[q]$ and checking at most $L\cdot\sum_{s=0}^{k}\binom{cov}{s}$ entries $D[q,B_q]$ (for any q in $\{L_j,\ldots,j-1\}$) in time $O(cov)$ each (for computing $d(B_j,B_q)$, since $cov_j,cov_q\leq cov$). Therefore, since $OPT[j]$ can be updated in

constant time during the computation of the entries $D[j,B_j]$, we have that the overall time complexity of the simple implementation is $O((\sum_{s=0}^{k}\binom{cov}{s})^2 \cdot cov \cdot L \cdot m)$.

The time complexity can be improved to $O(\sum_{s=0}^{k}\binom{cov}{s} \cdot cov \cdot L \cdot m)$ by computing an *intermediate projection table* and applying an approach inspired by the one presented in Patterson et al. [120]. Let $\mathscr{M}$ be a gapless fragment matrix. Given two columns p_{j_1} and p_{j_2}, and a k-correction B_{j_1} for p_{j_1}, we define $\pi_{j_2}(B_{j_1})$ as the vector of size $|\mathscr{A}_{\mathscr{F}}(p_{j_1},p_{j_2})|$ that is obtained from B_{j_1} by keeping only elements that correspond to fragments that are in $\mathscr{A}_{\mathscr{F}}(p_{j_1},p_{j_2})$. We define the intermediate projection table for each column p_j, for each q in $\{L_j+1,\ldots,j-1\}$ and for each vector C representing a possible correction of the positions in $\mathscr{A}_{\mathscr{F}}(p_j,p_q)$, as follows:

$$\tilde{D}[q,j,C] = \min_{\forall B_q \in \beta_q : d(C,\pi_j(B_q))=0} D[q,B_q]. \tag{6.3}$$

Entry $\tilde{D}[q,j,\pi_j(B_q)]$ (and $\tilde{D}[q,j,\overline{\pi_j(B_q)}]$, where $\overline{\pi_j(B_q)}$ is the complement of $\pi_j(B_q)$) can be filled in $O(cov)$ time (needed to compute $\pi_j(B_q)$) while computing $D[q,B_q]$ and, consequently, the asymptotic overall time complexity does not change. Intuitively, $\tilde{D}[q,j,C]$ corresponds to the minimum number of corrections to obtain a k-corrected matrix $\mathscr{M}'$ for $\mathscr{M}$ on the first q columns such that p'_q is heterozygous and $d(C,\pi_j(p'_q))=0$. As a consequence, Eq. (6.1) can be equivalently rewritten as:

$$D[j,B_j] = \min \begin{cases} \tilde{D}[q,j,\pi_q(B_j)] + d(p_j,B_j) + \sum_{y=q+1}^{j-1} H(p_y) \\ OPT[L_j] + d(p_j,B_j) + \sum_{y=L_j+1}^{j-1} H(p_y) \end{cases} \tag{6.4}$$

In other words, with this recurrence, each entry $D[j,B_j]$ is computed using the entries $\tilde{D}[\cdot,j,\cdot]$ and, at the same time, it is used to update the entries $\tilde{D}[j,\cdot,\cdot]$ and $OPT[j]$ without changing the asymptotic time complexity. Since there exists $O(m \cdot L \cdot \sum_{s=0}^{k}\binom{cov}{s})$ entries $\tilde{D}[j,p,\pi_p(B_j)]$ with p in $\{j+1,\ldots,L-1\}$, it follows that the overall time complexity is $O(\sum_{s=0}^{k}\binom{cov}{s} \cdot cov \cdot L \cdot m)$. Notice that $O(cov^{k+1} \cdot L \cdot m)$ is a more intuitive, but less tight, bound.

Concerning space complexity and according to Eq. (6.4), the intermediate projection table $\tilde{D}[\cdot,\cdot,\cdot]$ can be stored instead of table $D[\cdot,\cdot]$. This takes $O(L \cdot m \cdot \sum_{s=0}^{k}\binom{cov}{s})$ space, since for any column p_j we only consider all the values q in $\{L_j+1,\ldots,j-1\}$ and $j-L_j \leq L$. Therefore, since the algorithm iteratively proceeds column-wise, when it is at the step corresponding to the column p_j, we just need to consider the entries $\tilde{D}[y,\cdot,\cdot]$ for all the columns p_y with $j \leq y \leq R_j$. For this reason, we just need $O(L \cdot L \cdot \sum_{s=0}^{k}\binom{cov}{s})$ space to store that window of the projection table.

Furthermore, if we consider a general fragment matrix $\mathscr{M}$ modelling the gaps as zero-weight elements (using the approach described before), the number of k-corrections B_j in β_j for a column p_j increases to $2^g \cdot \sum_{s=0}^{k}\binom{\widehat{cov}}{s}$, where $\widehat{cov}$ is the

non-hole coverage and g is the maximum number of gaps in a column (hence 2^g is the number of all the combinations of values for gap elements). This is because we apply the corrections only to the non-hole elements, whereas we try all the possible values for the holes. As a consequence, the overall time complexity becomes In addition, as shown in Section 6.2, for the backtracking phase we need two tables requiring $O(m)$ and $O(m \cdot \sum_{s=0}^{k} \binom{\widehat{cov}}{s})$ space, respectively.

6.2 Backtracking

During the computation of the DP tables, we can store some information in order to easily reconstruct the corresponding optimal solution $\mathscr{M}'$ and, consequently, the two reconstructed haplotypes. The matrix $\mathscr{M}'$ can be obtained by backtracking. For this reason, we need to store two *backtrace tables* $BT_1[j]$ and $BT_2[j,B_j]$ for any column p_j and for any k-correction B_j in β_j. In order to reconstruct $\mathscr{M}'$, each entry $BT_1[j]$ encodes few information for any column p'_j. In particular, on the assumption that there is no heterozygous column p'_p in $\{p'_{j+1},\ldots,p'_{R_j-1}\}$, $BT_1[j]$ tells whether p'_j is heterozygous or homozygous. In the former case, $BT_1[j]$ also stores the optimal k-correction B_j, whereas in the latter case $BT_1[j]$ stores $\underline{0}$ or $\underline{1}$ as optimal correction. Furthermore, each entry $BT_2[j,B_j]$ encodes the index (if it exists) of the rightmost heterozygous column p'_q in $\{p'_{L_j+1},\ldots,p'_{j-1}\}$. When p'_q exists, $BT_2[j,B_j]$ also stores its optimal k-correction B_q and whether p'_q is equal or complementary to p'_j on the fragments in $\mathscr{A}_{\mathscr{F}}(p_q,p_j)$. Notice that we need another table $\tilde{D}'[\cdot,\cdot,\cdot]$ with the same size of $\tilde{D}[\cdot,\cdot,\cdot]$ to remember the optimal k-correction B_q with $d(\pi_q(B_j),B_q)=0$ to store in $BT_2[j,B_j]$, since $\pi_q(B_j)$ may be just a portion of B_q that we need to find the entry $BT_2[q,B_q]$. The storing of the backtrace table $BT_1[j]$ takes $O(m)$ space and $BT_2[j,B_j]$ takes $O(m \cdot \sum_{s=0}^{k} \binom{cov}{s})$ space since each entry of the tables encodes a constant number of information.

We also propose an alternative scheme for the backtrace table $BT_2[j,B_j]$ defining an entry $BT_2[q,j,\pi_q(B_j)]$ for each q in $\{L_j+1,\ldots,j-1\}$. Each entry $BT_2[q,j,\pi_q(B_j)]$ stores the optimal k-correction B_q and whether p'_q is equal or complementary to p'_j on the fragments in $\mathscr{A}_{\mathscr{F}}(p_q,p_j)$ (in the same way as $BT_2[j,B_j]$), under the assumption that p'_q is the rightmost heterozygous column in $\{p'_{L_j+1},\ldots,p'_{j-1}\}$. This option takes $O(m \cdot L \cdot \sum_{s=0}^{k} \binom{cov}{s})$ space, that is asymptotically worse than the first option. However, we notice that the second option needs less space in practice. We suggest two possible explanations: the first option needs the adding table $\tilde{D}'[\cdot,\cdot,\cdot]$ and the exact number of entries $\sum_{j=1}^{m}\sum_{q=j-1}^{L_j+1}\sum_{s=0}^{k}\binom{|\mathscr{A}_{\mathscr{F}}(p_j,p_q)|}{s}$ of the second option can be lower than the exact number of entries $\sum_{j=1}^{m}\sum_{s=0}^{k}\binom{cov_j}{s}$ of the first option.

6.3 Implementation

A prototypical implementation of HapCol is available under the terms of the GPL at http://hapcol.algolab.eu/. Since coverage varies across columns, HapCol adaptively adopts a different maximum number k_j of corrections for each column p_j computed as the smallest integer such that the probability that p_j contains more than k_j errors is at most α, with α given as input. Such a probability is computed assuming that sequencing errors are uniformly distributed with a substitution error rate ϵ (given as input), an assumption which reflects the characteristics of future-generation sequencing technologies. Therefore, the two parameters given in input to HapCol are ϵ and α and can be chosen by the user depending on the estimated sequencing (substitution) error rate and on the user's preference towards better performances (larger α) or increased probability of finding a feasible solution (smaller α).

The strategy currently implemented for choosing the maximum number of corrections per column assumes that errors are uniformly distributed. However, it can be easily modified in order to process datasets produced by technologies with different error profiles (even those with systematic errors, especially if the average error rate is low, such as current Illumina technologies) and/or to automatically increase the values k_j until a feasible solution exists.

6.4 Proof of Correctness

First, notice that from Lemma 4.1 easily follows Corollary 6.1.

Corollary 6.1 *Given a gapless fragment matrix $\mathcal{M}$, $\mathcal{M}$ is conflict free if and only if each submatrix $\mathcal{M}''$ induced by any subset of columns is conflict free.*

Then, the correctness of Eq. (6.1) and (6.1) is proved by the following lemmas.

Lemma 6.2 *Let M be a gapless matrix and $\mathcal{M}'$ a k-corrected matrix for $\mathcal{M}$ on the first j columns obtained with h corrections. Then:*

1. *If p'_j is heterozygous and B_j is a k-correction for p_j such that $d(p'_j, B_j) = 0$, then $D[j, B_j] \leq h$.*
2. *$OPT[j] \leq h$.*

Proof We prove the lemma by induction on the number m of columns of matrix $\mathcal{M}$. If $j = 0$, the lemma obviously holds since p_0 is the first dummy empty column, $D[0, \cdot] = 0$ and $OPT[0] = 0$ by definition.

Assume by induction that the lemma holds for each $i < j$.

Firstly, we prove statement 1. Let p'_q be the rightmost heterozygous column of $p'_1,\ldots,p'_{j-1}$.

Assume that $\mathscr{A}_{\mathscr{F}}(p'_q,p'_j) \neq \varnothing$. It follows that p'_q with $j-L_i+1 \leq q \leq j$ and that the columns $p'_{q+1},\ldots p'_{j-1}$ are homozygous. By Corollary 6.1, the matrix $\mathscr{M}''$ induced by $p'_1,\ldots,p'_q$ is conflict free and it is a k-corrected matrix. Since p'_q is heterozygous, a k-correction B_q for p_q exists in β_q with $d(p'_q,B_q)=0$ such that $\mathscr{M}''$ is obtained with h_q corrections. As consequence, $D[q,B_q] \leq h_q$ by induction. Therefore, since p'_j is heterozygous, a k-correction B_j for p_j exists in β_j with $d(p'_j,B_j)=0$ such that $d(B_q,B_j)=0$ by Lemma 4.1. Hence, the matrix $\mathscr{M}'$ induced by columns $p'_1,\ldots,\mathscr{M}'j$ can be computed with $h=h_q+d(B_j,p_j)+\sum_{i=q+1}^{j-1}H(p_i)$ corrections. Since $D[j,B_j] \leq \min\limits_{\forall B_{q'}\in\beta_q : d(B_{q'},B_j)=0} D[q,B_q]+d(B_j,p_j)+\sum_{i=q+1}^{j-1}H(p_i) \leq h_q+d(B_j,p_j)+\sum_{i=q+1}^{j-1}H(p_i)$, we conclude that $D[j,B_j] \leq h$.

Assume that $\mathscr{A}_{\mathscr{F}}(p'_q,p'_j)=\varnothing$. It follows that p'_q with $1 \leq q \leq L_j$ by the fact that there are no gaps and that the columns $p'_{L_j+1},\ldots p'_{j-1}$ are homozygous. By Corollary 6.1, the matrix $\mathscr{M}''$ induced by $p'_1,\ldots,p'_q$ is conflict free and it is a k-corrected matrix. As consequence, if $\mathscr{M}''$ is obtained with h corrections, then $OPT[L_j] \leq h_j$ by induction. Since the columns $p'_{L_j},\ldots,p'_j$ are in accordance by Lemma 4.1, $\mathscr{M}'$ can be obtained with $h=h_j+\sum_{i=L_j}^{j-1}H(p_i)+d(p_j,B_j)$. Since $D[j,B_j] \leq OPT[L_j]+\sum_{i=L_j}^{j-1}H(p_i)+d(p_j,B_j) \leq h_j+\sum_{i=L_j}^{j-1}H(p_i)+d(p_j,B_j)$, we conclude that $D[j,B_j] \leq h$.

Finally, we prove statement 2.

Assume that p'_j is homozygous. By Corollary 6.1, the matrix $\mathscr{M}''$ induced by $p'_1,\ldots,p'_{j-1}$ is conflict free and it is a k-corrected matrix. As consequence, if $\mathscr{M}''$ is obtained with h_j corrections, then $OPT[j-1] \leq h_j$ by induction. Since by Lemma 4.1 the columns $p'_1,\ldots,p'_j$ are in accordance, $\mathscr{M}'$ can be obtained with $h=h_j+H(p_j)$ corrections. Then $OPT[j] \leq OPT[j-1]+H(p_j) \leq h_j+H(p_j)$, we conclude that $OPT[j] \leq h$.

Assume that p'_j is heterozygous. It follows that there exists a k-correction B_j for p_j exists in β_j with $d(B_j,p'_j)=0$ such that $\mathscr{M}'$ is obtained with h corrections. As a consequence, $D[j,B_j] \leq h$ by statement 1. Since $OPT[j] \leq \min_{\forall B_i\in\beta_j} D[i,B_i] \leq D[j,B_j]$, we conclude that $OPT[j] \leq h$. □

Lemma 6.3 *Let M be a gapless matrix.*

1. *For any k-correction B_j of p_j in β_j, if $D[j,B_j]=h<\infty$, then there exists a k-corrected matrix $\mathscr{M}'$ obtained from M on the first j columns with h corrections, such that p'_j is heterozygous and $d(p'_j,B_j)=0$.*

2. *If $OPT[j] = h < \infty$, then there exists a k-corrected matrix $\mathscr{M}'$ obtained from M on the first j columns with h corrections.*

Proof We prove the lemma by induction on the number m of columns of matrix $\mathscr{M}$. If $j = 0$, the lemma obviously holds since p_0 is the first dummy empty column, since $D[0,\cdot\,] = 0$ and $OPT[0] = 0$ by definition.

Assume by induction that the lemma holds for each $i < j$.

Firstly, we prove statement 1.

Assume that Case 1 of Eq. (6.1) holds. It follows that a k-correction B_q for p_q exists in β_q with $L_j + 1 \leq q \leq j-1$ and $d(B_j,B_q) = 0$ such that $D[j,B_j] = D[q,B_q] + d(p_j,B_j) + \sum_{y=q+1}^{j-1} H(p_y) = h$. Since $h < \infty$, for any p_i with $q+1 \leq i \leq j-1$, it holds $H(p_i) < \infty$. By inductive hypothesis there exists a k-corrected matrix $\mathscr{M}''$ obtained with $D[q,B_q]$ corrections and induced by columns $p'_1,\ldots,p'_q$, where p'_q is heterozygous and $d(p'_q,B_q) = 0$. Let p'_j be a heterozygous column with $d(p'_j,B_j) = 0$. Since $d(p'_j,p'_q) = 0$, the matrix $\mathscr{M}'$ induced by columns $p'_1,\ldots,p'_q,p'_{q+1},\ldots,p'_{j-1},p'_j$, where $p'_{q+1},\ldots,p'_{j-1}$ are homozygous, is conflict free by Lemma 4.1, it is a k-corrected matrix by construction and it can be obtained with $h = D[q,B_q] + \sum_{i=q+1}^{j-1} H(p'_i) + d(p_j,B_j)$ corrections.

Assume that Case 2 of Eq. (6.1) holds. It follows that $D[j,B_j] = OPT[L_j] + d(p_j,B_j) + \sum_{y=L_j+1}^{j-1} H(p_y) = h$. Since $h < \infty$, for any p_i with $L_j + 1 \leq i \leq j-1$, it holds $H(p_i) < \infty$. By inductive hypothesis there exists a k-corrected matrix $\mathscr{M}''$ obtained with $OPT[L_j]$ corrections and induced by columns $p'_1,\ldots,p'_{L_j}$. Let p'_j be a heterozygous column with $d(p'_j,B_j) = 0$. Since $\mathscr{A}_{\mathscr{F}}(p'_j,p'_{L_j}) = \varnothing$, the matrix $\mathscr{M}'$ induced by columns $p'_1,\ldots,p'_{L_j},p'_{L_j+1},\ldots,p'_{j-1},p'_j$, where $p'_{L_j+1},\ldots,p'_{j-1}$ are homozygous, is conflict free by Lemma 4.1, it is k-corrected by construction and it can be obtained with $h = OPT[L_j] + d(p_j,B_j) + \sum_{y=L_j+1}^{j-1} H(p_y)$ corrections.

Finally, we prove statement 2.

Assume that Case 1 of Eq. (6.2) holds. It follows that $OPT[j] = OPT[j-1] + H(p_j) = h$. Since $h < \infty$, it holds $H(p_j) < \infty$. By inductive hypothesis there exists a k-corrected matrix $\mathscr{M}''$ obtained with $OPT[j-1]$ corrections and induced by columns $p'_1,\ldots,p'_{j-1}$. Let p'_j be a homozygous column. Since the columns $p'_1,\ldots,p'_{j-1},p'_j$ are in accordance by Definition 4.1, the matrix $\mathscr{M}'$ induced by $p'_1,\ldots,p'_{j-1},p'_j$ is conflict free by Lemma 4.1, it is k-corrected by assumption and it can be obtained with $h = OPT[j-1] + H(p_j)$ corrections.

Assume that Case 2 of Eq. (6.2) holds. It follows that $OPT[j] = \min_{\forall B_j \in \beta_j} D[j,B_j] = h$. We conclude by statement 1 that there exists a k-corrected matrix $\mathscr{M}'$ obtained with h corrections and induced by columns $p'_1,\ldots,p'_j$. □

Chapter 7

Results

Some of the results in this chapter are published in:

Y. Pirola[†], S. Zaccaria[†], R. Dondi, G. W. Klau, N. Pisanti, and P. Bonizzoni. HapCol: accurate and memory-efficient haplotype assembly from long reads. *Bioinformatics*, 32(11): 1610, 2015.

[†]joint first authorship

We experimentally compared accuracy and performance of HapCol with that of state-of-the-art haplotype assembly approaches on both real (Section 7.1) and simulated datasets (Section 7.2). The experimental comparison on the real long read dataset is focused on evaluating the accuracy of the tools under the *all-heterozygous* assumption since such a standard benchmark dataset has low average coverage (~3x) and contains only heterozygous SNP positions. We also assessed accuracy and performances of the tools while varying coverage, read length, and sequencing/indel error rate on simulated long read datasets with characteristics similar to those of the "future-generation" sequencing technologies that are currently (or soon) available (coverage up to 25x, read length up to 50 000 bases, substitution error rate up to 5%, and indel rate equal to 10%) [20, 76, 132].

We compared HapCol with three state-of-the-art haplotyping tools specifically designed for handling long reads, namely, RefHap, which was shown to be one of the most accurate heuristic methods [41], ProbHap, a recent probabilistic method which has been shown to be sensibly more accurate than RefHap [87], and WhatsHap, the first exact approach for the weighted MEC problem specifically designed for long reads [120, 121]. At higher coverages, applications such as SNP calling or validating which SNPs are really heterozygous in the given sample (for example, there could be a significant portion of positions that, due to sequencing errors, appears to be heterozygous, but that should be predicted as homozygous) become feasible and reliable [112]. However, since these applications require that haplo-

types are reconstructed without the all-heterozygous assumption, on the simulated datasets we only considered WhatsHap and HapCol as they do not rely on this assumption.

The analyses focused on the accuracy of the reconstructed haplotypes and on the performances of the tools. Accuracy of the reconstructed haplotypes has been evaluated in terms of *(switch) error rate* [18] (i.e., the number of inconsistencies over contiguous phased variants) and in terms of phased positions (i.e., the number of positions for which the tool gave a phase prediction over the total number of positions that can be phased using the fragments given as input). Performances of the tools has been evaluated in terms of running time and peak memory usage, as reported by the Unix utility `time`. All the tests have been performed on a server equipped with four Intel Xeon E5-4610v2 CPUs and 256GB of RAM.

7.1 Real Data

The real dataset (called "NA12878 dataset") is the one produced using a fosmid-based technology from the HapMap sample NA12878 by Duitama et al. [41]. This dataset is considered a standard benchmark for comparing haplotyping algorithms on long reads, since the haplotypes of individual NA12878 were independently and confidently reconstructed using the sequenced genomes of the individual and of her parents. The dataset is composed of 271 184 reads with average length of ~40kb and with average coverage of ~3x. The reference haplotypes are the trio-phased variant calls from the GATK resource bundle [34], filtered on the 1 252 769 positions that are also covered by the fragments of the NA12878 dataset.

HapCol, RefHap, ProbHap, and WhatsHap have been executed independently on each chromosome. HapCol and WhatsHap can be executed with or without the all-heterozygous assumption without affecting the exponential part of their time/space complexities. In this case, these two tools have been executed using the all-heterozygous assumption, since the positions covered by the dataset are heterozygous with high confidence and since the comparison between solutions obtained with different assumptions may lead to misleading results. Moreover, HapCol has been executed with $\epsilon = 5\%$ and $\alpha = 10^{-3}$ and, for this choice of the parameters, a feasible solution existed for each chromosome. Table 7.1 reports, for each tool, the overall error rate and the percentage of phased positions over all the phasable positions, the total running time, and the peak of memory for the whole dataset (i.e., for all the chromosomes).

On this dataset, HapCol reconstructed the most accurate haplotypes and phased the largest number of positions compared with the other tools. In particular, HapCol improves the accuracy obtained by WhatsHap, ProbHap and RefHap by around 6%, 43%, and 48%, respectively. Furthermore, HapCol is also the tool which

Tool	error [%]	phased [%]	time [*sec*]	mem. [GB]
HapCol	**1.91**	**99.88**	332	2.1
WhatsHap	2.02	99.73	172	23.9
ProbHap	3.36	98.02	1205	0.6
RefHap	3.68	97.75	**43**	**0.5**

Table 7.1: Comparison of four haplotyping tools on the NA12878 real dataset. Accuracy is given in terms of phasing error ("error") and total phased positions ("phased") of the reconstructed haplotypes, while performances are given in terms of total running time ("time") expressed in seconds and the peak memory usage ("mem.") expressed in GB. Best results for each column are highlighted in boldface.

phases the largest number of positions. In fact, HapCol phases 0.15% more positions than WhatsHap, 2.03% more than ProbHap, and 2.18% more than RefHap.

To the contrary, RefHap was the fastest and most memory-efficient tool among the four considered. This was expected, as RefHap is a heuristic-based method, while the other ones are exact (albeit they minimize different objective functions). Overall, all the tools can be run with modest/medium computing resources. Indeed, each one analyzed the dataset in less than 25 minutes and using less than 24GB of memory. However, while HapCol and WhatsHap concluded in a few minutes, ProbHap was significantly slower than the others (~20 minutes) and, possibly, it could not be able to scale to datasets with higher coverage.

HapCol and WhatsHap required significantly more memory than ProbHap and RefHap (4 times and 44 times, respectively). However, such a peak of memory usage is due to a small number of consecutive positions on chromosomes 2, 3, and 10 where coverage is high (up to 30x), but most of values are gaps (all but 2-4 non-gap alleles on average). In these regions the performances of HapCol and WhatsHap degrade since both approaches model the gaps as zero-weight elements and must essentially "guess" the alleles at those positions. Clearly, in this case phase prediction is not reliable and a simple pre-filtering step can easily find (and possibly remove) such positions from further analyses. If we exclude chromosomes 2, 3, and 10, then WhatsHap becomes the fastest tool (30 *sec*), followed by RefHap (35 *sec*), by HapCol (60 *sec*), and by ProbHap, that remains the slowest tool (956 *sec*). In terms of memory usage, HapCol turns out to be the most memory-efficient method (0.06 GB), followed by WhatsHap (0.16 GB), by ProbHap (0.48 GB), and by RefHap (0.54 GB).

7.2 Simulated Data

We used simulated datasets to assess how accuracy and performances change while the characteristics of the dataset (coverage, especially) vary. As motivated before, in this part we focused on the tools that can work also without the all-heterozygous assumption, namely HapCol and WhatsHap. The simulation of the datasets has been performed as in Patterson et al. [121]. The dataset consists of a ground truth, which was assembled by inserting all known variants of chromosomes 1 and 15 of J. Craig Venter's genome into the human reference genome (hg18), mapped simulated long reads of lengths 1 000, 5 000, 10 000, and 50 000 bases with varying uniform substitution rates 1% and 5%, and with a uniform indel distribution of 10% at 30x coverage. These rates reflect the characteristics of the long read data generated by the future-generation sequencing technologies [20, 76, 132]. From each set of simulated reads, five datasets was obtained by randomly extracting a maximal subset with (maximum) coverage of 15x, 20x, and 25x.

HapCol has been executed with two combinations of its input parameters – namely $\epsilon = 5\%$ with $\alpha = 10^{-2}$, and $\epsilon = 5\%$ with $\alpha = 10^{-3}$ – in order to assess the behavior of HapCol depending on the choice of the parameters. We remark that some fragment matrices could not admit a feasible solution for the k-cMEC problem with some choices of parameter k (depending on ϵ and α in the implementation), while the same instances have always a feasible solution for the (unconstrained) MEC problem. Both tools have been executed on all the instances, but HapCol terminated on some of them because no feasible solution existed for that choice of the input parameters. WhatsHap, which should be able to find a feasible solution for all the instances, computed a solution only for the instances with coverage 15x and 20x, while, as expected [121], it was not able to successfully conclude the execution on the instances with coverage 25x since it exhausted the available memory (256GB).

Table 7.2 reports, for any combination of input parameters ϵ and α, the number of instances with a feasible solution (column "feas."), the average error of the reconstructed haplotypes, the average running time, and the average memory usage over all the instances of a given chromosome (Venter chromosome 1 and chromosome 15), coverage (15x and 20x), and substitution error rate e (1% and 5%) (the indel error rate is fixed to 10%, thus is not reported). The results presented in the table refer only to the subset of instances which have a feasible solution for the k-cMEC problem and are averaged over the read lengths. Since WhatsHap was not able to successfully terminate on any instance with coverage 25x, the results on that subset of instances are separately reported (only for HapCol) on Table 7.3.

First, as expected, the number of instances with a feasible solution increases as the combination of parameters ϵ and α allows more corrections per column. Indeed, for $\epsilon = 5\%$ and $\alpha = 10^{-2}$ the maximum numbers of corrections per column (not

			Chromosome 15							Chromosome 1						
			feas.	error [%]		time [*sec*]		mem. [GB]		feas.	error [%]		time [*sec*]		mem. [GB]	
ϵ / α	*cov*	*e*	·/20	wh	hc	wh	hc	wh	hc	·/20	wh	hc	wh	hc	wh	hc
$5\%/10^{-2}$	15x	1	17	2.26	2.24	18	6	1.7	0.1	15	2.40	2.40	47	17	4.5	0.3
		5	20	1.98	1.98	19	6	1.8	0.1	8	2.42	2.44	46	17	4.4	0.3
	20x	1	18	1.77	1.76	487	53	52.9	0.6	7	1.84	1.84	1241	155	129.2	2.0
		5	18	1.76	1.76	490	48	52.7	0.6	4	2.07	2.08	1249	132	129.0	1.6
$5\%/10^{-3}$	15x	1	20	2.12	2.11	19	25	1.8	0.3	20	2.35	2.36	48	64	4.6	0.8
		5	20	1.98	1.98	19	22	1.8	0.3	19	2.35	2.35	49	56	4.7	0.7
	20x	1	20	1.82	1.81	485	218	52.8	2.2	19	1.95	1.94	1306	586	138.0	5.6
		5	20	1.67	1.67	497	200	53.6	2.0	19	2.07	2.08	1347	526	138.5	5.1

Table 7.2: Comparison between HapCol (hc) and WhatsHap (wh) on instances with coverage ("*cov*") 15x and 20x, substitution error rate ("*e*") 1% and 5%, and indel error rate fixed to 10% by considering the number of instances with feasible solutions ("feas."), the average phasing error ("error") of the reconstructed haplotypes, the average running time ("time"), and the average maximum used memory ("mem."). HapCol has been executed with two different combinations of ϵ and α: $5\%/10^{-2}$ and $5\%/10^{-3}$.

shown) are quite low and, as a consequence, a feasible solution does not exists for many instances, especially for those with high substitution error rates e. For the other combination of parameters (namely, $\epsilon = 5\%$ and $\alpha = 10^{-3}$) the number of instances with a feasible solution rapidly increases. This trend, albeit less evident for chromosome 15, is clear for chromosome 1. Noticeably, when $\epsilon = 5\%$ and $\alpha = 10^{-3}$, a feasible solution exists for all but 3 instances with coverage at most 20x.

In terms of accuracy of the reconstructed haplotypes, on all the instances HapCol obtained the same phasing error rate of WhatsHap, which, in turn, was shown to be competitive with other state-of-the-art approaches [121]. This observation supports the validity of the newly introduced k-cMEC problem as a computational model for haplotype assembly on long reads.

Albeit HapCol and WhatsHap achieve the same accuracy, in terms of performances HapCol is both faster and significantly more memory-efficient than WhatsHap. In particular, on average, HapCol is at least twice faster than WhatsHap when the coverage is 20x even for the largest values of maximum number k_j of corrections per column (that is, when $\epsilon = 5\%$ and $\alpha = 10^{-3}$). Moreover, with the same parameters and with read length of 10 000 bases (a typical average read length in the foreseeable future), HapCol is almost three times faster than WhatsHap, allowing to process a single dataset in less than 11 minutes (on average). Concerning memory usage, we observe the same general trend, except that differences are even more evident. In fact, the average memory usage of WhatsHap on chromosome 1 (the largest one) at coverage 20x is ~138GB, while HapCol requires only ~5GB.

		Chromosome 15				Chromosome 1			
e	*cov*	feas. ·/5	error [%]	time [*sec*]	mem. [GB]	feas. ·/5	error [%]	time [*sec*]	mem. [GB]
1	20x	5	1.40	311	3.8	5	1.66	832	9.5
	25x	5	1.25	1457	16.1	4	1.52	4272	40.7
5	20x	5	1.24	277	3.3	5	1.71	737	8.5
	25x	5	1.14	1466	15.2	4	1.55	4357	39.2

Table 7.3: Comparison between instances with coverage 20x and 25x for HapCol with $\epsilon = 5\%$ and $\alpha = 10^{-3}$. Read length was fixed to 50 000 bases and the attributes are the same of Table 7.2.

Moreover, on instances with read length at most 50 000 bases, WhatsHap requires up to 164GB, while HapCol never requires more than 10GB. As a consequence, with HapCol, the analysis of a genome-wide dataset at coverage 20x is feasible even on a standard workstation/small server.

As noticed before, three instances do not admit a feasible solution with $\epsilon = 5\%$ and $\alpha = 10^{-3}$. However, by setting $\epsilon = 5\%$ and $\alpha = 10^{-4}$, a feasible solution exists also for these instances and the error rate of the solution obtained by HapCol is equal to that obtained by WhatsHap. In terms of performance, HapCol is slower than WhatsHap on the single instance with coverage 15x and it has a similar running time of WhatsHap on the two instances with coverage 20x (~21 minutes, on average). Noticeably, the amount of memory required by HapCol on these two instances (~9GB, on average) is ~15 times lower than that required by WhatsHap (~143GB, on average), a further confirmation of the memory-efficiency of HapCol even when more corrections per columns are allowed. Furthermore, these results confirm that a simple strategy that progressively increases the number of corrections allowed in each column until a solution is found would be practicable, since it always leads to a solution while keeping the memory usage as low as possible.

A comparison between HapCol and WhatsHap is not possible on instances with coverage 25x, since WhatsHap was not able to solve these instances within the available amount of memory. Hence, we evaluated how accuracy and performances of HapCol vary between instances with coverage 20x and 25x (Table 7.3). For space constraints, we report the results only for $\epsilon = 5\%$ and $\alpha = 10^{-3}$, that is for the parameters for which the maximum number of instances has a feasible solution. Moreover, in order to not discard the effect of read lengths, we focused only on instances with read length 50 000 bases. Such a read length has been chosen since almost all these instances have a feasible solution for both coverage 20x and 25x. We observe that increasing coverage from 20x to 25x allows to improve accuracy (~9%)

of the reconstructed haplotypes (as we already observed for coverage 15x and 20x). Moreover, the increased accuracy is mainly due to a significant reduction (approx. -10% on average, data not shown for space constraints) of the number of ambiguous positions, leading to an increased number of phased SNP positions. Concerning performances, instances with coverage 25x can be still analyzed with modest computing equipments; indeed HapCol completed the tests on chromosome 1 in less than 73 minutes and using less than 40GB of main memory.

Chapter 8

Discussion

Minimum Error Correction is a prominent combinatorial problem for haplotype assembly. Investigating the approximation complexity and the fixed-parameter tractability of MEC has proven useful to develop practical haplotype assembly tools [8, 73, 120, 125]. Despite in this work we addressed some issues that were left open, some other theoretical questions still need an answer.

In this work, we showed that, under the Unique Games Conjecture, MEC is not approximable within any constant factor. However, the approximation complexity of Gapless MEC and the computational complexity of Binary MEC are still unknown. It would be interesting to explore whether Lemma 4.1, that we used in this work for achieving a direct 2-approximation algorithm for Binary MEC and an FPT algorithm for Gapless MEC, is also useful for answering to these open questions.

In Section 5.4.2, we presented a fixed-parameter algorithm for k-ploid MEC when parameterized by fragment length ℓ and the number k of haplotypes. If applied on diploid MEC, it has a worse time complexity than that specifically presented for the diploid case (Section 5.2). Unfortunately, the algorithm for diploid MEC relies on Lemma 5.5, that cannot be easily extended to the k-ploid case. For this reason, another interesting research direction is to investigate whether a novel definition of accordance can be proposed in order to extend both Lemma 4.1 and the characterization of conflict free fragment matrices given by Lemma 5.5 to the k-ploid case and, hence, to derive a parameterized algorithm based on such a characterization.

Recent advances in sequencing technologies are radically changing the characteristics of the produced data. For example, long gapless reads with sequencing errors uniformly distributed will likely be common in the near future. In this work, we start to study of algorithms that exploit these characteristics but further improvements (either of the underlying models or of the algorithm's time complexity) will be essential to face the growing availability of these data. Furthermore, the drop in sequencing costs allows large-scale studies of rare diseases. In fact, they are

usually caused by rare mutations that can only be reliably discovered by sequencing several related individuals. Hence, we expect an increasing interest in the study of new formulations extending MEC on structured populations (where additional constraints induced by the Mendelian laws of inheritance improve the accuracy of the reconstructed haplotypes [123]), as initially done in [72].

We have proposed an exact algorithm, called HapCol, for the weighted k-cMEC, a new variant of the wMEC problem that takes into account the main characteristics of future-generation sequencing technologies, namely the uniform distribution of sequencing errors and the increasing length of sequenced reads.

We showed that the haplotypes computed by HapCol on a real benchmark dataset are at least as accurate as those computed by current state-of-the-art approaches. This result supports the validity of the additional constraints imposed by the k-cMEC problem.

Furthermore, HapCol is able to overcome the traditional all-heterozygous assumption and to process datasets with coverage 25x on standard workstations/small servers, while the current state-of-the-art methods either rely on this assumption or become unfeasible on coverages over 20x. Thanks to these results, HapCol is potentially able to directly perform SNP calling or heterozygous SNPs validation that become feasible and reliable on coverage up to 25x.

HapCol has been specifically designed to exploit the uniform distribution of errors that characterizes "future-generation" sequencing technologies, but it can be successfully applied on sequencing data with a different error distribution by choosing the maximum number k of errors per position according to the error model. Furthermore, HapCol can be easily extended in order to adaptively increase the value of k (on certain columns) until a feasible solution exists. This strategy allows to process datasets affected by systematic sequencing errors without a great impact on the performance if the average error rate is low (such as in the current Illumina sequencing technologies).

An interesting future direction would be the extension of the k-cMEC problem to deal with individuals related by structures such as trios or pedigrees [18]. Indeed, the Mendelian laws of inheritance induce further constraints that may improve the accuracy of the reconstructed haplotypes, as shown for example by [123].

Part II

Quantification of Intra-Tumor Heterogeneity

Chapter 9

Background

Cancer results from an evolutionary process where somatic mutations accumulate in a population of cells during the lifetime of an individual [114]. Thus, a tumor consists of heterogeneous subpopulations of cells, or *clones*. Each clone comprises cells that share a unique complement of somatic mutations. Quantifying this intra-tumor heterogeneity has been shown to be important in cancer treatment [149]. While intra-tumor heterogeneity complicates the identification of mutations in bulk-sequencing data from a tumor sample containing millions of cells, it also provides a signal for inferring the tumor composition – the number and proportion of clones within a sample – as well as the ancestral history of somatic mutations during cancer development [63]. Thus, a number of methods have been developed to infer the *evolutionary history* of a tumor from DNA sequencing data from one or more samples [63, 67, 104, 113, 143]. The evolutionary history of the clones can be described by a phylogenetic tree whose leaves correspond to extant clones and whose edges are labeled by mutations. Computational inference of phylogenetic trees is a fundamental problem in species evolution [53], and has recently been studied extensively for tumor evolution in the case where mutations are single-nucleotide variants [47, 79, 100, 128, 155].

One class of mutations that are particularly useful for inferring tumor composition and tumor evolution are copy-number aberrations (CNAs), which include duplications and deletions of large genomic regions. Copy number aberrations include segmental deletions and amplifications that affect large genomic regions, and are common in many cancer types [26]. As a result of these events, the number of copies of genomic regions, or *segments*, along a chromosome can deviate from the diploid, two-copy state of each position in a normal chromosome. Understanding these events and the underlying evolutionary tree that relates them is important in predicting disease progression and the outcome of medical interventions [55]. CNAs can be readily detected from DNA sequencing data, in fact, intuitively, the copy number of each genomic segment is proportional to its copy number [21, 113, 148].

Their direct direct effect on sequencing reads and their presence in most tumors make them good candidates for identifying distinct clones and for inferring their evolutionary history. However, there are two major challenges in using CNAs to quantify intra-tumor heterogeneity and evolution.

The first challenge is that nearly all cancer sequencing studies perform bulk sequencing that measures mutations in a mixtures of millions of different cells in a tumor sample. As such, tumor samples are *mixed* since they correspond to heterogeneous compositions of distinct clones. While single-cell sequencing provides a higher resolution measurement of tumor heterogeneity, it remains a specialized technique that is cost prohibitive for whole genome analysis of thousands of cells [108]. Thus, we require techniques to deconvolve CNA measurements from mixed tumor samples. Typically, CNAs are detected in sequencing data by examining the depth of aligned sequencing reads to genomic regions and segmentation algorithms partition the genome into *segments* with the same copy number [10, 148]. However, when a sample is heterogeneous as result of a mixture of distinct clones, a fractional copy number may be obtained for each segment instead of an integer copy number.

A number of methods have been developed to infer tumor composition from fractional copy numbers, taking advantage of the fact that larger CNAs perturb thousands-millions of sequencing reads, providing a signal to infer their proportions, even with modest coverage sequencing [21, 54, 71, 113, 116, 148]. However, these methods have certain limitations that limit their applicability and performance. For example, ASCAT [148] and ABSOLUTE [21] use the data from heterogeneous samples for inferring the tumor purity (the proportion of normal clone in a sample), but they do not distinguish the copy numbers of different tumor clones. Other methods, such as THetA [116], Battenberg [113], cloneHD [54] and TITAN [71], infer the clonal composition independently for each sample by modeling the fractional copy numbers as the product of the integer copy numbers of the extant clones with their proportions, a process known as *deconvolution*. However, one can obtain more information by jointly considering more samples from the same tumor [63], as successfully done for single-nucleotide mutations [46, 99] or non-integer copy numbers [134]. Moreover, multiple ways to deconvolve fractional copy numbers may be present, especially without imposing a structure on the inferred CNAs. Therefore, the inference of distinct clones may benefit from jointly inferring their evolution.

The second challenge in using CNAs to quantify intra-tumor heterogeneity and evolution is that one requires a model of CNA evolution. Defining such a model is not straightforward because CNAs can overlap, and thus positions in the genome cannot be treated independently. Standard phylogenetic models represent a genome as a sequence of "characters" with mutations acting independently on

individual characters. A number of models have been introduced to study CNA evolution, and these models can be classified into two categories. The first considers *single events* such that each of those independently affects the copy number of a single segment [24, 104]. However, these models do not account for dependency between adjacent segments in the genome. The second, called MEDICC [136], considers the effects of CNAs on multiple segments as *interval events* that amplify or delete copies of contiguous segments. The input to MEDICC is a set of *copy-number profiles*, vectors of integers defining the copy-number state of each segment. These profiles are measured for multiple samples from a tumor using DNA microarrays or DNA sequencing. The edit distance (non symmetric) from profile **a** to **b** was defined as the minimum number of amplifications and deletions of segments required to transform **a** into **b**. Using this distance measure, the authors applied heuristics to reconstruct phylogenetic trees. However, all of the studies applying these methods either assume that each sample is homogeneous and consisting of a single clone [101, 138, 143] or first attempt to infer the clones independently on each sample before performing a phylogenetic analysis of CNAs [104].

This second part of the thesis concerns the inference of the CNAs of distinct tumor clones and their evolution. The contributions of this part are twofold. First, we improve the model of interval events started in [136]. In fact, the complexity of the method used by MEDICC was not analyzed and a formal analysis of some combinatorial aspects of this model was started only later by Shamir *et al*. [139] by proving that the distance from one profile to another can be computed in linear time. As such, we formalize the *Copy-Number Tree (CNT)* problem (Figure 10.2) that aims to find the most parsimonious evolution of clones explained by the minimum number of interval events (Chapter 10), we show that CNT is NP-hard (Chapter 11), and we derive an *integer linear programming (ILP)* that solves this problem (Chapter 12). We assess the efficiency and quality of our algorithm on simulated instances (Chapter 13).

Second, we propose an approach combining the deconvolution of fractional copy numbers from multiple samples with the inference of CNAs that describes the evolution of the clones. We introduce the *Copy-Number Tree Mixture Deconvolution (CNTMD)* problem (Figure 10.3) that aims to deconvolve the fractional copy numbers into the integer copy numbers of the extant clones and their proportions such that the evolution of the clones is explained by a minimum number of copy number aberrations modeled as interval events (Chapter 10). We design a coordinate-descent algorithm for solving this problem (Chapter 12) and we compare our method with alternative approaches on real-size simulations (Chapter 13). We find that combining the deconvolution of fractional copy numbers with a proper tree model based on interval events outperforms all other methods on different number of samples. We apply our method on multi-sample sequencing data of a prostate-cancer

patient [67] (Chapter 13). Our inference shows well-supported patterns that reveal the clonal composition in terms of CNAs.

Chapter 10

Problem Formulations

Some of the results in this chapter are published in:

M. El-Kebir, B. J. Raphael, R. Shamir, R. Sharan, S. Zaccaria, M. Zehavi, and R. Zeira. Copy-number evolution problems: Complexity and algorithms. In *International Workshop on Algorithms in Bioinformatics (WABI)*, pages 137–149. Springer, 2016.

S. Zaccaria[†], M. El-Kebir[†], G. Klau, and B. J. Raphael. The copy-number tree mixture deconvolution problem and applications to multi-sample bulk sequencing tumor data. In *Research in Computational Molecular Biology (RECOMB)*, 2017, in press.

[†]joint first authorship

In this chapter, we introduce the Copy-Number Tree (CNT) and Copy-Number Tree Mixture Deconvolution (CNTMD) problems. First, we define the integer copy-number profiles and the related interval events in Section 10.1. In Section 10.2 we formally define the copy-number tree that is used to describe the evolutionary history of distinct clones in terms of CNAs. Additionally, in the same Section 10.2 we formally introduce CNT that for given profiles seeks a copy-number tree composed of a minimum number of interval events. Finally, in Section 10.3 we introduce CNTMD aiming to deconvolve the fractional copy numbers of multiple heterogeneous samples into the profiles of distinct clones and their proportions such that the evolutionary history of these clones is described by a copy-number tree comprising a minimum number of interval events.

10.1 Profiles and Events

Following the model in [45, 136, 139], we represent a chromosome from the reference genome as a sequence of m *segments*. A *copy-number profile*, or *profile* for

short, specifies the number of copies of each segment in a clone. Formally, a profile $\mathbf{c}_i = [c_{s,i}]$ is a (column) vector of m integers whose entries $c_{s,i} \in \mathbb{N}$ indicate the number of copies of segment s in a clone i. For simplicity, we consider a single chromosome in the definitions.

We consider mutations that amplify or delete contiguous segments. A *interval event*, or *event* for short, increases or decreases copy numbers of contiguous segments of a profile $\mathbf{c}_i$. Formally, an event is a triple (s,t,b) with segments $s \leq t$ and integer $b \in \mathbb{Z}$. If b is positive then the event is an *amplification* and the non-zero segments between s and t are incremented by b, whereas for negative b the events is a *deletion* and the same segments are decremented by at most $|b|$ (segments must be non-negative). Thus, the event (s,t,b) applied on $\mathbf{c}_i$ results in $\mathbf{c}'_i$ such that

$$c'_{\ell,i} = \begin{cases} \max\{c_{\ell,i} + b, 0\}, & \text{if } s \leq \ell \leq t \text{ and } c_{\ell,i} \neq 0, \\ c_{\ell,i} & \text{otherwise.} \end{cases} \tag{10.1}$$

for each segment ℓ. Thus, once a segment ℓ has been lost, i.e. $c_{\ell,i} = 0$, it can never be regained (or deleted).

10.2 Copy-Number Tree (CNT)

We model the evolutionary process that led to n extant tumor clones by a *copy-number tree* T defined as follows.

Definition 10.1 *Given a number n of clones, a* copy-number tree *is a rooted full binary tree on n leaves, such that each vertex $v_i \in V(T)$ is labeled by a profile $\mathbf{c}_i$ and each edge (v_i, v_j) is labeled by a set $\mathscr{E}_{i,j}$ of events. The root vertex $r(T)$, corresponding to the* normal clone*, is diploid, i.e. $c_{s,r(T)} = 2$ for each segment s.*

The requirement that T is a full binary tree (each vertex of T has either zero or two children) is without loss of generality, as each vertex with out-degree larger than 3 of a general tree can be split into vertices of out-degree 2, and each vertex with out-degree 1 can be removed and the associated events assigned to the outgoing edge (Fig. 10.1). To account for the case where $r(T)$ has out-degree 1, given an instance $(\mathbf{c}_1, \ldots, \mathbf{c}_k, e)$ we solve a second instance $(\mathbf{c}_1, \ldots, \mathbf{c}_k, \mathbf{c}_{k+1}, e)$ with an additional profile $\mathbf{c}_{k+1}$ consisting of 2's. The result is the minimum-cost tree among the two instances. Thus, each vertex $v_i \in V(T)$ has either zero or two children and is labeled by a profile $\mathbf{c}_i$. To avoid degenerate solutions, we impose a maximum copy number $c_{\max} \in \mathbb{N}$ for each segment s of any vertex v_i of T such that $c_{s,i} \leq c_{\max}$. Moreover, each leaf $v_i \in L(T)$ corresponds to the clone i. As such, we order the vertices $V(T) = \{v_1, \ldots, v_{2n-1}\}$ such that $L(T) = \{v_1, \ldots, v_n\}$ and $r(T) = v_{2n-1}$. An edge $(v_i, v_j) \in E(T)$

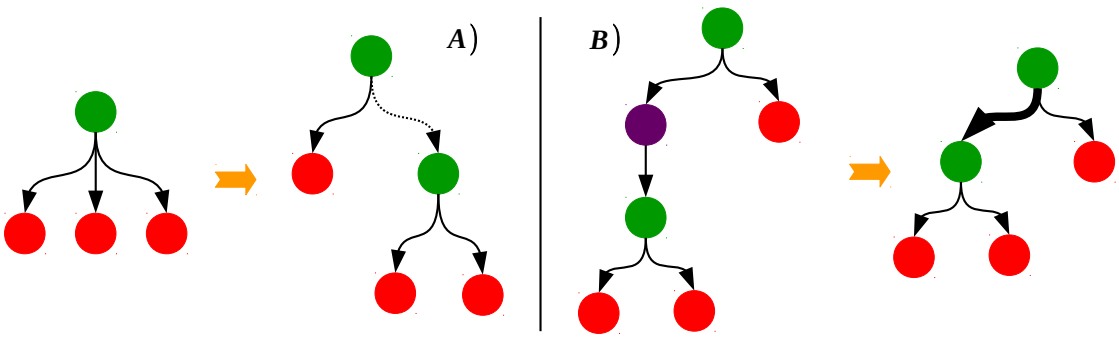

Figure 10.1: **Binarization of a general tree.** A) Each vertex with out-degree larger than 3 of a general tree can be split into vertices of out-degree 2 introducing 0-cost edges (dashed edge). B) Each vertex with out-degree 1 can be removed and the associated events assigned to the outgoing edge (wider edge).

relates a parent vertex v_i to its child v_j such that the label $\mathscr{E}(i,j)$ is a set of events that transform $\mathbf{c}_i$ to $\mathbf{c}_j$.

In general, the order in which events $\mathscr{E}(i,j)$ are applied matters. Following a result by Shamir et al. [139], it suffices to consider a set of events instead of an ordered sequence. In fact, any sequence of events, where amplifications and deletions occur in an arbitrary order, can be transformed into a *sorted* sequence, where deletions are followed by amplifications, without changing the cost of events, as defined in the following. The cost of an event (s,t,b) is the number of changes in the segment and is thus equal to $|b|$. Therefore, the cost $\Lambda(i,j)$ of an edge (v_i,v_j) is the total cost of the events in $\mathscr{E}(i,j)$, i.e. $\Lambda(i,j) = \sum_{(s,t,b)\in\mathscr{E}(i,j)} |b|$. The cost $\Lambda(T)$ of the tree T is the sum of the costs of all edges, i.e. $\Lambda(T) = \sum_{(v_i,v_j)\in E(T)} \Lambda(i,j)$.

Our observations correspond to the profiles $\hat{\mathbf{c}}_1,\ldots,\hat{\mathbf{c}}_n$ of n extant clones. Under the assumption of parsimony, the goal is to find a copy-number tree T^* of minimum cost whose leaves correspond to the extant clones. Furthermore, we assume that the maximum copy-number in the phylogeny is bounded by $c_{\max} \in \mathbb{N}$. We thus have the following problem.

Problem 10.1 (Copy-Number Tree (CNT)) *Given profiles $\hat{\mathbf{c}}_1,\ldots,\hat{\mathbf{c}}_n$ on m segments and an integer $c_{max} \in \mathbb{N}$, find a copy-number tree T^* such that (1) T^* has n leaves labeled by $\mathbf{c}_i = \hat{\mathbf{c}}_i$ for all $i \in \{1,\ldots,n\}$, (2) $c_{i,s} \leq c_{max}$ for all $v_i \in V(T^*)$ and $s \in \{1,\ldots,m\}$, and (3) $\Delta(T^*)$ is minimum.*

Note that by definition the profile of the root vertex $r(T)$ of any copy-number tree T is the vector whose entries are all 2's. As such, this must hold as well for the minimum-cost tree T^* which always exists.

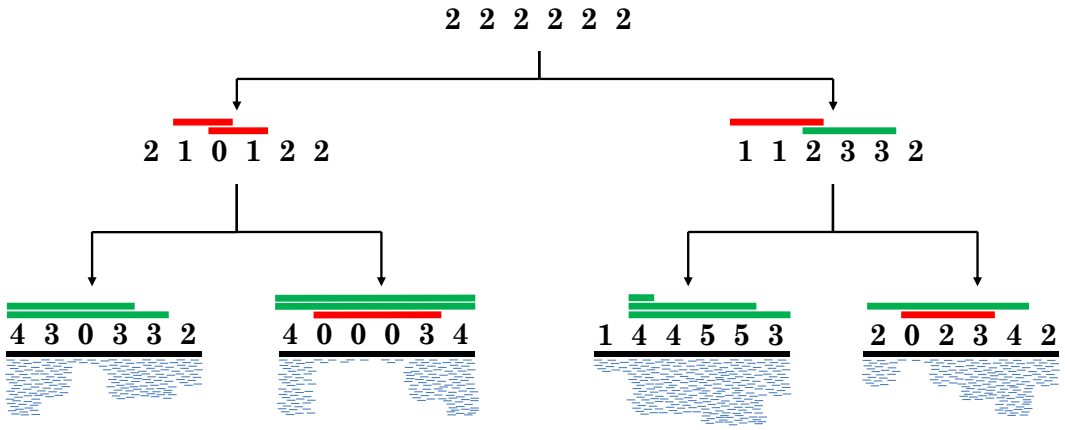

Figure 10.2: **Copy-Number Tree (CNT) problem.** As input we are given the copy-number profiles of four leaves, each profile is an integer vector that is inferred from data; e.g. the coverage of mapped reads (blue segments). The tree topology and profiles at internal vertices are found to minimize the total number of amplifications (green bars) and deletions (red bars). The displayed scenario has 14 total events.

10.3 Copy-Number Tree Mixture Deconvolution (CNTMD)

In the ideal case of single-cell sequencing data with no errors, each clone is a single cell and we observe the copy-number profiles $\hat{\mathbf{c}_1}, \ldots, \hat{\mathbf{c}_n}$ of n tumor clones. As such, in CNT we wish to find the most parsimonious explanation, i.e. a minimum-cost copy-number tree T^* whose n leaves are labeled by $\hat{\mathbf{c}_1}, \ldots, \hat{\mathbf{c}_n}$. However, with bulk-sequencing data the observations correspond to k *samples* obtained from a single tumor in different regions or at different time points. Each sample corresponds to a *mixture* of n extant clones of an unknown copy-number tree in unknown proportions. Recall that m is the number of segments. Our observations are thus described by the $m \times k$ *fractional copy-number matrix* $F = [f_{s,p}]$ where the *fraction* $f_{s,p} \in \mathbb{R}_{\geq 0}$ is the average copy number of segment s in sample p.

Let T be a copy-number profile tree with n leaves. We represent the profiles of the clones of T by the $m \times n$ *copy-number matrix* $C = [c_{s,i}]$ such that the i-th column of C corresponds to the profile $\mathbf{c}_i$ of clone i, i.e. $C = (\mathbf{c}_1, \ldots, \mathbf{c}_n)$. We say that C *generates* T if T is a copy-number tree whose leaves are labeled by the profiles in C and such that each internal vertex v_i is labeled by a profile c_i with $c_{s,i} \leq c_{\max}$ for each segment s. The $n \times k$ *usage matrix* $U = [u_{i,p}]$ describes the *mixing proportion* $u_{i,p} \in \mathbb{R}_{\geq 0}$ of clone i in sample p such that the sum of the mixing proportions for each sample p is 1. The observed fractional copy-number fraction F is thus modeled by $F = CU$. We have the following problem.

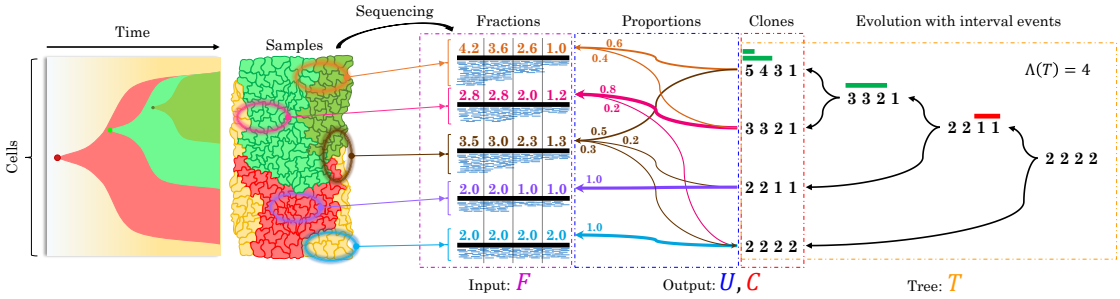

Figure 10.3: **Copy-Number Tree Mixture Deconvolution (CNTMD) problem.** A tumor acquires somatic mutations over time and thus consists of heterogeneous subpopulations of cells, or clones. The tumor clones are colored in red, dark and light green, whereas the normal clone is colored in yellow. Five samples are sequenced with bulk-sequencing technology yielding fraction copy-number matrix F. We model the evolution of the copy-number aberrations by a copy-number tree T (right), whose vertices are labeled by integer copy numbers, whose leaves correspond to clones, and whose edges are labeled by interval events. We combine the deconvolution of these fractional copy numbers (F) with the inference of the evolution of clones (T). More specifically, CNTMD deconvolves/factors F into the integer copy numbers C of the extant clones and their proportions U such that $F = CU$ and C generates by a copy-number tree T with the minimum number of interval events.

Problem 10.2 (Copy-Number Tree Mixture Deconvolution (CNTMD))
Given an $m \times k$ fractional copy-number matrix F, a number n of clones, and a maximum copy number c_{max}, find an $m \times n$ copy-number matrix C generating T^* and an $n \times k$ usage matrix U, such that $F = CU$ and $\Lambda(T^*)$ is minimum.

The goal of the problem is thus to deconvolve the observed fractional copy-numbers F as mixtures of the copy-number profiles C of the leaves of the most parsimonious tree T^*, according to mixing proportions U. Figure 10.3 depicts the scheme of the approach and its intuitive idea.

Chapter 11

On the Computational Complexity

Some of the results in this chapter are published in:

M. El-Kebir, B. J. Raphael, R. Shamir, R. Sharan, S. Zaccaria, M. Zehavi, and R. Zeira. Complexity and algorithms for copy-number evolution problems. *Algorithms for Molecular Biology*, 2016, in press.

In this chapter we show that CNT is NP-hard by reduction from the Maximum Parsimony Phylogeny (MPP) problem [58]. In MPP, we seek to find a *binary phylogeny* T, which is a full binary tree whose vertices are labeled by binary vectors of size n. The cost of a binary phylogeny T is defined as the sum of the Hamming distances of the two binary vectors associated with each edge. We are only given the leaves of an unknown binary phylogeny in the form of k binary vectors $\mathbf{b}_1, \ldots, \mathbf{b}_k$ of size n, and the task is to find a minimum-cost binary phylogeny T with k leaves such that each leaf $v_i \in L(T)$ is labeled by $\mathbf{b}_i$ and the root is labeled by a vector of all 0s. We consider the decision version where we are asked whether there exists a binary phylogeny T with cost at most h. This problem is NP-complete [58].

We start by defining the transformation (Fig. 11.1). Let $\mathbf{b}_1, \ldots, \mathbf{b}_k$ be an instance of MPP such that $|\mathbf{b}_i| = n$. The corresponding CNT-instance has parameter $e = 2$ and profiles $\mathbf{c}_1, \ldots, \mathbf{c}_{k+1}$ of length $n + (n-1)nk$. Each input profile $\mathbf{c}_i$, where $i \in \{1, \ldots, k\}$, is defined as

$$\begin{aligned} \mathbf{c}_i = \phi(\mathbf{b}_i) = \\ \begin{pmatrix} \phi(b_{i,1}) & \Omega & \phi(b_{i,2}) & \Omega & \cdots & \Omega & \phi(b_{i,k}) \end{pmatrix} \end{aligned} \tag{11.1}$$

where

$$\phi(b_{i,s}) = \begin{cases} 1, & \text{if } b_{i,s} = 1, \\ 2, & \text{otherwise} \end{cases} \tag{11.2}$$

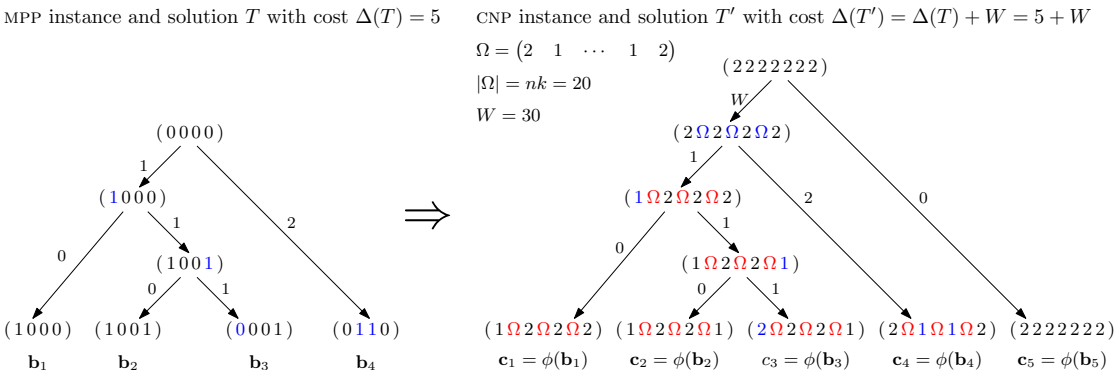

Figure 11.1: **Transformation of an MPP instance to a CNT instance.** Left shows an MPP instance and solution T, whereas right shows the corresponding CNT instance and solution T'. Edges are labeled by the cost of the associated events and their affected segments are colored in blue.

and Ω, called a *wall*, is a vector of size nk such that for each $j \in \{1, \ldots, nk\}$

$$\Omega_j = \begin{cases} 2, & \text{if } j \text{ is odd,} \\ 1, & \text{otherwise.} \end{cases} \tag{11.3}$$

Informally, $\mathbf{c}_i$ is defined as a vector consisting of *true segments* (which correspond to the original values) that are separated by *walls* (which are vectors Ω of alternating $2, 1$ values of length nk). The purpose of wall segments Ω is to prevent an event from spanning more than one true segment. Profile $\mathbf{c}_{k+1}$ consists of only 2's, and plays a role in initializing the wall elements Ω immediately from the all 2's root. This transformation can be computed in polynomial time, and it is used in the following proof of hardness.

Theorem 11.1 *The CNT problem is NP-hard.*

Proof We claim that MPP instance, composed of $\mathbf{b}_1, \ldots, \mathbf{b}_k$ such that $|\mathbf{b}_i| = n$, admits a binary phylogeny T with cost at most h if and only if the corresponding CNT instance, composed of $\mathbf{c}_1, \ldots, \mathbf{c}_{k+1}$ and $e = 2$ such that $|\mathbf{c}_i| = n$, admits a copy-number tree T' with cost at most $h + W$ where $W = (n-1)nk/2$. Note that $(n-1)nk$ is even, and thus $W \in \mathbb{N}$. Intuitively, W represents the cost of 'initializing' the wall elements Ω.

($\Rightarrow$) Let T be a binary phylogeny with cost $\Delta(T) \leq h$. We denote by $\mathbf{b}_i$ the binary vector of vertex $v_i \in V(T)$. For each true segment $s \in [n]$, the corresponding segment in the transformation is denoted by $\alpha(s)$. We show that given T we can construct a copy-number tree T' such that $\Delta(T') = \Delta(T) + W$. Tree T' is composed of a root vertex $r(T')$ whose two children correspond to tree T (rooted at $r(T)$) and an additional leaf

w labeled by $\mathbf{c}_{k+1}$. The remaining vertices $v \in V(T') \setminus \{w\}$ are labeled by $\mathbf{c}_i = \phi(\mathbf{b}_i)$ (see Eq. (11.1)). The edge $(r(T'), w)$ of T' relates two vertices with the same profile and thus has cost 0. The other edge $(r(T'), r(T))$ has cost W, which corresponds to the number of wall segments that need to be initialized to 1 (these are common to all leaves $\mathbf{c}_1, \ldots, \mathbf{c}_k$). Let's consider an edge (v_i, v_j) of T with Hamming distance ζ. First, observe that the Hamming distance equals the number of flips required to transform $\mathbf{b}_i$ into $\mathbf{b}_j$. We describe how to obtain a sequence of events $\sigma(i,j)$ on the corresponding edge (v_i, v_j) in T' such that $\delta(i,j) = \zeta$. Consider segment $s \in [n]$. A flip from 0 to 1 at segment s corresponds to a deletion event $(\alpha(s), \alpha(s), -1)$. Conversely, a flip from 1 to 0 in segment s corresponds to an amplification event $(\alpha(s), \alpha(s), +1)$. Recall that $\delta(i,j) = \sum_{(s,t,b)\in\sigma(v_i,v_j)} |b|$. It thus follows that $\Delta(T') = \Delta(T) + W$. Since $\Delta(T) \leq h$, we thus have $\Delta(T') \leq h + W$.

($\Leftarrow$) Let T' be a copy-number tree with cost $\Delta(T') \leq h + W$. We denote by $\mathbf{c}_i$ the profile of vertex $v_i \in V(T')$. We show that T' can be transformed into a binary phylogeny T such that $\Delta(T) \leq h$. We distinguish two cases $h \geq nk + 1$ and $h \leq nk$.

1. If $h \geq nk + 1$, we can construct a naive binary phylogeny T whose internal vertices are labeled with the same binary vector as the root (all 0s). The cost of T is at most kn, and thus $\Delta(T) \leq nk + 1 \leq h$.

2. Now let's consider the case where $h \leq nk$. We assume without loss of generality that $n \geq 4$. Now, $h < W$ since $nk < W$ for $n \geq 4$. Hence, $\Delta(T') < 2W$. Recall that the root vertex $r(T')$ has 2's at every segment including the walls. We claim that $r(T')$ has two children one of which is a leaf labeled by $\mathbf{c}_{k+1}$. Assume for a contradiction that this is not the case and that the two children split $L(T')$ into two sets L_1 and L_2 such that $|L_1| > 1$ and $|L_2| > 1$. Thus there exist distinct leaves $v_1 \in L_1$ and $v_2 \in L_2$ such that for the respective profiles it holds that $\mathbf{y}_1 \neq \mathbf{c}_{k+1}$ and $\mathbf{y}_2 \neq \mathbf{c}_{k+1}$. Now the cost to initialize the wall elements of $\mathbf{y}_1$ and $\mathbf{y}_2$ is at least $2W$, which yields a contradiction. It thus follows that tree T' must be composed of a root vertex $r(T')$ whose first child corresponds to tree T'' (rooted at $r(T'')$) and second child is a leaf w labeled by $\mathbf{c}_{k+1}$. We focus our attention on T''.

   We claim that there is no event in T'' that covers more than one true segment. Assume for a contradiction that such an event (s,t,b) exists. By construction, segments s and t span at least one wall Ω. In our restricted setting where $e = 2$ and where the leaves of T'' do not contain 0s, the event (s,t,b) can only be applied if all segments from s to t have the same value. As such, this event must be preceded by at least nk other events (which is the length of a wall Ω). These events may be on the same edge or any ancestral edge. Therefore, $\Delta(T'') \geq nk + 1$, which is a contradiction. Hence, events in T'' where $\Delta(T'') \leq nk$ span at most one true segment.

Finally, we show how to construct a binary phylogeny T from T'' such that $\Delta(T) \leq h \leq nk$. T has the same topology of T''. Moreover, each vertex $v_i \in V(T)$ is labeled by a binary vector $\mathbf{b}_i$ such that $\mathbf{c}_i = \phi(\mathbf{b}_i)$. Let's consider an edge (v_i, v_j) of T'' labeled by events $\sigma(i,j)$ and with cost $\delta(i,j) = \zeta$. Each event $(s,t,b) \in \sigma(i,j)$ spans at most one true segment (but may contain parts of a wall Ω). Let $X \subseteq [n]$ be the set of true segments spanned by events in $\sigma(i,j)$. Observe that $|X| \leq \zeta$. Therefore, the Hamming distance between $\mathbf{b}_i$ and $\mathbf{b}_j$ is at most $|X|$. Hence, $\Delta(T) \leq \Delta(T'') \leq h$.

□

Chapter 12

Methods

Some of the results in this chapter are published in:

M. El-Kebir, B. J. Raphael, R. Shamir, R. Sharan, S. Zaccaria, M. Zehavi, and R. Zeira. Complexity and algorithms for copy-number evolution problems. *Algorithms for Molecular Biology*, 2016, in press.

S. Zaccaria[†], M. El-Kebir[†], G. Klau, and B. J. Raphael. The copy-number tree mixture deconvolution problem and applications to multi-sample bulk sequencing tumor data. In *Research in Computational Molecular Biology (RECOMB)*, 2017, in press.

[†]joint first authorship

This chapter describes the methods that we design for dealing with the CNT and CNTMD problems introduced in Chapter 10. Firstly, in Section 12.1 we present a ILP formulation for solving CNT. Next, in Section 12.2 we present a coordinate-descent algorithm for solving a distance-based variant of CNTMD. This algorithm includes an extended variant of the previous ILP formulation and comprises several steps whose details are described in the same Section 12.2. The implementations of all the methods presented in this section are currently available upon request.

12.1 ILP Formulation for CNT

In this section we describe an ILP for CNT consisting of $O(k^2 n + kn \log e)$ variables and $O(k^2 n + kn \log e)$ constraints. Let $(\mathbf{c}_1, \ldots, \mathbf{c}_k, e)$ be an instance of CNT. Recall that we seek to find a full binary tree with k leaves. We define a directed graph G that contains any full binary tree with k leaves as a spanning tree. As such, $|V(G)| = 2k - 1$. The vertex set $V(G)$ consists of a subset $L(G)$ of leaves such that $|L(G)| = k$. We denote by $r(T) \in V(G) \setminus L(G)$ the vertex that corresponds to the root vertex. Throughout the following, we consider an order $v_1, \ldots, v_k, \ldots, v_{2k-1}$ of the

vertices in $V(G)$ such that $v_1 = r(T)$ and $\{v_k, \dots, v_{2k-1}\} = L(G)$. The edge set $E(G)$ has edges $\{(v_i, v_j) \mid 1 \leq i < k, 1 \leq i < j \leq 2k-1\}$. We denote by $N^-(j)$ the set of vertices incident to an outgoing edge to j. Conversely, $N^+(i)$ denotes the set of vertices incident to an incoming edge from i. We make the following two observations.

Observation 12.1 *G is a directed acyclic graph.*

Observation 12.2 *Any copy-number tree T is a spanning tree of G.*

We now proceed to define the set of feasible solutions (X, Y) to a CNT instance $(\mathbf{c}_1, \dots, \mathbf{c}_k, e)$ by introducing constraints and variables modeling the tree topology, and vertex labeling and edge costs. More specifically, variables $X = [x_{i,j}]$ encode a spanning tree T of G and variables $Y = [y_{i,s}]$ encode the profiles of each vertex such that X and Y combined induce edge costs. In the following we provide more details.

Tree Topology. The goal is to enforce that we select a spanning tree T of G that is a full binary tree. To do so, we introduce a binary variable $x_{i,j} \in \{0, 1\}$ for each edge $(v_i, v_j) \in E(G)$ indicating whether the corresponding edge (v_i, v_j) is in T. Note that by construction $i < j$. We require that each vertex $v \in V(G) \setminus \{v_1\}$ has exactly one incoming edge in T.

$$\sum_{i \in N^-(j)} x_{i,j} = 1 \qquad 1 < j \leq 2k-1 \tag{12.1}$$

We require that each vertex $v \in V(G) \setminus L(G)$ has two outgoing edges in T.

$$\sum_{j \in N^+(i)} x_{i,j} = 2 \qquad 1 \leq i < k \tag{12.2}$$

Vertex Labeling and Edge Costs. We introduce variables $y_{i,s} \in \{0, \dots, e\}$ that encode the copy-number state of segment s of vertex v_i. Since the profiles of each leaf as well as the root vertex are given, we have the following constraints.

$$y_{1,s} = 2 \tag{12.3}$$

$$y_{i,s} = c_{i-k+1,s} \tag{12.4}$$

for each $i \in [k, 2k-1]$ and each $s \in [1, n]$.

Next, we encode a set $\sigma(v_i, v_j)$ of events that transform the profile $\mathbf{y}_i$ of v_i into profile $\mathbf{y}_j$ of v_j. Recall that an event is a triple (s, t, b) and corresponds to an amplification if $b > 0$ and a deletion otherwise. We model the cost of the amplifications and the cost of the deletions covering any segment s with two separate variables.

Table 12.1: Case analysis on the values of variables $y_{i,s}$ and $y_{j,s}$

	$a_{i,j,s}$	$d_{i,j,s}$	additional
(a) $y_{i,s} = 0 \wedge y_{j,s} = 0$	$\leq e$	$\leq e$	
(b) $y_{i,s} \neq 0 \wedge y_{j,s} \neq 0$	$\leq e$	$< y_{i,s}$	$y_{j,s} + d_{i,j,s} = y_{i,s} + a_{i,j,s}$
(c) $y_{i,s} \neq 0 \wedge y_{j,s} = 0$	$\leq e$	$\geq y_{i,s}$ $\leq e$	
(d) $y_{i,s} = 0 \wedge y_{j,s} \neq 0$	*infeasible*	*infeasible*	*infeasible*

Variables $a_{i,j,s} \in \{0,\ldots,e\}$ correspond to the cost of the amplifications in $\sigma(v_i, v_j)$ covering segment s. Variables $d_{i,j,s} \in \{0,\ldots,e\}$ correspond to the cost of the deletions in $\sigma(v_i, v_j)$ covering segment s.

Now, we consider the effect of amplifications and deletions on a segment s. Following a result from [139], we have that there exists an optimal solution such that for each edge (v_i, v_j) there are two sets of events $\sigma^-(v_i, v_j)$ and $\sigma^+(v_i, v_j)$ that yield $y_{j,s}$ from $y_{i,s}$ by first applying $\sigma^-(v_i, v_j)$ followed by $\sigma^+(v_i, v_j)$. If a subset of the events in $\sigma^-(v_i, v_j)$ results in segment s reaching value 0, the remaining amplifications and deletions will not change the value of that segment. We distinguish the following four different cases (Table 12.1).

(a) $y_{i,s} = 0$ and $y_{j,s} = 0$: Since both segments have value 0, the number of amplifications $a_{i,j,s}$ and deletions $d_{i,j,s}$ are between 0 and e.

(b) $y_{i,s} \neq 0$ and $y_{j,s} \neq 0$: Since $y_{j,s} > 0$, the number of deletions $d_{i,j,s}$ must be strictly smaller than $y_{i,s}$. Moreover, it must hold that $y_{j,s} + d_{i,j,s} = y_{i,s} + a_{i,j,s}$.

(c) $y_{i,s} \neq 0$ and $y_{j,s} = 0$: Recall that following a result in [139], deletions precede amplifications. As such, the number of deletions $d_{i,j,s}$ must be at least $y_{i,s}$.

(d) $y_{i,s} = 0$ and $y_{j,s} \neq 0$: Once a segment s has been lost it cannot be regained. As such, this case is infeasible.

To capture the conditions of the four cases, we introduce binary variables $\bar{y}_{i,s} \in \{0,1\}$ and constraints such that $\bar{y}_{i,s} = 1$ iff $y_{i,s} \neq 0$.

$$y_{i,s} = \sum_{q=0}^{\lfloor \log_2(e) \rfloor + 1} 2^q \cdot z_{i,s,q} \tag{12.5}$$

$$\bar{y}_{i,s} \leq \sum_{q=0}^{\lfloor \log_2(e) \rfloor + 1} z_{i,s,q} \tag{12.6}$$

$$\bar{y}_{i,s} \geq z_{i,s,q} \tag{12.7}$$
$$z_{i,s,q} \in \{0,1\} \tag{12.8}$$

for each $i \in [1, 2k-1]$, each $s \in [1,n]$, and each $q \in [0, \lfloor \log_2(e) \rfloor + 1]$. Since $a_{i,j,s}, d_{i,j,s} \in \{0,\ldots,e\}$, the upper bound constraints involving e are covered. In particular, case (a) is captured in its entirety. We capture case (b) with the following constraints.

$$y_{j,s} \leq y_{i,s} - d_{i,j,s} + a_{i,j,s} + 2e(2 - \bar{y}_{i,s} - \bar{y}_{j,s}) \tag{12.9}$$
$$y_{j,s} + 2e(2 - \bar{y}_{i,s} - \bar{y}_{j,s}) \geq y_{i,s} - d_{i,j,s} + a_{i,j,s} \tag{12.10}$$
$$d_{i,j,s} \leq y_{i,s} - 1 + (e+1)(2 - \bar{y}_{i,s} - \bar{y}_{j,s}) \tag{12.11}$$

for each segment $s \in [1,n]$ and each edge $(v_i, v_j) \in E(G)$. In the case of $\bar{y}_{i,s} = 1$ and $\bar{y}_{j,s} = 1$, constraints (12.9) and (12.10) model the equation $y_{j,s} + d_{i,j,s} = y_{i,s} + a_{i,j,s}$, whereas constraints (12.11) ensure that $d_{i,j,s} < y_{i,s}$. Next, we model case (c) using the following constraints.

$$y_{i,s} \leq d_{i,j,s} + e(1 - \bar{y}_{i,s} + \bar{y}_{j,s}) \tag{12.12}$$

for each segment $s \in [1,n]$ and each edge $(v_i, v_j) \in E(G)$. Finally, the following constraints, which encode that if $x_{i,j} = 1$ then $\bar{y}_{i,s} = 0$ implies $\bar{y}_{j,s} = 0$, prevent case (d) from happening.

$$(1 - x_{i,j}) + \bar{y}_{i,s} \geq \bar{y}_{j,s} \tag{12.13}$$

for each segment $s \in [1,n]$ and each edge $(v_i, v_j) \in E(G)$.

The cost of a tree T is the sum of the costs of the events associated to each edge $(v_i, v_j) \in E(T)$. We model the cost of an edge (v_i, v_j) as the sum of the number of amplifications and deletions that start at each segment s. Variables $\bar{a}_{i,j,s} \in \{0,\ldots,e\}$ and $\bar{d}_{i,j,s} \in \{0,\ldots,e\}$ represent the number of new amplifications and deletions, respectively, that start at segment s. We model this using the following constraints.

$$\bar{a}_{i,j,s} \geq a_{i,j,s} - a_{i,j,s-1} \tag{12.14}$$
$$\bar{d}_{i,j,s} \geq d_{i,j,s} - d_{i,j,s-1} \tag{12.15}$$
$$a_{i,j,0} = 0 \tag{12.16}$$
$$d_{i,j,0} = 0 \tag{12.17}$$

for each segment $s \in [1,n]$ and each edge $(v_i, v_j) \in E(G)$.

The objective is to minimize the cost of the events of the selected tree T, which corresponds to

$$\min \sum_{(v_i,v_j) \in E(G)} \sum_{1 \leq s \leq n} x_{i,j} \cdot (\bar{a}_{i,j,s} + \bar{d}_{i,j,s}) \tag{12.18}$$

We model the product using the following constraint.

$$w_{i,j,s} \geq \bar{a}_{i,j,s} + \bar{d}_{i,j,s} - (1 - x_{i,j}) \cdot 2e \tag{12.19}$$

for each segment $s \in [1,n]$ and each edge $(v_i, v_j) \in E(G)$.

In the following Section 12.1.1, we report the complete ILP formulation.

12.1.1 Copy-Number Tree Problem: Complete ILP

The ILP formulation is reproduced in its entirety below. We define $M = \lfloor \log_2(e) \rfloor + 1$.

$$\min \sum_{(v_i,v_j)\in E(G)} \sum_{1\leq s\leq n} w_{i,j,s}$$

$$\sum_{i\in N^-(j)} x_{i,j} = 1 \qquad 1 < j \leq 2k-1$$

$$\sum_{j\in N^+(i)} x_{i,j} = 2 \qquad 1 \leq i < k$$

$$y_{1,s} = 2 \qquad 1 \leq s \leq n$$

$$y_{i,s} = c_{i-k+1,s} \qquad k \leq i \leq 2k-1, 1 \leq s \leq n$$

$$y_{i,s} = \sum_{q=0}^{M} 2^q \cdot z_{i,s,q} \qquad 1 \leq i \leq 2k-1, 1 \leq s \leq n$$

$$\bar{y}_{i,s} \leq \sum_{q=0}^{M} z_{i,s,q} \qquad 1 \leq i \leq 2k-1, 1 \leq s \leq n$$

$$\bar{y}_{i,s} \geq z_{i,s,q} \qquad 1 \leq i \leq 2k-1, 1 \leq s \leq n, 0 \leq q \leq M$$

$$y_{j,s} \leq y_{i,s} - d_{i,j,s} + a_{i,j,s} + 2e(2 - \bar{y}_{i,s} - \bar{y}_{j,s}) \qquad 1 \leq s \leq n, (v_i,v_j) \in E(G)$$

$$y_{j,s} + 2e(2 - \bar{y}_{i,s} - \bar{y}_{j,s}) \geq y_{i,s} - d_{i,j,s} + a_{i,j,s} \qquad 1 \leq s \leq n, (v_i,v_j) \in E(G)$$

$$d_{i,j,s} \leq y_{i,s} - 1 + (e+1)(2 - \bar{y}_{i,s} - \bar{y}_{j,s}) \qquad 1 \leq s \leq n, (v_i,v_j) \in E(G)$$

$$y_{i,s} \leq d_{i,j,s} + e(1 - \bar{y}_{i,s} + \bar{y}_{j,s}) \qquad 1 \leq s \leq n, (v_i,v_j) \in E(G)$$

$$(1 - x_{i,j}) + \bar{y}_{i,s} \geq \bar{y}_{j,s} \qquad 1 \leq s \leq n, (v_i,v_j) \in E(G)$$

$$\bar{a}_{i,j,s} \geq a_{i,j,s} - a_{i,j,s-1} \qquad 1 \leq s \leq n, (v_i,v_j) \in E(G)$$

$$\bar{d}_{i,j,s} \geq d_{i,j,s} - d_{i,j,s-1} \qquad 1 \leq s \leq n, (v_i,v_j) \in E(G)$$

$$a_{i,j,0} = 0 \qquad (v_i,v_j) \in E(G)$$

$$d_{i,j,0} = 0 \qquad (v_i,v_j) \in E(G)$$

$$w_{i,j,s} \geq \bar{a}_{i,j,s} + \bar{d}_{i,j,s} - (1 - x_{i,j}) \cdot 2e \qquad 1 \leq s \leq n, (v_i,v_j) \in E(G)$$

$$x_{i,j} \in \{0,1\} \qquad (v_i,v_j) \in E(G)$$

$$y_{i,s} \in \{0,\ldots,e\} \qquad 1 \leq i \leq 2k-1, 1 \leq s \leq n$$

$$\bar{y}_{i,s} \in \{0,1\} \qquad 1 \leq i \leq 2k-1, 1 \leq s \leq n$$

$$z_{i,s,q} \in \{0,1\} \qquad 1 \leq i \leq 2k-1, 1 \leq s \leq n, 0 \leq q \leq M$$

$$a_{i,j,s}, d_{i,j,s} \in \{0,\dots,e\} \qquad 1 \le s \le n, (v_i, v_j) \in E(G)$$
$$\bar{a}_{i,j,s}, \bar{d}_{i,j,s} \in \{0,\dots,e\} \qquad 1 \le s \le n, (v_i, v_j) \in E(G)$$
$$w_{i,j,s} \in \{0,\dots,2e\} \qquad 1 \le s \le n, (v_i, v_j) \in E(G)$$

12.2 Coordinate-descent Algorithm for CNTMD

We suspect that CNTMD is computationally hard, as the related unmixed version, the CNT problem, is NP-hard (Chapter 11). In fact, we need to solve an instance of CNT in CNTMD for computing the cost of the tree. Moreover, other similar deconvolution problems under a tree constraint are NP-hard as well [44, 46]. As such, we design an algorithm inspired by the coordinate-descent paradigm for solving a distance-based version of CNTMD where we aim to infer copy-numbers C with n clones (columns) and mixing proportions U that minimize the distance between the *observed* and *inferred* fractional copy numbers:

$$\|F - CU\| = \sum_{1 \le s \le m} \sum_{1 \le p \le k} \left| f_{s,p} - \sum_{1 \le i \le n} c_{s,i} u_{i,p} \right|. \tag{12.20}$$

Under a parsimony constraint, we impose a maximum cost $\Lambda_{\max}$ on the copy-number tree T generated by C. That is, we require that C generates T such that $\Lambda(T) \le \Lambda_{\max}$ and we consider the following problem.

Problem 12.1 (distance-based CNTMD (d-CNTMD)) *Given an $m \times k$ fractional copy-number matrix F, a number n of clones, a maximum copy-number c_{max}, and a maximum cost Λ_{max}, find (1) an $m \times n$ copy-number matrix C generating T and (2) an $n \times k$ usage matrix U such that $\Lambda(T) \le \Lambda_{max}$ and $\|F - CU\|$ is minimum.*

First, in Section 12.2.1 we describe a coordinate-descent algorithm for solving the above problem for a fixed maximum cost $\Lambda_{\max}$. Next, in Section 12.2.2 we present a scheme that searches for the maximum cost Λ^* that achieves a good trade-off between the distance $\|F - CU\|$ and the cost of the resulting tree T. Lastly, Section 12.2.3 reports several details about the implementation of the algorithm and its input parameters, as well as some consideration about features that could be integrated in the algorithm for exploiting additional information on the composition of the input samples.

12.2.1 Coordinate-descent Algorithm With a Fixed $\Lambda_{\max}$

Following the coordinate-descent paradigm, we split the variables of d-CNTMD and obtain two subproblems with the same objective of minimizing the distance

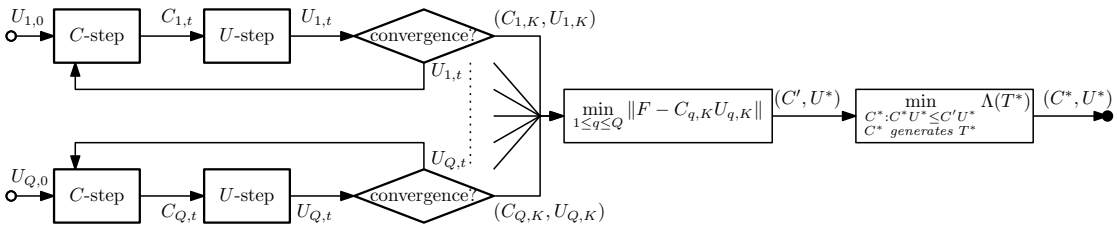

Figure 12.1: **Coordinate-descent algorithm.** A scheme describing the steps of the proposed algorithm.

$\|F - CU\|$. In these subproblems either matrix C or matrix U is fixed. An iteration t consists of two steps. In the C-step, we are given a usage matrix U_{t-1} and we search for a copy-number matrix C_t minimizing $\|F - C_t U_{t-1}\|$ such that C generates T with cost $\Lambda(T) \le \Lambda_{\max}$. Conversely, in the U-step we take the matrix C_t as input and seek a usage matrix U_t such that $\|F - C_t U_t\|$ is minimized.

To account for different local optima, we use Q different initial usage matrix $U_{0,0}, \ldots, U_{Q,0}$. We generate these usage matrices in a sparse way using a method based on random-number partitions (Section 12.2.1.4). It is easy to see that this scheme produces a sequence of matrices $(C_{q,t}, U_{q,t}), (C_{q,t+1}, U_{q,t+1}), \ldots$ such that $\|F - C_{q,t} U_{q,t}\| \ge \|F - C_{q,t+1} U_{q,t+1}\|$ as both $C_{q,t+1}$ and $U_{q,t+1}$ can be chosen equal to the previous matrices $C_{q,t}$ and $U_{q,t}$ resulting in the same distance $\|F - C_{q,t} U_{q,t}\|$. We iterate until $\|F - C_{q,t} U_{q,t}\|$ drops below a convergence threshold or the number of iterations reaches a specified number K.

Our algorithm computes $(C_{q,K}, U_{q,K})$ for each starting point $U_{q,0}$ and selects the matrices (C', U^*) that minimize the distance $\|F - C_{q,K} U_{q,K}\|$. However, multiple copy-number trees T' with $\Lambda(T') \le \Lambda_{\max}$ can be generated by C' as well as by a different copy-number matrix C'' with $\|F - C'' U^*\| = \|F - C' U^*\|$. Thus, to find T^* of minimum cost $\Lambda(T^*) \le \Lambda_{\max}$ generated by a copy-number matrix C^* with $\|F - C^* U^*\| = \|F - C' U^*\|$, we introduce a *refinement step* (Section 12.2.1.5). Figure 12.1 depicts the scheme of the coordinate-descent algorithm.

We present a LP formulation for the U-step in Section 12.2.1.1. Next, we present the main aspects of the ILP formulation for the C-step in Section 12.2.1.2 by highlighting the differences with the ILP for CNT previously designed in this chapter. A detailed description of this ILP for the C-step is subsequently presented in Section 12.2.1.3. Lastly, the procedure for generating the starting points is presented in Section 12.2.1.4, whereas the details of the refinement step are reported in Section 12.2.1.5.

12.2.1.1 *U-step* In the U-step, we are given fractional matrix F and copy-number matrix C, and seek a usage matrix U with real-valued entries mini-

mizing the distance $\|F - CU\|$. We linearize the distance function $\|F - CU\|$ and formulate the resulting the optimization problem as a linear program (LP) with $O(km)$ variables and $O(km)$ constraints. We model the absolute difference $|f_{s,p} - \sum_{1 \le i \le n} c_{s,i} u_{i,p}|$ by variables $\bar{f}_{s,p}$ for each sample p and segment s. We require that $\bar{f}_{s,p} \ge f_{s,p} - \sum_{1 \le i \le n} c_{s,i} u_{i,p}$ and $\bar{f}_{s,p} \ge \sum_{1 \le i \le n} c_{s,i} u_{i,p} - f_{s,p}$. Thus, the objective is $\sum_{1 \le s \le m, 1 \le p \le k} \bar{f}_{s,p}$. Moreover, we introduce variables $0 \le u_{i,p} \le 1$ that represent the usage of a clone i in sample p. We constraint the usages of each sample to sum to 1, i.e. $\sum_{1 \le i \le n} u_{i,p} = 1$ for each sample p. The full LP for the U-step is reproduced in its entirety below.

$$
\begin{aligned}
\min \sum_{1 \le p \le m} \sum_{1 \le s \le n} & \bar{f}_{p,s} \\
\bar{f}_{s,p} \ge f_{s,p} - & \sum_{1 \le i \le 2n-1} c_{s,i} x_{i,p} && 1 \le p \le k, 1 \le s \le m \\
\bar{f}_{s,p} \ge & \sum_{1 \le i \le 2n-1} c_{s,i} x_{i,p} - f_{s,p} && 1 \le p \le k, 1 \le s \le m \\
& \sum_{1 \le i \le n} x_{i,p} = 1 && 1 \le p \le k
\end{aligned}
$$

12.2.1.2 C-step In the C-step, we are given a fractional matrix F and a usage matrix U, and seek a copy-number matrix C with integer entries minimizing the distance $\|F - CU\|$. Similarly, to the U-step we linearize the distance function $\|F - CU\|$ by modeling the absolute different with variables $\bar{f}_{s,p}$ and their corresponding constraints. Since the variables $c_{s,i}$ that model matrix C are integer, we formulate the problem as an *integer* linear program (ILP) with $O(n^2 m + nm \log \Lambda_{\max} + km)$ variables and constraints. Our formulation introduces new constraints that improve upon the model introduced in Section 12.1. In the following we describe the main variables and constraints of the ILP. We provide a detailed description of the ILP formulation in Section 12.2.1.3.

More specifically, we are given $(F, U, c_{\max}, \Lambda_{\max})$ and seek a copy-number matrix C that minimizes $\|F - CU\|$ such that C generates a tree T whose cost $\Lambda(T)$ is at most $\Lambda_{\max}$. We introduce binary variables $X = [x_{i,j}]$ and integer variables $\tilde{C} = [c_{s,i}]$ to model the topology of T and to label its vertices and edges, respectively. From $\tilde{C}$, which encodes the copy-number profiles of all the vertices of T, the copy-number matrix C can be obtained, which encodes the copy-number profiles of the leaves of T.

Topology of T. Recall that T is a full binary tree (Definition 10.1). We build a directed acyclic graph $G = (V, E)$ containing all copy-number trees T with n leaves as spanning trees. A variable $x_{i,j}$ is introduced for each edge $(v_i, v_j) \in E$ indicating whether (v_i, v_j) is an edge of T. To encode that T is a full binary spanning tree of G,

we require that each non-root vertex has exactly one incoming edge and that each internal vertex has two outgoing edges.

Vertex and edge labeling. We introduce integer variables $c_{s,i} \in \{0, c_{\max}\}$ to encode the profiles of each vertex v_i. More precisely, $c_{s,i}$ encodes the copy-number state of segment s of vertex v_i. Since the profile of the root vertex is diploid, we set $c_{s,r(T)} = 2$ of every segment s of the root $r(T)$. From these profiles and the topology of T as captured by variables $x_{i,j}$, we obtain the events $\mathscr{E}(i,j)$ that transform the profile $\mathbf{c}_i$ into the profile $\mathbf{c}_j$ and their cost for any edge (v_i, v_j). We model the amplifications and deletions covering any segment s in $\mathscr{E}(i,j)$ with two separate variables $a_{s,i,j} \in \{0,\dots,c_{\max}\}$ and $d_{s,i,j} \in \{0,\dots,c_{\max}\}$, respectively. We model the cost of an edge (v_i, v_j) as the sum of the amplifications and deletions starting on each segment s by introducing variables $\bar{a}_{s,i,j} \in \{0,\dots,c_{\max}\}$ and $\bar{d}_{s,i,j} \in \{0,\dots,c_{\max}\}$, respectively. As such, $\bar{a}_{s,i,j}$ is equal to $\max\{a_{s,i,j} - a_{s-1,i,j}, 0\}$ and, symmetrically, $\bar{d}_{s,i,j}$ is equal to $\max\{d_{s,i,j} - d_{s-1,i,j}, 0\}$. We force $\bar{a}_{s,i,j}$ and $\bar{d}_{s,i,j}$ to be equal to 0 when the corresponding edge (v_i, v_j) is not in T, i.e. $x_{i,j} = 0$. Hence, the cost $\Lambda(T)$ is the sum of the costs of all the edges, and we require that this cost is at most $\Lambda_{\max}$. We can express the cost of the tree as the sum of the costs of all the edges thanks to the new constraints, compared to the previous model in Section 12.1 that force the edges to be empty and the corresponding variables to be equal to 0 when the related edge is not in T. Recall that, without loss of generality, deletions precede amplifications on each edge and that a segment s reaching 0 cannot change value anymore. As such, we distinguish four different cases on whether $c_{s,i}$ and $c_{s,j}$ are equal or different from 0. We provide additional details in Table 12.2 and in the following Section 12.2.1.3.

12.2.1.3 Detailed C-step In this section we report a detailed description of the ILP for the C-step described in the previous section. This ILP is an extended variant of the one that have been designed for CNT in Section 12.1. The ILP consists of $O(n^2m + nm\log\Lambda_{\max} + km)$ variables and $O(n^2m + nm\log\Lambda_{\max} + km)$ constraints. Let $(F, U, c_{\max}, \Lambda_{\max})$ be an instance of CNTMD with a fixed usage matrix U. We seek to find a collection C of n profiles generating a copy-number tree T^* with $\Lambda(T^*) \leq \Lambda_{\max}$. Recall that T^* is a full binary tree by definition. We define a directed graph G that contains any full binary tree with n leaves as a spanning tree. As such, $|V(G)| = 2n-1$. The vertex set $V(G)$ consists of a subset $L(G)$ of leaves such that $|L(G)| = n$. Recall that $r(T)$ is the root vertex and we are considering an ordering $v_1,\dots,v_k,\dots,v_{2n-1}$ of the vertices in $V(G)$ such that $v_{2n-1} = r(T)$ and $\{v_1,\dots,v_n\} = L(G)$. The edge set $E(G)$ has edges $\{(v_i, v_j) \mid n+1 \leq i \leq 2n-1, 1 \leq j < i\}$. We denote by $N^-(j)$ the set of vertices i with an outgoing edge to j. Conversely,

$N^+(i)$ denotes the set of vertices j with incoming edge from i. We make the following two observations.

Observation 12.3 *G is a directed acyclic graph.*

Observation 12.4 *Any copy-number tree T is a spanning tree of G.*

We now proceed to define the set of feasible solutions (X,Y) to the instance $(F,M,c_{\max},\Lambda_{\max})$ by introducing constraints and variables modeling the tree topology, vertex labeling and edge costs. More specifically, variables $X=[x_{i,j}]$ encode a spanning tree T of G and variables $Y=[y_{s,i}]$ encode the profiles of each vertex such that X and Y combined induce edge costs. Note that here Y corresponds to the previous $\tilde{C}$, but we use Y to have a symmetrical description to one in Section 12.1. In the following we provide more details.

Tree Topology. The goal is to select a spanning tree T of G that is a full binary tree. To do so, we introduce a binary variable $x_{i,j}\in\{0,1\}$ for each edge $(v_i,v_j)\in E(G)$ indicating whether the corresponding edge (v_i,v_j) is in T. Note that by construction $i<j$. We require that each vertex $v\in V(G)\setminus\{r(T)\}$ has exactly one incoming edge in T.

$$\sum_{i\in N^-(j)} x_{i,j}=1 \qquad\qquad 1\leq j<2n-1 \tag{12.21}$$

We require that each vertex $v\in V(G)\setminus L(G)$ has two outgoing edges in T.

$$\sum_{j\in N^+(i)} x_{i,j}=2 \qquad\qquad n<i\leq 2n-1 \tag{12.22}$$

Vertex Labeling and Edge Costs. We introduce variables $y_{s,i}\in\{0,\dots,c_{\max}\}$ that encode the copy-number state of segment s of vertex v_i. Since the profile of the root vertex is given, we have the following constraints.

$$y_{s,1}=2 \qquad\qquad 1\leq s\leq m \tag{12.23}$$

Next, we encode a set $\mathscr{E}(i,j)$ of events that transform the profile $\mathbf{y}_i$ of v_i into profile $\mathbf{y}_j$ of v_j. Recall that an event is a triple (s,t,b) and corresponds to an amplification if $b>0$ and a deletion otherwise. We model the cost of the amplifications and the cost of the deletions covering any segment s with two separate variables. Variables $a_{s,i,j}\in\{0,\dots,c_{\max}\}$ correspond to the cost of the amplifications in $\mathscr{E}(i,j)$ covering segment s. Variables $d_{s,i,j}\in\{0,\dots,c_{\max}\}$ correspond to the cost of the deletions in $\mathscr{E}(i,j)$ covering segment s. Notice that we ask $\mathscr{E}(i,j)$ to be empty when the corresponding edge (v_i,v_j) is not in T. As such, we design the constraints of

	$a_{s,i,j}$	$d_{s,i,j}$	additional
(a) $c_{s,i} = 0 \wedge c_{s,j} = 0$	$\leq c_{\max}$	$\leq c_{\max}$	
(b) $c_{s,i} \neq 0 \wedge c_{s,j} \neq 0$	$\leq c_{\max}$	$< c_{s,i}$	$c_{s,j}+d_{i,j,s} = c_{s,i}+a_{s,i,j}$
(c) $c_{s,i} \neq 0 \wedge c_{s,j} = 0$	$\leq c_{\max}$	$\leq c_{\max}, \geq c_{s,i}$	
(d) $c_{s,i} = 0 \wedge c_{s,j} \neq 0$	*infeasible*	*infeasible*	*infeasible*

Table 12.2: **Edge labeling.** Case analysis on the values of variables $c_{s,i}$ and $c_{s,j}$.

our model such that both $a_{s,i,j}$ and $d_{s,i,j}$ can be equal to zero whenever (v_i, v_j) is not in T. We force all the variables $a_{s,i,j}$ and $d_{s,i,j}$ to be 0 when (v_i, v_j) is not in T introducing the following constraints, that are useful for breaking ties.

$$a_{s,i,j} \leq c_{\max} x_{i,j} \qquad 1 \leq s \leq m, (v_i, v_j) \in E(G) \tag{12.24}$$

$$d_{s,i,j} \leq c_{\max} x_{i,j} \qquad 1 \leq s \leq m, (v_i, v_j) \in E(G) \tag{12.25}$$

Now, we consider the effect of amplifications and deletions on a segment s. As reported above, we assume that deletions are applied before amplifications. Moreover, if a subset of deletions results in segment s reaching value 0, the remaining amplifications and deletions will not change the value of that segment. We distinguish the following four different cases (Table 12.2).

(a) $y_{s,i} = 0$ and $y_{s,j} = 0$: Since both segments have value 0, the number of amplifications $a_{s,i,j}$ and deletions $d_{s,i,j}$ are between 0 and $\Lambda_{\max}$.

(b) $y_{s,i} \neq 0$ and $y_{s,j} \neq 0$: Since $y_{s,j} > 0$, the number of deletions $d_{s,i,j}$ must be strictly smaller than $y_{s,i}$. Moreover, it must hold that $y_{s,j} + d_{s,i,j} = y_{s,i} + a_{s,i,j}$.

(c) $y_{s,i} \neq 0$ and $y_{s,j} = 0$: Recall that following a result in [139] deletions precede amplifications. As such, the number of deletions $d_{s,i,j}$ must be at least $y_{s,i}$.

(d) $y_{s,i} = 0$ and $y_{s,j} \neq 0$: Once a segment s has been lost it cannot be regained. As such, this case is infeasible.

These cases are described in Table 12.2.

To capture the conditions of the four cases, we introduce binary variables $\bar{y}_{s,i} \in \{0,1\}$ and constraints such that $\bar{y}_{s,i} = 1$ iff $y_{s,i} \neq 0$.

$$y_{s,i} = \sum_{q=0}^{\lfloor \log_2(c_{\max}) \rfloor + 1} 2^q \cdot z_{i,s,q} \qquad 1 \leq i \leq 2n-1, 1 \leq s \leq m \quad (12.26)$$

$$\bar{y}_{s,i} \leq \sum_{q=0}^{\lfloor \log_2(c_{\max}) \rfloor + 1} z_{i,s,q} \qquad 1 \leq i \leq 2n-1, 1 \leq s \leq m \quad (12.27)$$

$$\bar{y}_{s,i} \geq z_{i,s,q} \qquad 1 \leq i \leq 2n-1, 1 \leq s \leq m, 0 \leq q \leq \lfloor \log_2(c_{\max}) \rfloor + 1 \quad (12.28)$$

$$z_{i,s,q} \in \{0,1\} \qquad 1 \leq i \leq 2n-1, 1 \leq s \leq m, 0 \leq q \leq \lfloor \log_2(c_{\max}) \rfloor + 1 \quad (12.29)$$

Since $a_{s,i,j}, d_{s,i,j} \in \{0, \ldots, c_{\max}\}$, the upper bound constraints involving $c_{\max}$ are covered. In particular, case (a) is captured in its entirety. We capture case (b) with the following constraints.

$$y_{s,j} \leq y_{s,i} - d_{s,i,j} + a_{s,i,j} + 2c_{\max}(3 - \bar{y}_{i,s} - \bar{y}_{j,s} - x_{i,j}) \qquad 1 \leq s \leq m, (v_i, v_j) \in E(G) \quad (12.30)$$

$$y_{s,j} + 2c_{\max}(3 - \bar{y}_{s,i} - \bar{y}_{s,j} - x_{i,j}) \geq y_{s,i} - d_{s,i,j} + a_{s,i,j} \qquad 1 \leq s \leq m, (v_i, v_j) \in E(G) \quad (12.31)$$

$$d_{i,j,s} \leq y_{s,i} - 1 + (c_{\max} + 1)(2 - \bar{y}_{s,i} - \bar{y}_{s,j}) \qquad 1 \leq s \leq m, (v_i, v_j) \in E(G) \quad (12.32)$$

In the case of $x_{i,j} = 1$ (i.e., (v_i, v_j) is in T), $\bar{y}_{s,i} = 1$, and $\bar{y}_{s,j} = 1$, constraints (12.30) and (12.31) model the equation $y_{s,j} + d_{s,i,j} = y_{s,i} + a_{s,i,j}$, whereas constraint (12.32) ensures that $d_{s,i,j} < y_{s,i}$. Notice that $d_{s,i,j}$ can be always equal to zero by constraint (12.32), hence we do not need to distinguish whether $x_{i,j} = 0$ or $x_{i,j} = 1$. Next, we model case (c), when $x_{i,j} = 1$, using the following constraints.

$$y_{s,i} \leq d_{s,i,j} + c_{\max}(2 - \bar{y}_{s,i} + \bar{y}_{s,j} - x_{i,j}) \qquad 1 \leq s \leq m, (v_i, v_j) \in E(G) \quad (12.33)$$

Finally, the following constraints, which encode that if $x_{i,j} = 1$ then $\bar{y}_{s,i} = 0$ implies $\bar{y}_{s,j} = 0$, prevent case (d) from happening.

$$1 - x_{i,j} + \bar{y}_{s,i} \geq \bar{y}_{s,j} \qquad 1 \leq s \leq m, (v_i, v_j) \in E(G) \quad (12.34)$$

The cost of a tree T is the sum of the costs of the events associated to each edge $(v_i, v_j) \in E(T)$. We model the cost of an edge (v_i, v_j) as the sum of the number of amplifications and deletions that start at each segment s. Variables $\bar{a}_{s,i,j} \in \{0, \ldots, c_{\max}\}$ and $\bar{d}_{s,i,j} \in \{0, \ldots, c_{\max}\}$ represent the number of new amplifications and deletions,

respectively, that start at segment s. We model this using the following constraints.

$$\bar{a}_{s,i,j} \geq a_{s,i,j} - a_{s-1,i,j} \qquad 1 \leq s \leq m, (v_i, v_j) \in E(G) \tag{12.35}$$
$$\bar{d}_{s,i,j} \geq d_{s,i,j} - d_{s-1,i,j} \qquad 1 \leq s \leq m, (v_i, v_j) \in E(G) \tag{12.36}$$
$$a_{0,i,j} = 0 \qquad (v_i, v_j) \in E(G) \tag{12.37}$$
$$d_{0,i,j} = 0 \qquad (v_i, v_j) \in E(G) \tag{12.38}$$

As before, we force $a_{s,i,j}$ and $d_{s,i,j}$ to be equal to 0 when the corresponding edge (v_i, v_j) is not in T in order to break the ties:

$$\bar{a}_{s,i,j} \leq c_{\max} x_{i,j} \qquad 1 \leq s \leq m, (v_i, v_j) \in E(G) \tag{12.39}$$
$$\bar{d}_{s,i,j} \leq c_{\max} x_{i,j} \qquad 1 \leq s \leq m, (v_i, v_j) \in E(G) \tag{12.40}$$

As such, the cost $\Lambda(T)$ of T is modeled as the sum of the costs overall the edges and the upper bound defined by $\Lambda_{\max}$ is encoded as follows

$$\sum_{(v_i,v_j) \in E(G)} \sum_{1 \leq s \leq m} \bar{a}_{s,i,j} + \bar{d}_{s,i,j} \leq \Lambda_{\max} \tag{12.41}$$

The objective is to minimize the sum of the distance $|F - CU|$ between observation and estimation. We model distance $|F - CU|$ introducing variables $\bar{f}_{s,p}$ for each sample p and each segment s, and using the following constraints

$$\bar{f}_{s,p} \geq f_{s,p} - \sum_{1 \leq i \leq n} y_{s,i} u_{p,i-k+1} \qquad 1 \leq p \leq k, 1 \leq s \leq m \tag{12.42}$$
$$\bar{f}_{s,p} \geq \sum_{1 \leq i \leq n} y_{s,i} u_{p,i-k+1} - f_{p,s} \qquad 1 \leq p \leq k, 1 \leq s \leq m \tag{12.43}$$

As a result, the objective function is encoded as follows

$$\min \sum_{1 \leq s \leq m} \sum_{1 \leq p \leq k} \bar{f}_{s,p} \tag{12.44}$$

Complete MILP for solving C-step We define $M = \lfloor \log_2(c_{\max}) \rfloor + 1$. The ILP formulation is reproduced in its entirety below.

$$\min \sum_{1\le s\le m}\sum_{1\le p\le k} \bar{f}_{s,p}$$

$$\sum_{(v_i,v_j)\in E(G)}\sum_{1\le s\le m} \bar{a}_{s,i,j} + \bar{d}_{s,i,j} \le \Lambda_{\max}$$

$$\bar{f}_{s,p} \ge f_{s,p} - \sum_{1\le i\le n} y_{s,i} u_{p,i-k+1} \qquad 1\le p\le k, 1\le s\le m$$

$$\bar{f}_{s,p} \ge \sum_{1\le i\le n} y_{i,s} u_{p,i-k+1} - f_{p,s} \qquad 1\le p\le k, 1\le s\le m$$

$$\sum_{i\in N^-(j)} x_{i,j} = 1 \qquad 1\le j<2n-1$$

$$\sum_{j\in N^+(i)} x_{i,j} = 2 \qquad n<i\le 2n-1$$

$$y_{s,1} = 2 \qquad 1\le s\le m$$

$$y_{s,i} = \sum_{q=0}^{\lfloor\log_2(c_{\max})\rfloor+1} 2^q \cdot z_{i,s,q} \qquad 1\le i\le 2k-1, 1\le s\le m$$

$$\bar{y}_{s,i} \le \sum_{q=0}^{\lfloor\log_2(c_{\max})\rfloor+1} z_{i,s,q} \qquad 1\le i\le 2k-1, 1\le s\le m$$

$$\bar{y}_{s,i} \ge z_{i,s,q} \qquad 1\le i\le 2k-1, 1\le s\le m, 0\le q\le \lfloor\log_2(c_{\max})\rfloor+1$$

$$y_{s,j} \le y_{s,i} - d_{s,i,j} + a_{s,i,j} + 2c_{\max}(3-\bar{y}_{i,s}-\bar{y}_{j,s}-x_{i,j}) \qquad 1\le s\le m, (v_i,v_j)\in E(G)$$

$$y_{s,j} + 2c_{\max}(3-\bar{y}_{s,i}-\bar{y}_{s,j}-x_{i,j}) \ge y_{s,i} - d_{s,i,j} + a_{s,i,j} \qquad 1\le s\le m, (v_i,v_j)\in E(G)$$

$$d_{i,j,s} \le y_{s,i} - 1 + (c_{\max}+1)(2-\bar{y}_{s,i}-\bar{y}_{s,j}) \qquad 1\le s\le m, (v_i,v_j)\in E(G)$$

$$y_{s,i} \le d_{s,i,j} + c_{\max}(2-\bar{y}_{s,i}+\bar{y}_{s,j}-x_{i,j}) \qquad 1\le s\le m, (v_i,v_j)\in E(G)$$

$$1 - x_{i,j} + \bar{y}_{s,i} \ge \bar{y}_{s,j} \qquad 1\le s\le m, (v_i,v_j)\in E(G)$$

$$\bar{a}_{s,i,j} \ge a_{s,i,j} - a_{s-1,i,j} \qquad 1\le s\le m, (v_i,v_j)\in E(G)$$

$$\bar{d}_{s,i,j} \ge d_{s,i,j} - d_{s-1,i,j} \qquad 1\le s\le m, (v_i,v_j)\in E(G)$$

$$a_{s,i,j} \le c_{\max} x_{i,j} \qquad 1\le s\le m, (v_i,v_j)\in E(G)$$

$$d_{s,i,j} \le c_{\max} x_{i,j} \qquad 1\le s\le m, (v_i,v_j)\in E(G)$$

$$\bar{a}_{s,i,j} \le c_{\max} x_{i,j} \qquad 1\le s\le m, (v_i,v_j)\in E(G)$$

$$\bar{d}_{s,i,j} \le c_{\max} x_{i,j} \qquad 1\le s\le m, (v_i,v_j)\in E(G)$$

$$a_{0,i,j} = 0 \qquad (v_i,v_j)\in E(G)$$

$$d_{0,i,j} = 0 \qquad (v_i,v_j)\in E(G)$$

$$x_{i,j} \in \{0,1\} \qquad (v_i,v_j)\in E(G)$$

$$y_{s,i} \in \{0,\dots,c_{\max}\} \qquad 1\le i\le 2n-1, 1\le s\le m$$

$$\bar{y}_{s,i} \in \{0,1\} \qquad 1\le i\le 2n-1, 1\le s\le m$$

$$z_{i,s,q} \in \{0,1\} \qquad 1\le i\le 2n-1, 1\le s\le m, 0\le q\le M$$

$$a_{s,i,j}, d_{s,i,j} \in \{0,\dots,c_{\max}\} \qquad 1\le s\le m, (v_i,v_j)\in E(G)$$

$$\bar{a}_{s,i,j}, \bar{d}_{s,i,j} \in \{0,\dots,c_{\max}\} \qquad 1\le s\le m, (v_i,v_j)\in E(G)$$

$$\bar{f}_{s,p} \in [0,\dots,c_{\max}] \qquad 1\le p\le k, 1\le s\le m$$

12.2.1.4 Generating starting points Here, we describe an algorithm based on random-number partition for generating the starting points, that are usage matrices $U_{q,0}$, in a sparse way. A usage matrix $U_{q,0} = [u_{i,p}]$ is a $n \times k$ matrix comprising a value $u_{i,p} \in \mathrm{R}$ for each clone i and each sample p such that each column sums up to 1, i.e. $\sum_{1 \leq i \leq n} u_{i,p} = 1$. Therefore, we independently generate each column $\mathbf{u}_p$ comprising the mixing proportions for sample p. Moreover, we randomly select the number of clone that are present on each sample. Let $\mathbf{u}$ be a column that we want to generate and L be the number of clones that are present in this sample. As such, we set to 0 the $n-L$ proportions in $\mathbf{u}$ and we generate the remaining L mixing proportions $u_{i,p} \geq 0$, such that $\sum_{1 \leq i \leq n} u_{i,p} = 1$, by Algorithm 2. Observe that we can generate the usage matrices in a more sparse way using this algorithm since the elements $b_1, \ldots, b_{L-1}$ defining the partition of $[0,1]$ can be chosen as $\tilde{b}_1, \ldots, \tilde{b}_{L-1}$ in $[0, \eta]$ such that $\eta \in \mathbb{N}$ and $b_j = \frac{\tilde{b}_j}{\eta}$.

Algorithm 2 Generate mixing proportions $u_1, \ldots, u_L$ that sum up to 1

1: Randomly choose $b_1, \ldots, b_{L-1}$ in [0,1]
2: Sort the elements in $\{b_1, \ldots, b_{L-1}\}$ to get a sequence $(b_1, b_2, .., b_{L-1})$ that defines a partition of [0, 1].
3: Set $u_i = b_i - b_{j-1}$ for each $i \in \{1, \ldots, L-1\}$ and $u_L = 1 - b_{L-1}$
4: **return** $u_1, \ldots, u_L$

12.2.1.5 Refinement step Our algorithm computes $(C_{q,K}, U_{q,K})$ for each starting point $U_{q,0}$ and selects the matrices (C', U^*) that minimize the distance $\|F - C_{q,K}U_{q,K}\|$. However, multiple copy-number trees T' with $\Lambda(T') \leq \Lambda_{\max}$ can be generated by C' as well as by a different copy-number matrix C'' with $\|F - C''U^*\| = \|F - C'U^*\|$. Thus, to find T^* of minimum cost $\Lambda(T^*) \leq \Lambda_{\max}$ generated by a copy-number matrix C^* with $\|F - C^*U^*\| \leq \|F - C'U^*\|$, we introduce a final step called *refinement step* that solves the following problem:

Problem 12.2 *Given an $m \times k$ fractional copy-number matrix F, a number n of clones, a $n \times k$ usage matrix U, a maximum copy-number c_{max}, and fixed value $\gamma \in \mathrm{R}$, find an $m \times n$ copy-number matrix C generating T, such that $\|F - CU\| \leq \gamma$ and $\Lambda(T)$ is minimum.*

As subsequently done in Section 12.2.2, we add a threshold of confidence ϵ to the fixed value $\|F - C'U^*\|$ of the distance such that $\gamma = \|F - C'U^*\| + \epsilon$. Intuitively, due to this threshold the algorithm admits "sufficiently-small" increases of the distance if they result in generated tree T^* with a lower cost. The algorithm uses the same value of ϵ for this refinement step and for the next searching scheme adopted for choosing Λ^* in Section 12.2.2.

We solve the refinement step by using a ILP formulation. More specifically, the ILP is a variant of the one that we designed for solving the C-step in Section 12.2.1.2. We swap the constraints imposing the number of events $\Lambda(T) \leq \Lambda_{\max}$ with the objective function. As such, the sum of the events $\sum_{(v_i,v_j)\in E(G)} \sum_{1\leq s\leq m} \bar{a}_{s,i,j} + \bar{d}_{s,i,j}$ represents the new objective function, instead the value $\sum_{1\leq s\leq m} \sum_{1\leq p\leq k} \bar{f}_{s,p}$ of the distance function is bounded by γ.

The refinement step can be useful for finding a copy-number matrix C^* generating the most parsimonious T^* with a fixed value of the distance found by the binary search scheme of our algorithm (Section 12.2.2). Let C', U^* be the copy-number matrix and usage matrix found by the binary search such that C' generates T'. By initializing the refinement step with (C', T'), we may improve the results of the binary search by finding a solution comprising a more parsimonious copy-number tree T^* generated by C^*, i.e. $\Lambda^* < \Lambda'$, with the fixed value of the distance $\|F - C'U^*\|$. Furthermore, the refinement step can be used in a two-step procedure for better exploring the results for a maximum cost Λ. First, we can apply our approach with the binary search scheme for finding a reasonable value Λ for the maximum cost providing a resulting copy-number matrix C that generates T. Next, we can apply the refinement step on the result (C, T) to find a copy-number matrix C^* generating T^* with $\Lambda(T^*) < \Lambda$. We can repeat this procedure until we find T^* with $\Lambda(T^*) = \Lambda$ and using many starting points. The main advantage of this two-step procedure is to be highly parallelizable and, thus, it can significantly benefits from the execution on a computer cluster.

12.2.2 Choosing $\Lambda_{\max}$ to Balance Cost $\Lambda(T)$ and Distance $\|F - CU\|$

The parameter $\Lambda_{\max}$ controls the tradeoff between the cost $\Lambda(T)$ of the tree T and the distance $\|F - CU\|$. We now describe a procedure for finding the smallest maximum cost Λ^* that achieves the largest decrease in the distance $\|F - CU\|$.

We indicate by (C^Λ, U^Λ) the matrices found by our approach with maximum cost Λ and we define $d(\Lambda) = \|F - C^\Lambda U^\Lambda\|$. Note that the objective function $d(\Lambda_t)$ is non-increasing with larger values of Λ_t. That is, if $\Lambda_t \geq \Lambda$ then $d(\Lambda_t) \leq d(\Lambda)$, as C^Λ generates T with cost $\Lambda(T) < \Lambda_t$. We define Λ^* as the maximum cost such that $d(\Lambda^*) = 0$ and for any $\Lambda_t \geq \Lambda^*$ the value $d(\Lambda_t)$ remains 0. Unfortunately, the value $d(\Lambda_t)$ depends on the number of starting points and is further confounded by the presence of noise that may result from mapping errors or amplification biases (such as GC-content bias). It thus reasonable to expect that $d(\Lambda^*) > 0$ and that small decreases in the value of $d(\Lambda_t)$ for any $\Lambda_t > \Lambda^*$ may be not significant due to these confounding factors. We therefore introduce the parameter ϵ such that Λ_2 provides a better solution than Λ_1 if and only if $d(\Lambda_1) - d(\Lambda_2) > \epsilon$. The threshold ϵ

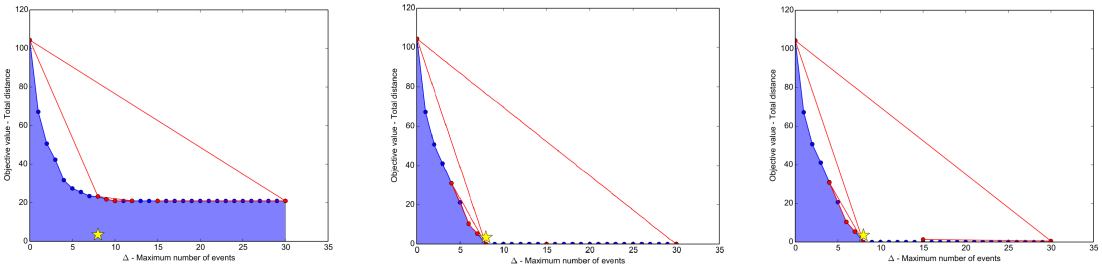

Figure 12.2: **Binary Search of Λ^*.** The figures show the behaviour of the binary search method for searching the value Λ^* of the maximum cost $\Lambda_{\max}$ on the same instance assuming different number n of leaves in $\{3,4,5\}$. The instance consists of a single chromosome with 20 segments, 4 extant clones, i.e. true number of leaves, and 10 samples. The results have been computed as described in Section 13.2.4. The blue line is showing the objective function $d(\Lambda_t)$ for all the values of Λ_t in $[0,30]$. Instead, the red points are the values considered by the searching algorithm and the red line link them following the order on which they are considered. The yellow start show the selected Λ^*.

is a user-specified parameter that controls the tradeoff between greater robustness to noise (larger ϵ) or more precision (smaller ϵ). We redefine Λ^* as the smallest integer whose solution cannot be improved by increasing the maximum cost, that is $d(\Lambda^*) - d(\Lambda_t) \leq \epsilon$ for any $\Lambda_t \geq \Lambda^*$. Note that ϵ also plays a role in the refinement step described in Section 12.2.1.5.

Now, we use that the function $d(\Lambda_t)$ is non-increasing for designing an algorithm based on binary search for finding the value Λ^*. We consider an interval $[\Lambda_{\mathrm{L}_0}, \Lambda_{\mathrm{R}_0}]$ containing Λ^* with high probability. Since the objective function $d(\Lambda_t)$ is monotonically non-increasing, we equivalently say that Λ^* is the minimum value whose solution is not better than the one of the rightmost Λ_{R_0}, i.e. $d(\Lambda^*) - d(\Lambda_{\mathrm{R}_0}) \leq \epsilon nk$. As such, we design an algorithm inspired to the binary search for finding Λ^* in $[\Lambda_{\mathrm{L}_0}, \Lambda_{\mathrm{R}_0}]$. The algorithm recursively considers an interval $[\Lambda_{\mathrm{L}_t}, \Lambda_{\mathrm{R}_t}]$ containing Λ^*. At each step, the objective value $d(\Lambda_{\mathrm{M}_t})$ is computed by our approach such that Λ_{M_t} is the integer in the middle of the interval $[\Lambda_{\mathrm{L}_t}, \Lambda_{\mathrm{R}_t}]$. If Λ_{R_0} provides a better solution than Λ_{M_t}, i.e. $d(\Lambda_1) - d(\Lambda_2) > \epsilon$, then we know Λ^* is contained in $[\Lambda_{\mathrm{M}_t}, \Lambda_{\mathrm{R}_t}]$. Otherwise, we know that Λ^* is contained in $[\Lambda_{\mathrm{L}_t}, \Lambda_{\mathrm{M}_t}]$ since the objective function is monotonically non-increasing. We recursively repeat the procedure on the next interval until Λ^* is the only left. The algorithm performs a logarithmic number of iterations. Figure 12.2 shows the results of this algorithm applied on the same instance assuming three different number of leaves.

12.2.3 Implementation and Additional Features

In this section, we present details about the implementation of our method and additional features for dealing with the characteristics of real data and exploiting additional information that may be available.

12.2.3.1 Implementation We implemented our method in C++. The LP for solving the U-step and the ILP for solving the C-step, as well as the ILP for the refinement step, have been implemented using CPLEX v12.6 [75]. Our method apply the algorithm described in Section 12.2.1 by using different starting points. We can consider each starting point independently. As such, to exploit the parallel nature of the algorithm, we based the implementation of the method on a parallel structure such that more starting points can be executed on parallel. Each *worker* executes the algorithm on a single starting point at time, and the total number of workers determine the number of starting points that are executed on parallel. The execution of the method ends when the workers executed all the starting points.

Now, we describe the approach that we applied for initializing the solution computed by either the U-step or the C-step. The iterative scheme applied by our method on each starting point (Section 12.2.1) produces a sequence of matrices $(C_t,U_t),(C_{t+1},U_{t+1}),\ldots$ such that $\|F-C_tU_t\| \geq \|F-C_{t+1}U_{t+1}\|$ as both C_{t+1} and U_{t+1} can be chosen equal to the previous C_t and U_t without changing the value of the distance $\|F-C_tU_t\|$. As such, the U-step find a usage matrix U_{t+1} that either is equal to U_t or it improves the value $\|F-C_{t+1}U_t\|$ of the distance. Symmetrically, the C-step find a copy-number matrix C_{t+1} that either is equal to C_t or it improves the value $\|F-C_tU_{t+1}\|$ of the distance. Hence, we can provide U_t and C_t as initial solutions that may be improved by the U-step and C-step, respectively. We do this by using the feature called "hotstart solution", that is provided by CPLEX's framework.

Next, we describe alternative search schemes that can be used by our method in addition to the standard binary search. Our method uses a search scheme based on the binary search paradigm for searching the maximum cost Λ^* on an interval $[\Lambda_{L_0},\Lambda_{R_0}]$. This approach is very efficient when considering a very larger interval. However, when considering small intervals a linear search could be preferred. As such, we implemented in our method the possibility to choose the search scheme for Λ^* in a interval $[\Lambda_{L_0},\Lambda_{R_0}]$:

1. **Binary search.** The running time of this search scheme is logarithmic in the size of the interval and it is ideal for considering large intervals. This is the standard search scheme used by our method since the searching interval is usually large.

2. **Linear search from $\Lambda_{\mathbf{L}_0}$ to $\Lambda_{\mathbf{R}_0}$.** The running time of this search scheme is linear in the size of the interval and it is ideal when considering small intervals. In fact, given a maximum cost $\Lambda_t \in [\Lambda_{\mathrm{L}_0}, \Lambda_{\mathrm{R}_0}]$, we can establish an initial solution in the ILP using the one found at the previous step with Λ_{t-1}, as we did above for U-step and C-step.

3. **Reverse Linear search from $\Lambda_{\mathbf{R}_0}$ to $\Lambda_{\mathbf{L}_0}$.** The running time of this search scheme is linear in the size of the interval. Recall that we are searching for a maximum cost Λ^* that is the rightmost integer in $[\Lambda_{\mathrm{L}_0}, \Lambda_{\mathrm{R}_0}]$ such that $d(\Lambda^*) - d(\Lambda_t) \leq \epsilon$ for any $\Lambda_t \geq \Lambda^*$ (Section 12.2.2). As such, when Λ^* is close to Λ_{R_0} this search scheme can be fast.

12.2.3.2 Additional Features Bulk-sequencing samples obtained from a single tumor are typically characterized by an admixture of normal cells. As such, our method allows to force the presence of the normal diploid clone in the inferred copy-number matrix C and in the correspondingly generated copy-number tree T. Whenever this feature is used, T is inferred such that the root $r(T)$ has an outgoing 0-cost edge to the clone labeled by $\mathbf{c}_0$ corresponding to the normal diploid clone. Therefore, the second outgoing edge is labeled with the clonal events since it relates $r(T)$ with the internal vertex that is ancestor of every tumor clone. Observe that our method consequently infers the tumor purity as the mixing proportion $u_{0,s}$ of the normal diploid clone $\mathbf{c}_0$ on each sample s. Moreover, forcing the presence of the normal diploid clone may have a significant impact on reducing the running time of the method as the profile $\mathbf{c}_0$ is fixed as well as the outgoing edges from $r(T)$.

Sometimes multiple signals from different sources can be used to have strong information about the clonal structure of some samples. For example, a sample can have strong evidences to be monoclonal (an example of this can be found in [104]). Therefore, we introduce an additional feature in our method that allows to specify whether a sample should be considered monoclonal. We do this in the C-step by encoding each mixing proportion $u_{i,s}$ as a binary variable (instead of a variable between 0 and 1) for each sample s that is specified as monoclonal.

Chapter 13

Results

Some of the results in this chapter are published in:

M. El-Kebir, B. J. Raphael, R. Shamir, R. Sharan, S. Zaccaria, M. Zehavi, and R. Zeira. Complexity and algorithms for copy-number evolution problems. *Algorithms for Molecular Biology*, 2016, in press.

S. Zaccaria[†], M. El-Kebir[†], G. Klau, and B. J. Raphael. The copy-number tree mixture deconvolution problem and applications to multi-sample bulk sequencing tumor data. In *Research in Computational Molecular Biology (RECOMB)*, 2017, in press.

[†]joint first authorship

In this chapter, we present and analyze the results obtained from applying the methods designed for CNT and CNTMD on simulated and real instances. Firstly, in Section 13.1 we generate instances composed of multiple integer copy-number profiles as they would be inferred from homogeneous samples. On these instances, we apply the ILP formulation designed for CNT in Chapter 12 and we analyze its performance. Next, in Section 13.2 we apply the coordinate-descent algorithm designed for CNTMD in Chapter 12 both on simulated instances and on a prostate-cancer dataset [67]. On simulated data, we compare our algorithm with alternative approaches described in the same Section 13.2. On real dataset, we consider the presence of consistent patterns on the results obtained with different parameters by our algorithm and we compare these results with the ones of previous published analyses [67].

13.1 Integer Copy Numbers

To assess the performance of the ILP for CNT, we simulated instances by randomly generating a full binary tree T with k leaves. We randomly labeled edges by events

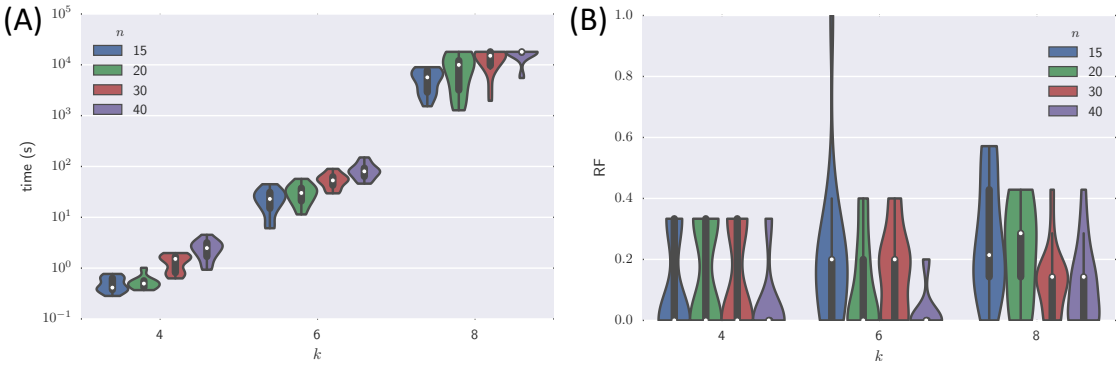

Figure 13.1: **Performance measures of the ILP algorithm for CNT.** Violin plots of running time in seconds (A) and normalized Robinson-Foulds metric for measuring the tree distance (B) for varying number k of leaves and number n of segments. Median values are indicated by a white dot in each plot. Results with $n \in \{5, 10\}$ segments are shown in Fig. 13.1.

according to a specified maximum number m of events per edge with amplifications/deletions ratio ρ. Specifically, we label an edge by d events where d is drawn uniformly from the set $\{1, \ldots, m\}$. For each event (s, t, b) we uniformly at random draw an interval $s \leq t$ and decide with probability ρ whether $b = 1$ (amplification) or $b = -1$ (deletion). The resulting instance of CNT is composed of the profiles $\mathbf{c}_1, \ldots, \mathbf{c}_k$ of the k leaves of T and e is set to the maximum value of the input profiles.

We considered varying numbers of leaves $k \in \{4, 6, 8\}$ and of segments $n \in \{5, 10, 15, 20, 30, 40\}$. In addition, we varied the number of events $m \in \{1, 2, 3\}$ and varied the ratio $\rho \in \{0.2, 0.4\}$. We generated three instances for each combination of k, n, m and ρ, resulting in a total of 324 instances.

We implemented the ILP in C++ using CPLEX v12.6 [75]. The implementation is available upon request. We ran the simulated instances on a compute cluster with 2.6 GHz processors (16 cores) and 32 GB of RAM each. We solved 302 instances (93.2%) to optimality within the specified time limit of 5 hours. Computations exceeding this limit were aborted and the best identified solution was considered. The instances that were not solved to optimality are a subset of the larger instances with $k = 8$ and $n \in \{20, 30, 40\}$. For these cases, we show in Fig. 13.1 the gap between the best identified solutions and their computed upper bounds. Moreover, Fig. 13.1 shows violin plots of running time, tree distance and optimality gap for all the simulated CNT instances.

For 323 out of 324 instances (99.7%) the tree inferred by the ILP has a cost that was at most the simulated tree cost. The only exception is an instance with $k = 8$ leaves and $n = 40$ segments that was not solved to optimality, and where the

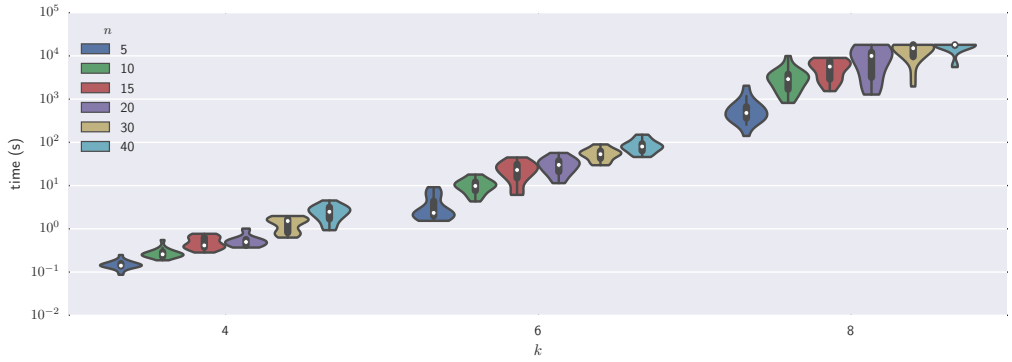

Figure 13.2: Violin plot showing the running time in seconds (log scale) for varying number k of leaves and number n of segments. Median values are indicated by a white dot in each plot.

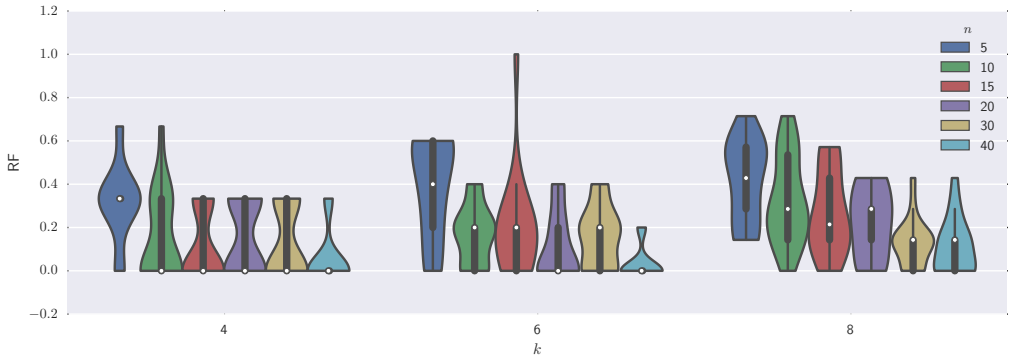

Figure 13.3: Violin plot showing the normalized Robinson-Foulds (RF) metric for varying number k of leaves and number n of segments. Median values are indicated by a white dot in each plot.

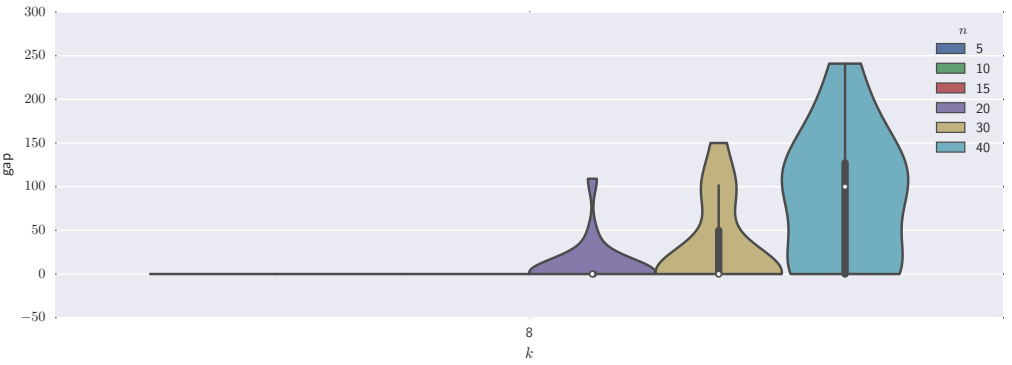

Figure 13.4: Violin plot showing the optimality gap (c) for varying number k of leaves and number n of segments. Median values are indicated by a white dot in each plot.

inferred cost was 15 vs. a simulated cost of 14. These results empirically validate the correctness of our ILP implementation.

We observe that the running time increases with the number of leaves and to a lesser extent with the number of segments (Fig. 13.1A). In addition, we assessed the distance between topologies of the inferred and simulated trees using the Robinson-Foulds (RF) metric [133]. To allow for a comparison across varying number of leaves, we normalized by the total number of splits to the range [0,1] such that a value of 0 corresponds to the same topology of both trees. For 264 instances (81.4%) the normalized RF was at most 0.35. For $k = 4$ leaves the median RF value was 0, which indicates that for at least 50% of these instances the simulated tree topology was recovered. Fig. 13.1B shows the distribution of normalized RF values with varying numbers of leaves and segments. Given a fixed number of leaves, the normalized RF value decreases with increasing number of segments. This indicates that the maximum parsimony assumption becomes more appropriate with larger number of segments, which is not surprising since amplifications and deletions are less likely to overlap. In addition, we observed (data not shown) that running time and RF values are not affected by varying values of m and ρ. In summary, we have shown that our ILP scales to practical problem instance sizes with $k = 6$ and up to $n = 40$ segments, which is a reasonable size for applications to real data [135, 143].

13.2 Fractional Copy Numbers

We applied our CNTMD algorithm to simulated data and to data from a patient from a prostate cancer dataset [67]. We present alternative approaches to our method for CNTMD in Section 13.2.1. On simulated data, we compare all the methods in Section 13.2.2. While, in Section 13.2.3 we present and analyze the results of our method applied on the prostate-cancer datasets. The experimental details about the generation of the simulated instances and the execution of our algorithm are reported in Section 13.2.4. Lastly, experiments on additional simulated instances and the related results are described in Section 13.2.5.

13.2.1 Alternative methods

We benchmarked CNTMD on simulated data, comparing its performance to several other approaches, which we now describe and whose features we summarize in Table 13.1. The first alternative approach is a "factorization-only" approach that aims to factorize a fractional copy-number matrix F into a copy-number matrix C and a usage matrix U such that $F = CU$ without imposing a tree constraint. Published approaches to this problem perform this factorization (sometimes called deconvolution) independently on each sample [54, 104, 113, 116] – one exception is [134],

method	tree	interval events	factorization	ref.
Integer Matrix Factorization (IMF)	no	no	yes	
Copy-Number Tree (CNT)	yes	yes	no	[45, 137]
soft CNT	yes	yes	no	[45]
Copy-Number Tree Mixture Deconvolution (CNTMD)	yes	yes	yes	this paper
single CNTMD	yes	no	yes	this paper

Table 13.1: **Different features of alternative methods.** Different methods for the inference of the integer copy numbers of distinct clones and their evolution from multiple heterogeneous samples. For each method, the supported features (factorization, tree reconstruction and interval events) are listed.

but this infers non-integer copy numbers and it has not been applied to multiple samples from the same tumor. These methods do not take into account any information from the context and may provide unlikely profiles characterized by many interval events without a reasonable structure (Fig. 13.5). To the best of our knowledge, there is no current method that perform this factorization, especially when F comprises multiple vectors and C is composed only of integers. Thus, we implemented *Integer Matrix Factorization (IMF)* which performs the factorization by splitting the variables, C and U, and applying a coordinate-descent algorithm. Another class of approaches use the same copy number model as CNTMD, but assume each sample is homogeneous. One strategy is to first round the entries of F before inferring a copy-number tree. We will do this by solving the CNT problem with an ILP model [45], mimicking the strategy that has been used when running MEDICC ([101, 138, 143]). We also consider a second rounding approach, which we call *soft CNT*, where we round the fractions in F in order to obtain a copy-number matrix C generating T of minimum cost, extending the ILP formulation of CNT from [45]. Finally, we also consider a variant of CNTMD, which we call *single CNTMD*. Here, we replace the interval events by single events; this is equivalent to a model where the cost of an interval event depends on the number of segments in the interval. However, the single event model is not a good representation of true copy number aberrations in cancer, as the length distribution of somatic copy number aberrations is not simply a function of length [156]. Such a copy number model was used by [104] and [24] for inferring the evolution comprising the minimum number of single events from the profile of clones inferred independently from each sample. Fig. 13.5 shows an example highlighting the weaknesses of all the alternative methods presented above.

Implementation of Alternative Methods Here, we present the details about the implementation of each alternative method reported in Table 13.1.

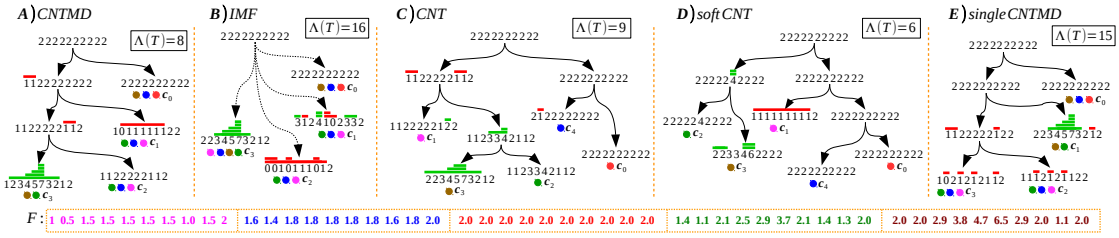

Figure 13.5: **Alternative methods infer trees that differ significantly from the true tree, which is inferred by our approach CNTMD.** Copy-number trees inferred by the methods in Table 13.1 where deletions $(s,t,-1)$ are red and amplifications $(s,t,1)$ are green. (A) Shows the true tree composed of four clones $\mathbf{c}_0$ (normal), $\mathbf{c}_1$, $\mathbf{c}_2$, $\mathbf{c}_3$ with a cost of 8. This tree is correctly retrieved by CNTMD. All the alternative methods fail to infer the clonal mutation $(1,2,-1)$. (B) The tree inferred by IMF contains too many events and differs significantly from the true tree. (C-D) CNT and soft CNT infer clones that are very different from the true clones that comprise the corresponding mixed samples such as $\mathbf{c}_1$ in (C) and $\mathbf{c}_2$ in (D). (E) single CNTMD splits the effect of the deletion $(1,8,-1)$ across two distinct clones $\mathbf{c}_2$ and $\mathbf{c}_3$ resulting in a cost of 15.

- **IMF.** We solve IMF by using a coordinate-descent algorithm similar to the one applied by our approach. Moreover, we generalize the ILP we proposed for the C-step by simply removing the variables and the constraints that we used for modeling the tree T and its labels (Section 12.2.1). As such, we implement this method considering the same structure used for the implementation of our method (Section 12.2.3).

- **CNT.** We implement the CNT method as a two-step procedure. In the first step, we round each fraction $f_{s,p}$ in the fractional copy-number matrix F to the closest integer obtaining an integer copy-number profile for each sample. In the second step, we infer the copy-number tree from such retrieved profiles using the ILP model proposed in [45].

- **soft CNT.** We implement the soft CNT method by modifying the ILP-based method proposed in [45]. The proposed ILP encodes each value of a copy-number profile as an integer variable $c_{s,i}$ for each segment s of each clone i. Thus, the value of this variable is fixed equal to the corresponding integer in input. Instead, we have in input the fractions $f_{s,i}$, since each sample s corresponds to the clone i in this case. Therefore, we introduce new constraints requiring that $c_{s,i}$ is bounded between $\lfloor f_{s,p} \rfloor$ and $\lceil f_{s,p} \rceil$.

- **single CNTMD.** single CNTMD corresponds to CNTMD where single events are considered instead of interval events. Therefore, we design a method for solving single CNTMD as a variant of our method for CNTMD. In particular,

we differently model the cost of the edges and, consequently, of the tree in the ILP for C-step. Recall that modeling single events is equivalent to model interval events whose cost is equal to the number of covered segments. As such, we model the cost of an edge (v_i, v_j) as the sum of the amplifications and deletions covering each segment s by introducing variables $\bar{a}_{s,i,j} \in \{0, \dots, c_{\max}\}$ and $\bar{d}_{s,i,j} \in \{0, \dots, c_{\max}\}$, respectively (recall that in our model $\bar{a}_{s,i,j}$ and $\bar{d}_{s,i,j}$ encode the sum of the amplifications and deletions starting at segment s instead). Hence, $\bar{a}_{s,i,j}$ and $\bar{d}_{s,i,j}$ correspond to $a_{s,i,j}$ and $d_{s,i,j}$.

13.2.2 Benchmark on Simulated Data

We compare CNTMD with the methods described above on simulated instances composed of 22 chromosomes with a total of 350 segments, instances that are of the same scale as real data. The number of segments per chromosome ranges from 5 to 50. We randomly generate three copy-number trees, denoted by $\hat{T}$, which in turn were generated by copy number matrices $\hat{C}$ composed of four tumor clones plus the normal diploid clone. We mix the leaves of each tree by a usage matrix $\hat{U}$ and obtain fractional copy-number matrices with $k \in \{2, 5, 10\}$ samples. For each tree and value for k, we generate three instances. Thus, we consider 27 simulated instances in total. Additional details on the setup of the experiments are presented in the following Section 13.2.4. Moreover, Section 13.2.5 reports the results on additional simulated instances.

We use three quality measures to compare the inferred tree T, inferred copy-number matrix C, and inferred usage matrix U to the simulated $\hat{T}$, $\hat{C}$ and $\hat{U}$. We compare T to $\hat{T}$ by considering the relative difference of events $|\Lambda(T) - \Lambda(\hat{T})|/\Lambda(\hat{T})$. To compare U to $\hat{U}$, we need to associate each inferred clone i to a corresponding true clone $\hat{i}$. Similarly to [44, 99], we search for a maximum-weight matching that minimizes the value of the *usage difference* $\|U - \hat{U}\|$ in a bipartite graph where there is a an edge $(v_i, v_{\hat{i}})$ with weight $|\mathbf{c}_i - \mathbf{c}_{\hat{i}}|$ for all pairs $(i, \hat{i})$. To compare C to $\hat{C}$, we compute a maximum-weight bipartite matching on the same complete bipartite graph where the edges are weighted by a similarity metric, called *single leaf consistency (sLC)*. sLC metric is computed by solving the *Copy-Number Triplet (CNT)* problem [45] between two inferred copy-number profiles. Given two profiles $\mathbf{c}_i, \mathbf{c}_j$, the corresponding value of sLC is the minimum cost of a copy-number tree with two leaves labeled by $\mathbf{c}_i, \mathbf{c}_j$ and with an unfixed root. In this case, the root corresponds to the closest ancestor of $\mathbf{c}_i, \mathbf{c}_j$ and sLC is equal to zero if and only if $\mathbf{c}_i, \mathbf{c}_j$ are equal. The value of *total leaf consistency (LC)* used for compare the methods (Fig. 13.7) corresponds to the total weight of the corresponding matching normalized by the number of clones and chromosomes.

Fig. 13.9 shows the results on the simulations. First, we observe that CNTMD, which combines both factorization and a proper interval tree-based model, outper-

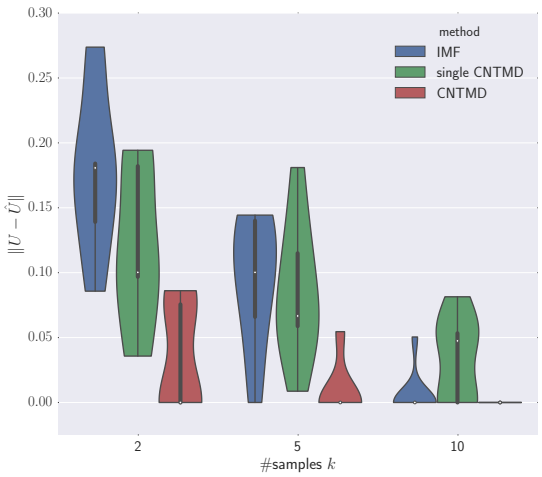

Figure 13.6: $\|U - \hat{U}\|$

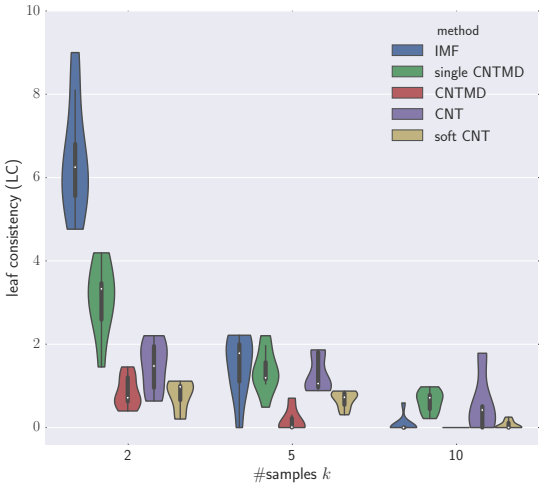

Figure 13.7: total LC

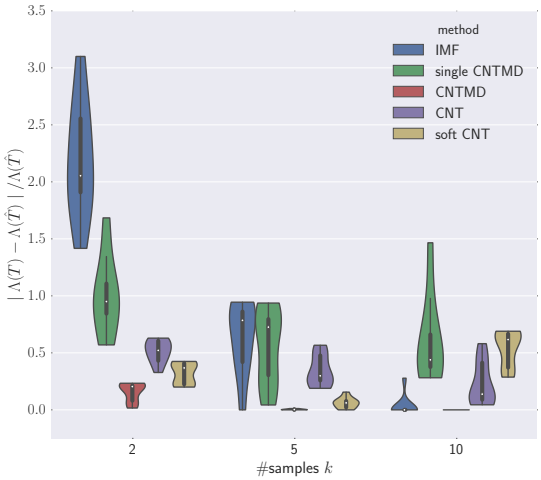

Figure 13.8: $|\Lambda(T) - \Lambda(\hat{T})|/\Lambda(\hat{T})$

Figure 13.9: **CNTMD outperforms alternative methods on simulated data.** Comparison of five methods across 27 simulated datasets with $k \in \{2, 5, 10\}$ samples, consisting of 4 tumor clones and a normal diploid clone, each with a total of 350 segments across 22 chromosomes. (A) Normalized usage difference $\|U - \hat{U}\|$ between true and inferred mixing proportion. Methods CNT and soft CNT are not shown, as these methods do not compute U. (B) Leaf consistency (LC) measure. (C) Difference $|\Lambda(T) - \Lambda(\hat{T})|/\Lambda(\hat{T})$ between the cost of the inferred tree T and the cost of the true tree $\hat{T}$. Each simulated instance was solved with the parameter n set to the true number of clones.

forms all other methods across all number of samples. Second, we see that the quality metrics improve with increasing number k of samples for all the meth-

ods. This is especially the case for the factorization-based methods (IMF, single CNTMD, CNTMD), where differences in the clonal composition across samples provide a strong signal for deconvolution (Fig. 13.6-13.7-13.8). In contrast, the rounding methods (CNT and soft CNT), show only a modest improvement with increasing number of samples (Fig. 13.7- 13.8), which is not surprising since rounding does not directly exploit differences in clonal composition across samples. Finally, observe that with a small number of samples ($k = 2$), CNTMD dramatically outperforms IMF (Fig. 13.6- 13.8), demonstrating how CNTMD leverages the extra information given by the tree constraint. Moreover, by not accounting for interval events, single CNTMD results in copy-number trees that are inconsistent with the simulated trees and have many more events (Fig. 13.8).

13.2.3 Application to Prostate Cancer Dataset

We apply our approach on prostate cancer patient A22 from the dataset of Gundem et al. [67]. Patient A22 comprises 10 samples. We use the published fractional copy numbers that were obtained by the Battenberg algorithm [113], which relies on the sample purity and tumor ploidy estimated by the ASCAT package [148].

Since the true clonal structure of these samples is unknown, we examine the consistency of different measures on the results obtained by running CNTMD with varying number of clones $n \in \{2,\ldots,8\}$. We observe a number of patterns that suggest that there are six clones in the tumor that are distinguishable by copy number aberrations; in comparison [67] estimate 16 clones using single nucleotide variants.

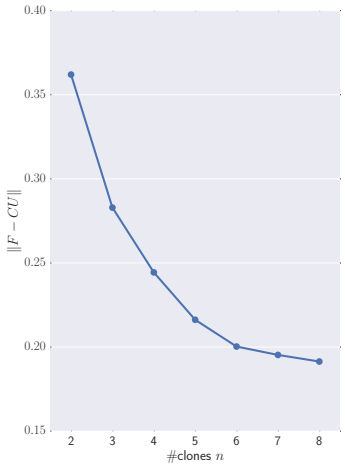

Figure 13.10: **The distance decreases with increasing number of clones n and flattens with $n > 6$ clones.** The y-axis shows the normalized value of the distance $\|F - CU\|$ for each n in $\{2,\ldots,8\}$.

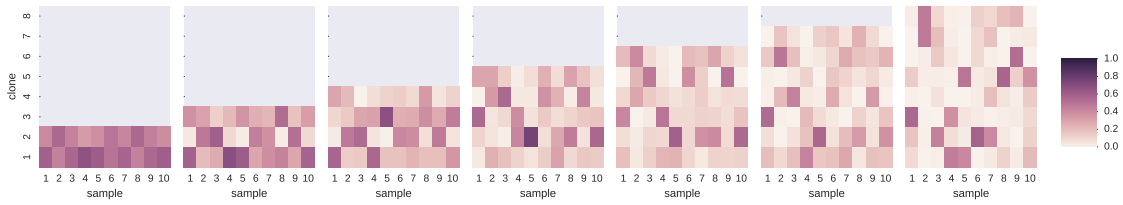

Figure 13.11: **Proportions are well-supported with a number n of clones up to 6.** Inferred usage matrices C for n in $\{2,\ldots,8\}$.

First, we observe that the value of $\|F - CU\|$ decreases significantly with increasing values of n (Fig 13.10). However, the rate of decrease declines for $n > 6$, suggesting that additional clones are not providing substantial gain in fitting the observed copy number fractions. Second, we find that the entries of the usage ma-

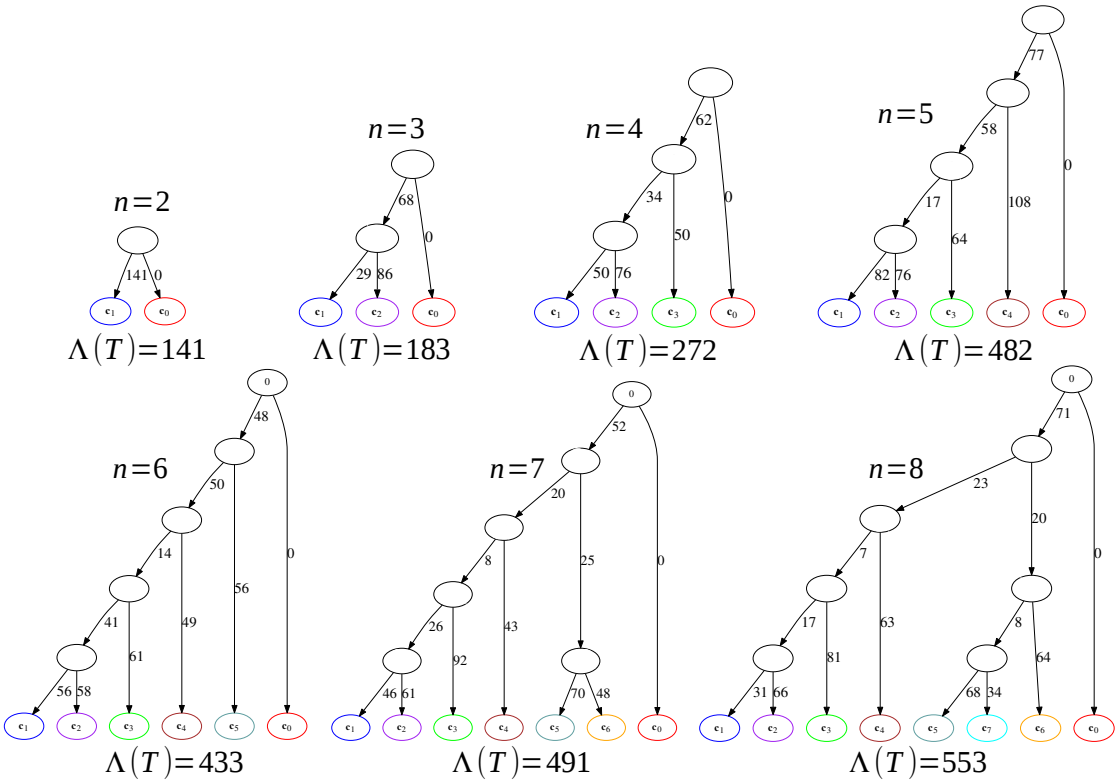

Figure 13.12: **The clones of trees with $n \leq 6$ build the same cascading topology and well-supported edges, whereas trees with $n > 6$ clones conserve the same cascading topology and introduce less-supported edges.** For each copy-number tree T with $n \in \{2, \ldots, 8\}$ clones, we show the cost $\Lambda(T)$ and label the edges by their corresponding costs. The colors of leaves map corresponding clones in the topologies. The red clone corresponds to the normal diploid clone.

trix U for $n \leq 6$ have well-supported proportions with reasonable mixing proportions for each clone in several samples (Fig. 13.11). In contrast, for $n \geq 7$, we identify clones with very low mixing proportions across samples (such as $\mathbf{c}_5$ for $n = 7$ and $\mathbf{c}_4$ for $n = 8$) suggesting that the additionally inferred clones are not supported by the data. Third, we consider the topologies and costs of inferred trees with varying number of clones and find that the tree $n = 6$ clones best describes the data. We find that most of the clonal events, which are events that are shared by all tumor clones and occur on the first branch of the tree, are consistent across the majority of the trees with $n \leq 6$ clones (Fig. 13.16). Moreover, the clones of the trees with $n \leq 6$ build a cascading topology introducing an additional branch for every increase in n (Fig. 13.12). While, with $n > 6$ clones the trees conserve the same cascading topology and each additional clone splits a previous clone (from the tree with $n-1$ clones)

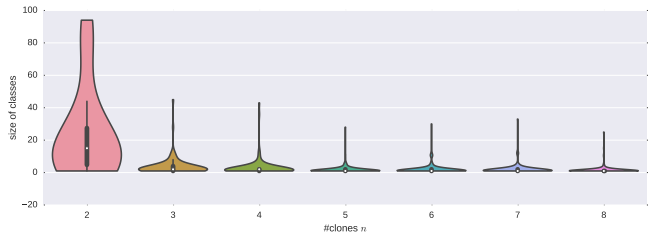

Figure 13.13: **Classes of segments with the same evolutionary history is decreasing by increasing the number n of clones**. The figure shows the distribution of the size of classes for each number $n \in \{2,\ldots,8\}$.

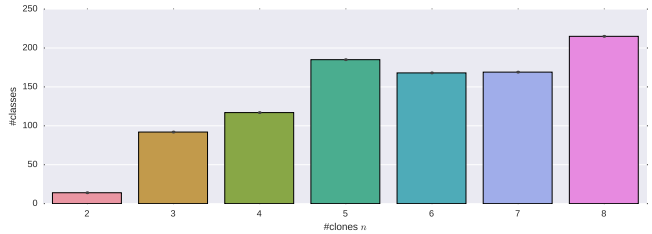

Figure 13.14: **Classes of segments with the same evolutionary history is decreasing by increasing the number n of clones.** The figure shows the number of classes for each number n of clones in $\{2,\ldots,8\}$.

into two new sibling clones, potentially overfitting the data. The total number of events, $\Lambda(T)$, stabilizes between $n = 5$ and $n = 6$ before increasing again for $n \geq 6$. The trees with more than six clones have several edges with only a few events as opposed to the trees with $n \leq 6$ clones. In sum, these findings suggest that the tree with $n = 6$ clones provides a good explanation of the data in comparison with the other trees that either overfit the data ($n > 6$) or do not accurately represent the clonal structure of the data ($n < 6$).

Finally, we examine the relationship between the inferred matrix C and the observed fractional copy number matrix F, checking whether segments with close values of F across samples are assigned the same copy number values in C, as we vary the number n of clones. We do this by partitioning the segments into *classes* with the same evolutionary history in the inferred tree T (which is derived from the inferred C). Specifically, we define a class to be a set of segments that have the same copy-number change on all edges of T. Consequently, segments in the same class have the same copy number in all the clones. By increasing n, we observe that the number of classes increases (Fig. 13.14), whereas their size decreases (Fig. 13.13). However, the size and number of classes do not significantly change with $n \geq 6$.

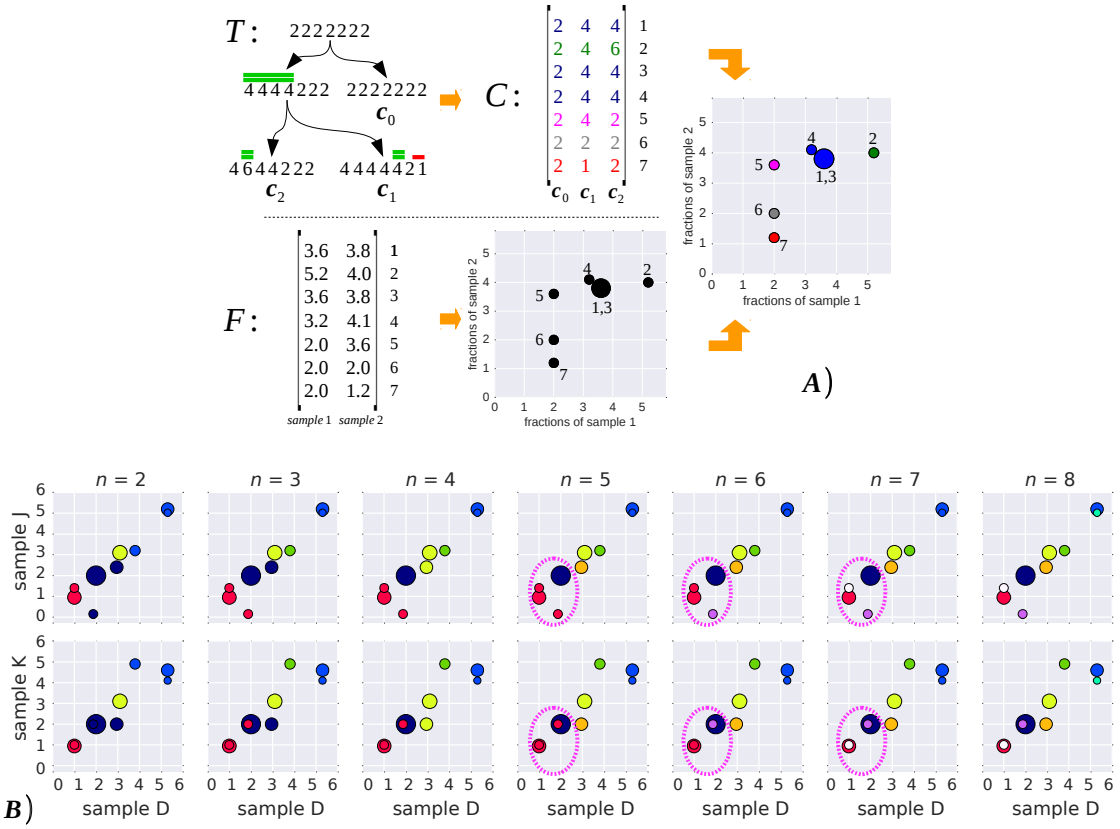

Figure 13.15: **Classes of segments with the same evolutionary history highlight consistency of the inferred solutions with the input data.** (A) We use the inferred tree T for partitioning seven segments into classes. All segments that have the same copy-number change on each edge of T are in the same class (indicated by the color). Next, we consider a pair of samples and plot the fractions of the segments (size of the dot indicates the number of segments with the same class and same fractions). The class is *consistent* if the all the segments of the same class are located close to each other. (B) Fractional copy numbers for three A22 prostate cancer samples: D, K and J. The largest dot contains 14 segments. Each column represents a value of $n \in \{2,\ldots,8\}$. The consistency of the classes improves with increasing n. The red class in $n = 5$ is composed of two subsets of segments: segments that have one copy in all the considered samples, and segments that have two copies in samples D,K and zero copies copies in sample J. With $n = 6$ these two subsets are separated into different classes (red and purple). With $n = 7$ one more class (white) is introduced for these segments, potentially overfitting the data.

Next, we assess the consistency between these classes and F. For each pair of samples p_1, p_2, we plot the fractional copy numbers of each segment in these samples, coloring segments by their class (overlapping segments with the same values result in larger dots). Fig. 13.15 gives a schematic of this procedure. We see that for $n < 6$,

segments in the same class are apart in at least one pair of samples (red, dark blue, and green clusters in Fig. 13.15), suggesting a poor fit to the data. On the other hand, for $n > 6$, segments with slightly different fractional copy numbers are separated (red/white clusters for $k = 7$ and light blue/cyan clusters for $k = 8$), suggesting overfitting of the data. Thus, this analysis also indicates that $n = 6$ appears to provide a reasonable partition into classes.

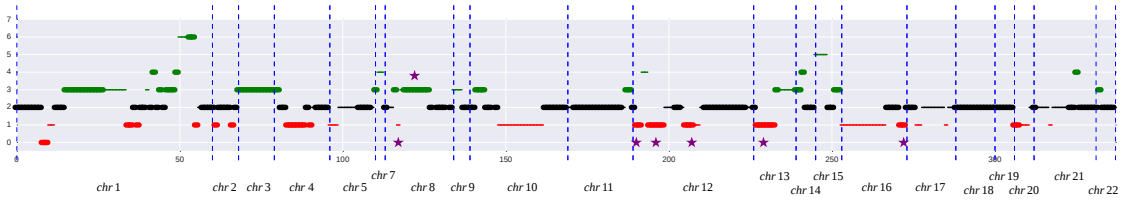

Figure 13.16: **Well-supported clonal events correspond to published clonal CNAs.** This plot shows the copy number of the clonal events inferred with $n = 6$ clones. We indicate separate chromosomes with dashed blue lines (chromosome 6 has been filtered out due to outliers). Green lines indicate amplifications and red lines indicate deletions. Note that the lengths are proportional to the number of segments and not to the corresponding genome length. Thick lines indicate events that are shared by the majority of the inferred trees T (with varying n). Purple stars indicate events that correspond to published clonal CNAs [67].

We also compare our inferred clonal copy number aberrations (CNAs) to the published clonal CNAs in [67] (i.e., CNAs labeling edges in the published trees in [67]): We observe that several clonal events in our inferred T correspond to the these CNAs (Fig 13.16): three inferred deletions on chr13 match the reported 13q deletion; 1 inferred deletion on chr 12 matches the reported 12p deletion; 1 deletion on chr8 matches the 8p LOH; 1 amplification on the same chr 8 matches the 8q gain; and two chr 16 deletions match the reported 16q LOH. More interestingly, most of these events are clonal in the majority of the inferred trees for every n (Fig 13.16). Thus, other recurrent and well-supported events in the inferred tree T are likely to be real, giving additional information about the clonal composition of these samples.

13.2.4 Experimental setup

In this section we report details about the experiments and their execution. First, we give details about the generation of simulated instances and about their execution. Next, we describe how we run our algorithm on the prostate-cancer dataset.

Simulation setup Here, we firstly describe the procedure for generating the simulated instances for a number n of clones and a number k of samples, and then we describe the main parameters that we used for executing the considered instances.

Each instance was obtained by randomly generating a copy-number tree T with n leaves and mixing these in order to obtain the m samples. The topology of T has been randomly chosen among all the possibilities and the edges have been randomly labeled by a maximum of 3 events for each edge. Each event (s,t,b) is generated by arbitrarily choosing the extremities s,t and selecting b equal to either +1 or -1 with a probability of 70% and 30%, respectively. We obtained the mixing proportions through a Dirichlet distribution such that each sample is a mixture of a subset of randomly-chosen extant clones. The samples are thus obtained by mixing the copy-number profiles of the n leaves accordingly to the corresponding proportions. Moreover, we randomly introduced noise in the resulting samples such that each fraction $f_{s,p}$ is perturbed by a noise selected from a normal distribution with a mean equal to 0 and a variance equal to the 10% of the value $f_{s,p}$. Note that for each combination of parameters we generate 3 random instances.

Each instance has been executed by randomly selecting 160 starting points $\{U_{1,0},\dots,U_{Q,0}\}$, considering at most $K = 5$ iterations, and granting at most 32GB of memory and 45 minutes of running time for each iteration. Moreover, we considered large interval equal to $[0,1000]$ for finding the maximum cost Λ^*, and we use a threshold ϵ equal to 0.01 for managing the presence of noise in the data.

Real-data setup. To better deal with these large datasets, we apply our method using a two-step procedure as described in Section 12.2.1.5. Firstly, we execute each instance on large intervals $[0,2000]$ for searching the maximum cost Λ^* imposing few hours as time limit, with 1 hour as time limit of each iteration and $K = 5$, and randomly choosing 240 starting points. This first step provides clues on smaller intervals that may contain Λ^*. Hence, we execute again our approach on these smaller intervals providing an higher time limit (12 hours for each starting point), a larger number of starting points (few hundreds 600), and exploiting the refinement step described in Section 12.2.1. Moreover, we select a value of the threshold ϵ varying between 10^{-2} and 10^{-5}. We execute the instances on a cluster of computers.

13.2.5 Additional Experiments on Simulations

Here, we present the results on additional simulated instances to assess the quality of the results even in the presence of small variations between the true number $\hat{n}$ of clones and the inferred number n of clones. We generate the simulations using the procedure described in Section 13.2.4 obtaining instances consisting of a single chromosome with 20 segments, a number $\hat{n}$ of true clones in $\{4,6,8\}$, and a number

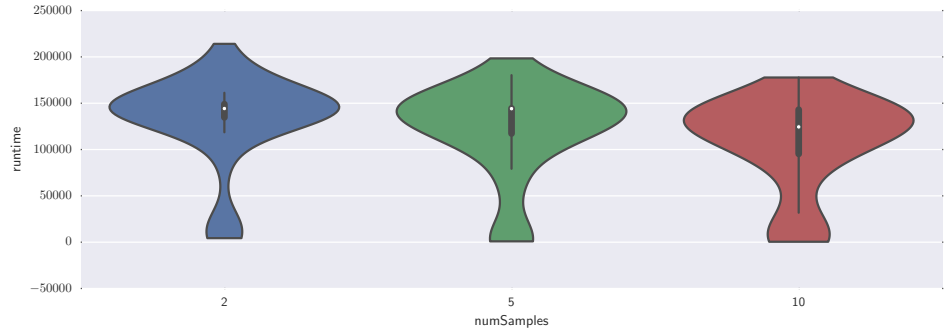

Figure 13.17: **Our method scales by increasing the number k of samples.** Running time for all the instances on different number k of samples in $\{2,5,10\}$.

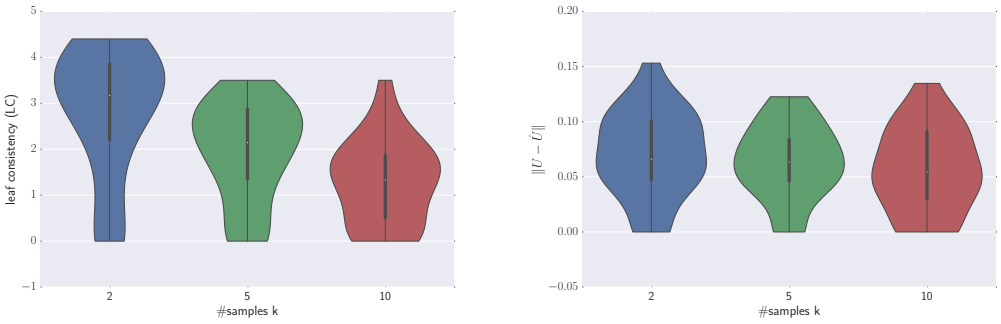

Figure 13.18: **Quality measures improve by increasing the number k of samples.** (A) Leaf consistency (LC) and (B) usage difference ($\|U - \hat{U}\|$) for increasing number k of samples in $\{2,5,10\}$.

k of samples of $\{2,5,10\}$. We vary the number n of inferred clones by increasing or decreasing $\hat{n}$ by 1.

We consider the running time of our method and we assess the quality of the results in terms of leaf consistency (LC) and usage differences $\|U - \hat{U}\|$ (Section 13.2.2). Fig. 13.17 shows the running time of our method for each number k of samples. We observe that by increasing the number k of samples, the running time of our approach decreases as the convergence threshold is easier to reach. As such, our method can scale with an increasing number of samples. Next, we consider the quality measures LC and $\|U - \hat{U}\|$ with increasing the number k of samples in Fig. 13.2.5 and Fig. 13.2.5. As described for the main simulated instances on Section 13.2.2, both the quality measures improves by increasing k. Here, we analyze how LC and $\|U - \hat{U}\|$ vary when considering small variations between the true

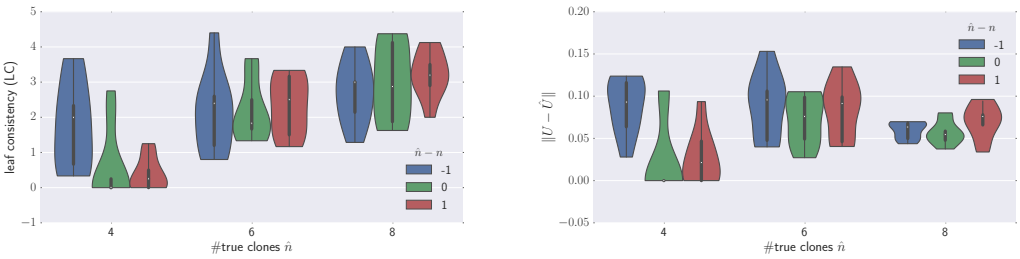

Figure 13.19: **Variations between true number $\hat{n}$ of clones and inferred number n of clones are small.** (A) Leaf consistency (LC) and (B) usage difference ($\|U-\hat{U}\|$) for increasing number of true clones $\hat{n}$ and different number of inferred clones such that the difference $\hat{n}-n$ is in $\{-1,0,1\}$.

number $\hat{n}$ of clones and the inferred number of clones n, i.e. $\hat{n}-n$ (Fig. 13.2.5 and Fig. 13.2.5). We observe that, with the exception of $\hat{n}=4$ and $n=3$, small variations $\hat{n}-n$ do not have a significant impact on the quality measures. This observation is especially true by increasing the $\hat{n}$. In fact, the difference results in terms of LC and $\|U-\hat{U}\|$ for small variations $\hat{n}-n$ by increasing the value of $\hat{n}$ and n.

Chapter 14

Discussion

In this part of the thesis we formally introduced two problems for the analysis of tumor evolution in terms of copy-number aberrations. CNT infers the evolution when the integer copy-number profile of each clones is available, whereas we formulated CNTMD for simultaneously inferring the clonal structure and their evolution from multiple mixed samples of the same tumor.

Firstly, we showed that the general CNT problem is NP-hard and gave an ILP solution. Moreover, we assessed the performance of our tree reconstruction on simulated data. While all formulations describe copy-number profiles on a single chromosome, our results readily generalize to multiple chromosomes. In addition, while our formulations presently lack the phasing step performed in [135], both the ILP formulation can be extended to support phasing. We note that experiments on real cancer sample data are required to establish the relevance of our formulations. To this end, several extensions to our models might be required. These include handling fractional copy-number values that are a result of most experiments and handling missing data for some segments. Furthermore, signals from other kind of mutations can be jointly considered to improve the accuracy of the results. In particular, SNVs are often considered in many other works [67, 104, 113]. Moreover, for the considered patient we found that a number 6 of clones is a reasonable explanation when considering only CNAs, in contrast to the 16 clones that have been inferred in [67] by considering only single-nucleotide variants (SNVs). This result may indicate that CNAs have less resolution in the distinction of clones. However, the signal from copy numbers is not confounded by other signals, unlike the case of SNVs. Therefore, this work is the first step toward an high-resolution approach, alternative to others [35, 46, 78], for inferring tumor composition by merging the signals from both CNAs and SNVs.

Next, we formulated the Copy-Number Tree Mixture Deconvolution (CNTMD) Problem, and derived a coordinate-descent algorithm, with alternating ILP and LP steps, to solve this problem. CNTMD builds a phylogenetic tree describing copy

number evolution directly from mixed-samples, thus addressing an important issue in applying phylogenetic analysis to tumor samples. We show that CNTMD outperforms approaches that only perform deconvolution – thus ignoring the phylogenetic relationship between samples – or that build phylogenetic trees assuming that each sample is homogeneous, consisting of a single clone. We also apply CNTMD to a complex dataset from prostate cancer, building a phylogenetic tree containing multiple distinct clones, mixed in different proportions across samples. These results demonstrate the feasibility of our approach to real-sized datasets. Moreover, for the considered patient we found that a number 6 of clones is a reasonable explanation when considering only CNAs, in contrast to the 16 clones that have been inferred in [67] by considering only single-nucleotide variants (SNVs). This result may indicate that CNAs have less resolution in the distinction of clones. However, the signal from copy numbers is not confounded by other signals, unlike the case of SNVs. Therefore, this work is the first step toward an high-resolution approach, alternative to others [35, 46, 78], for inferring tumor composition by merging the signals from both CNAs and SNVs.

There are a number of directions for future work. First, the hardness of CNTMD remains open. Moreover, the method does not scale with the number of clones, that are the leaves of the copy-number tree. As such, when considering more than 9 clones, one should use heuristics to estimate a solution of the C-step. Other directions include methods to better address noise in the copy number fractions, using confidence intervals or posterior distributions to model the uncertainty in entries of F. In addition, one could use model selection or regularization approaches to estimate the number of clones in a tree and avoid overfitting. Another avenue of investigation is to incorporate uncertainty in the segmentation of the genome into the model. One could also augment the phylogenetic reconstructions with single-cell measurements including FISH [24] or single-cell sequencing [32]. Finally, one could extend the approach using more sophisticated models of genome evolution, including models that include additional genome rearrangements and complex patterns of duplication – some promising work in this direction is found in [95, 98, 103, 115].

Chapter 15

Conclusions

> Biology easily has 500 years of exciting problems to work on.
>
> Donald E. Knuth (1938)

This last chapter firstly presents several directions for future work for the contributions investigated in this thesis. In particular, we propose a way to integrate the different signals from SNPs and CNAs that have been considered separately in Part I and in Part II. Next, the chapter ends with some considerations concerning the significance of the combinatorial-optimization scheme that guided all the contributions in this work. We report these considerations directly referring to some of the contributions in both the parts of the thesis.

15.1 Future Directions

In this thesis we investigate questions related to the inference of genomic variants from sequencing reads of normal and cancer genomes. For haplotype assembly, we answer several open questions related to the tractability and approximability of the MEC problem that concerns aspects of practical interest and formally extend the problem to the analysis of polyploid genomes. Based on the structure of the problem emerging from this study, we design the algorithm HapCol aiming to assembly the long reads obtained from future-generation sequencing technologies by exploiting the uniform distribution of sequencing errors that characterize these promising data. For the quantification of intra-tumor heterogeneity, we formally introduce the CNT problem that given the copy-number profiles of certain clones aims to infer the evolution of CNAs as the copy-number tree composed of a minimum number of interval events. We show that CNT is NP-hard and we design a ILP formulation that scales to instances of practical size. However, most cancer sequencing studies

consider heterogeneous samples from bulk sequencing that comprise many cells of different clones. Therefore, we introduce the CNTMD problem that given multiple heterogeneous samples aims to deconvolve these into the copy numbers of extant clones and their proportions such that, under a principle of parsimony, the evolution of CNAs is described by a minimum number of interval events. We design a coordinate-descent algorithm for solving a distance-based variant of CNTMD that incorporates an extended version of the ILP for CNT. This algorithm outperforms alternative approaches of simulated data. While, on data from a prostate cancer patient the method provides a much higher-resolution view of copy number evolution of this cancer than published analysis.

Several extensions of the contributions for both the applications in Part I and in Part II are particularly interesting. We investigated the haplotype assembly when a single individual is considered. However, one can obtain and exploit more information from jointly considering more people in parental relationship. For the sake of simplicity, we can consider a *trio* comprising father, mother, and child, but the concept is easily generalizable to larger families. In this case, we have in input sequencing reads from each individual. The main goal is still to reconstruct the pair of haplotypes of each individual, but a trio provides both constraints on the solution and a way to infer the mutations called *recombinations*. The child inherits one haplotype from the father and the other from the mother, therefore the haplotype assembly may highly benefit from jointly considering the individuals of the family. However, the haplotype that a child inherits from one of the parents may be a combination of the two haplotypes of the parent due to the effect of recombinations. As such, the joint haplotype assembly of the individuals in a family can be also applied to infer those recombinations needed to fully characterize the inheritance process.

Another interesting extension concerns the model of CNAs. We modeled the copy numbers of a genome as copy-number profiles that indicate the number of copies for each genomic segment, and we model the effects of CNAs as interval events for capturing their impact on multiple contiguous segments. However, such profiles encode only the information about the number of copies, but not about their position in the genome. In fact, a CNA amplifying or deleting genomic segments may change the genomic structure of the affected copies: an amplification may introduce a copy of a segment in a different position, whereas a deletion make consecutive two previously non-adjacent segments. As such, a model encoding information about the genomic positions of the copies may by useful to better model the effects of subsequent CNAs on the same segments. Promising attempts in this direction are found in [95, 103, 115].

Lastly, interesting directions concern the integration of the signals coming from CNAs and SNPs that we considered separately in this thesis. The human genome comprises two copies of each genomic segment, one on each haplotype, and we infer

the total copy number of each segment from the corresponding number of reads. The two copies of a segment have different values, or *alleles*, on a covered SNP. The *B-Allele Frequency (BAF)* measures the ratio of the reads covering the two distinct alleles. As such, a shift in the BAF of a SNP can be used to distinguish the number of copies in the maternal and paternal haplotypes. This approach revealed to be promising in previous works [113, 136]. Hence, CNTMD can be extended using a similar approach to distinguish the copy numbers on the two distinct haplotypes. We can model the evolution of the two copies independently and, consequently, we can infer the CNAs that occurred on the same haplotype. SNPs are not the only frequent form of point mutations. *Single-Nucleotide Variants (SNVs)* are somatic point mutations and, consequently, they are not typically inherited and shared among a population. Somatic SNVs are another frequent form of mutations in cancer. The proportion of clones comprising a specific SNV can be inferred from the ratio of reads with that mutation [47, 99], however this ratio is affected by the copy number of the corresponding genomic segment [46]. A CNA amplifying or deleting the region that contains a SNV increases or decreases, respectively, the number of reads covering that mutation and affects the corresponding proportion. Therefore, building upon the work done here for CNTMD, we can integrate the signals from SNVs with the ones from CNAs and, even, SNPs.

15.2 Closing Remarks

All the contributions in both the parts of this thesis have been guided by a scheme based on combinatorial optimization and described in Chapter 2. The scheme starts from a biological questions and is composed of four steps such that each of these is crucial for the next. The main challenge of the first step is to formulate a problem that models the aspects significant for the application. Hence, this step is important for obtaining an approach able to provide a meaningful answer to the starting biological question. Moreover, a careful investigation of the application may reveal characteristics to exploit for obtaining a problem formulation with a structure helpful in the designing of the algorithm. For example, in Part I we introduce a new problem formulation that exploits the uniform distribution of sequencing errors characterizing the long reads from future-sequencing technologies. Modeling this characteristic allows to design an algorithm requiring less computational resources by limiting the number of feasible solutions. Clearly, when considering more characteristics of the application we need to deal with a trade-off between obtaining a more refined model of the biological process and a more complex problem formulation with too many details.

The second steps aims to study the computational complexity of the problem formulation and its structure. Based on the properties of the problem structure, the

computational complexity reveals the kind of algorithms we can design. For MEC, we characterize the relationship between a feasible solution and a bipartion of the fragments. Whereas, for CNT we identify a sorting of the interval events characterizing optimal solutions. Using these properties, we reduce known NP-hard problems to MEC and CNT showing they are hard as well. Studying the computational complexity is not an easy task, especially for some problems. In fact, the computational complexity of CNTMD remains unknown. However, we can still suspect that CNTMD is NP-hard, because the related "subproblem" CNT is NP-hard. This an example corroborating the benefits of a "progressive" approach that deals with a biological question by starting from a simpler problem formulation and proceeding with more refined variants. For example, we apply this approach in Part II where we firstly consider the CNT problem that assumes to know the copy numbers of each clone and then the CNTMD problem extending the approach to copy numbers mixed in heterogeneous samples.

The goal of the third step is the design of an algorithm for the proposed problem formulation. An efficient algorithm can be designed if the problem is in P, otherwise one of the technique described in Chapter 2 for dealing with hardness can be applied. In this step the problem structure revealed in the previous step is crucial. In fact, for the MEC problem we identify a necessary and sufficient condition for the feasible solutions based on a property of consecutive columns. Therefore, we design a fixed-paramater tractable algorithm based on a dynamic programming that considers iteratively each column of the fragment matrix. Whereas, for CNT and CNTMD we design two ILP formulations based on the sorting of interval events in optimal solutions and on expressing the edge cost only considering the starting and ending positions of the interval events.

The last step concerns the validation of the algorithm and the interpretation of the results obtained on biological data for answering the starting question. Both the parts of this thesis show the benefits of validating the algorithm on simulated instances. In fact, such kind of experiments are useful in Part I for comparing the algorithm to current state-of-the-art methods on instances with a size and characteristics that are not currently available, but reflect the data produced by new technologies. Instead, simulated instances reveal to be useful in Part II for assessing the performance of the methods when the ground-truth is known. In addition, simulated instances are useful because a meaningful comparison require a significant amount of instances that often are not available from biological data. Obviously, real data represent the ultimate goal for answering the starting biological question. When benchmark instances are available as in Part I, the quality of the results can be evaluated with measures that capture the crucial aspects of the application. Conversely, when benchmark instances are not available as in Part II, we establish the quality of the solutions by determining the presence of common pat-

terns with previous analyses or by looking to the consistency of the results when varying the input parameters.

Bibliography

[1] 1000 Genomes Project Consortium et al. An integrated map of genetic variation from 1,092 human genomes. *Nature*, 491(7422):56–65, 2012.

[2] D. Aguiar and S. Istrail. HapCompass: A fast cycle basis algorithm for accurate haplotype assembly of sequence data. *J. of Computational Biology*, 19(6):577–590, 2012.

[3] D. Aguiar and S. Istrail. Haplotype assembly in polyploid genomes and identical by descent shared tracts. *Bioinformatics*, 29(13):352–360, 2013.

[4] M. Aldinucci, A. Bracciali, T. Marschall, M. Patterson, N. Pisanti, and M. Torquati. High-performance haplotype assembly. In *Computational Intelligence Methods for Bioinformatics and Biostatistics - 11th International Meeting, CIBB 2014, Cambridge, UK, June 26-28, 2014, Revised Selected Papers*, pages 245–258, 2014.

[5] A. Atamtürk and M. W. Savelsbergh. Integer-programming software systems. *Annals of Operations Research*, 140(1):67–124, 2005.

[6] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and approximation: Combinatorial optimization problems and their approximability properties*. Springer Science & Business Media, 1999.

[7] P. Bachmann. *Die analytische zahlentheorie*, volume 2. Teubner, 1894.

[8] V. Bansal and V. Bafna. HapCUT: an efficient and accurate algorithm for the haplotype assembly problem. *Bioinformatics*, 24(16):i153–i159, 2008.

[9] E. Bao, T. Jiang, and T. Girke. BRANCH: boosting RNA-Seq assemblies with partial or related genomic sequences. *Bioinformatics*, 29(10):1250–1259, 2013.

[10] L. Baumbusch, J. Aarøe, F. Johansen, J. Hicks, H. Sun, L. Bruhn, K. Gunderson, B. Naume, V. Kristensen, K. Liestøl, A.-L. Børresen-Dale, and O. Lingjærde. Comparison of the agilent, roma/nimblegen and illumina platforms for classification of copy number alterations in human breast tumors. *BMC Genomics*, 9(1):379, 2008.

[11] N. Beerenwinkel, S. Beretta, P. Bonizzoni, R. Dondi, and Y. Pirola. Covering pairs in directed acyclic graphs. In *Language and Automata Theory and Applications (LATA)*, volume 8370 of *LNCS*, pages 126–137. Springer, 2014.

[12] N. Beerenwinkel, S. Beretta, P. Bonizzoni, R. Dondi, and Y. Pirola. Covering pairs in directed acyclic graphs. *The Computer Journal*, 58(7):1673–1686, 2015.

[13] E. Berger, D. Yorukoglu, J. Peng, and B. Berger. Haptree: A novel bayesian framework for single individual polyplotyping using ngs data. *PLoS Comput Biol*, 10(3), 03 2014.

[14] A. Biere, M. Heule, and H. van Maaren. *Handbook of satisfiability*, volume 185. ios press, 2009.

[15] P. Bonizzoni, G. Della Vedova, R. Dondi, and J. Li. The haplotyping problem: An overview of computational models and solutions. *J. Comput. Sci. Technol.*, 18(6): 675–688, 2003.

[16] P. Bonizzoni, R. Dondi, G. W. Klau, Y. Pirola, N. Pisanti, and S. Zaccaria. On the fixed parameter tractability and approximability of the minimum error correction problem. In *26th Annual Symposium on Combinatorial Pattern Matching (CPM)*, volume 9133 of *LNCS*, pages 100–113, 2015.

[17] B. Browning and S. Browning. Haplotypic analysis of Wellcome Trust case control consortium data. *Human Genetics*, 123(3):273–280, 2008.

[18] S. R. Browning and B. L. Browning. Haplotype phasing: existing methods and new developments. *Nature reviews. Genetics*, 12(10):703–714, 2011.

[19] J. Bryois, A. Buil, D. M. Evans, J. P. Kemp, S. B. Montgomery, D. F. Conrad, K. M. Ho, S. Ring, M. Hurles, P. Deloukas, G. Davey Smith, and E. T. Dermitzakis. Cis and trans effects of human genomic variants on gene expression. *PLOS Genetics*, 10(7): 1–11, 07 2014.

[20] M. O. Carneiro, C. Russ, M. G. Ross, S. B. Gabriel, C. Nusbaum, and M. A. DePristo. Pacific Biosciences sequencing technology for genotyping and variation discovery in human data. *BMC genomics*, 13(1):375, 2012.

[21] S. L. Carter, K. Cibulskis, E. Helman, A. McKenna, H. Shen, T. Zack, P. W. Laird, R. C. Onofrio, W. Winckler, B. A. Weir, et al. Absolute quantification of somatic dna alterations in human cancer. *Nature biotechnology*, 30(5):413–421, 2012.

[22] M. J. Chaisson, J. Huddleston, M. Y. Dennis, P. H. Sudmant, M. Malig, F. Hormozdiari, F. Antonacci, U. Surti, R. Sandstrom, M. Boitano, et al. Resolving the complexity of the human genome using single-molecule sequencing. *Nature*, 517(7536):608–611, 2015.

[23] Z.-Z. Chen, F. Deng, and L. Wang. Exact algorithms for haplotype assembly from whole-genome sequence data. *Bioinformatics*, 29(16):1938–45, 2013.

[24] S. A. Chowdhury, S. E. Shackney, K. Heselmeyer-Haddad, T. Ried, A. A. Schäffer, and R. Schwartz. Algorithms to model single gene, single chromosome, and whole genome copy number changes jointly in tumor phylogenetics. *PLoS Comput Biol*, 10 (7):1–19, 07 2014.

[25] R. Cilibrasi, L. Van Iersel, S. Kelk, and J. Tromp. The complexity of the single individual SNP haplotyping problem. *Algorithmica*, 49(1):13–36, 2007.

[26] G. Ciriello et al. Emerging landscape of oncogenic signatures across human cancers. *Nat Genet*, 45:1127–1133, 2013.

[27] D. F. Conrad and M. E. Hurles. The population genetics of structural variation. *Nature genetics*, 39:S30–S36, 2007.

[28] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158. ACM, 1971.

[29] T. H. Cormen. *Introduction to algorithms*. MIT press, 2009.

[30] C. Darwin. *The origin of species*. Lulu. com, 1872.

[31] S. Das and H. Vikalo. SDhaP: haplotype assembly for diploids and polyploids via semi-definite programming. *BMC Genomics*, 16(1):260, 2015.

[32] A. Davis and N. E. Navin. Computing tumor trees from single cells. *Genome biology*, 17(1):1, May 2016.

[33] L. De Moura and N. Bjørner. Z3: An efficient smt solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer, 2008.

[34] M. A. DePristo et al. A framework for variation discovery and genotyping using next-generation dna sequencing data. *Nature Genetics*, 43(5):491–498, 2011.

[35] A. G. Deshwar, S. Vembu, C. K. Yung, G. H. Jang, L. Stein, and Q. Morris. Phylowgs: reconstructing subclonal composition and evolution from whole-genome sequencing of tumors. *Genome biology*, 16(1):1, 2015.

[36] L. Ding, M. C. Wendl, J. F. McMichael, and B. J. Raphael. Expanding the computational toolbox for mining cancer genomes. *Nature Reviews Genetics*, 15(8):556–570, 2014.

[37] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical programming*, 91(2):201–213, 2002.

[38] R. Dondi. New results for the Longest Haplotype Reconstruction problem. *Discrete Applied Mathematics*, 160(9):1299–1310, 2012.

[39] R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

[40] J. Duitama, T. Huebsch, G. McEwen, E.-K. Suk, and M. R. Hoehe. ReFHap: a reliable and fast algorithm for single individual haplotyping. In *BCB*, pages 160–169. ACM, 2010.

[41] J. Duitama et al. Fosmid-based whole genome haplotyping of a HapMap trio child: evaluation of single individual haplotyping techniques. *Nucleic Acids Research*, 40: 2041–2053, 2012.

[42] J. Eberwine, J.-Y. Sul, T. Bartfai, and J. Kim. The promise of single-cell sequencing. *Nature methods*, 11(1):25–27, 2014.

[43] P. Eirew et al. Dynamics of genomic clones in breast cancer patient xenografts at single-cell resolution. *Nature*, 518(7539):422–426, 2015.

[44] M. El-Kebir, L. Oesper, H. Acheson-Field, and B. J. Raphael. Reconstruction of clonal trees and tumor composition from multi-sample sequencing data. *Bioinformatics*, 31 (12):i62–i70, 2015.

[45] M. El-Kebir, B. J. Raphael, R. Shamir, R. Sharan, S. Zaccaria, M. Zehavi, and R. Zeira. Copy-number evolution problems: Complexity and algorithms. In *International Workshop on Algorithms in Bioinformatics (WABI)*, pages 137–149. Springer, 2016.

[46] M. El-Kebir, G. Satas, L. Oesper, and B. J. Raphael. Inferring the mutational history of a tumor using multi-state perfect phylogeny mixtures. *Cell Systems*, 3(1):43–53, 2016.

[47] M. El-Kebir et al. Reconstruction of clonal trees and tumor composition from multi-sample sequencing data. *Bioinformatics*, 31(12):i62–i70, 2015.

[48] N. Eriksson, L. Pachter, Y. Mitsuya, S.-Y. Rhee, C. Wang, B. Gharizadeh, M. Ronaghi, R. W. Shafer, and N. Beerenwinkel. Viral population estimation using pyrosequencing. *PLoS Comput Biol*, 4(5):e1000074, 05 2008.

[49] W. J. Faison, A. Rostovtsev, E. Castro-Nallar, K. A. Crandall, K. Chumakov, V. Simonyan, and R. Mazumder. Whole genome single-nucleotide variation profile-based phylogenetic tree building methods for analysis of viral, bacterial and human genomes. *Genomics*, 104(1):1–7, 2014.

[50] J. S. Farris. Parsimony and explanatory power. *Cladistics*, 24(5):825–847, 2008.

[51] J. Felsenstein. The number of evolutionary trees. *Systematic Biology*, 27(1):27–33, 1978.

[52] J. Felsenstein. Evolutionary trees from dna sequences: a maximum likelihood approach. *Journal of Molecular Evolution*, 17(6):368–376, 1981.

[53] J. Felsenstein. *Inferring Phylogenies*. Macmillan Education, 2004.

[54] A. Fischer, I. Vázquez-García, C. J. Illingworth, and V. Mustonen. High-definition reconstruction of clonal composition in cancer. *Cell reports*, 7(5):1740–1752, 2014.

[55] R. Fisher et al. Cancer heterogeneity: implications for targeted therapeutics. *Brit J Cancer*, 108(3):479–485, 2013.

[56] W. M. Fitch. Toward defining the course of evolution: minimum change for a specific tree topology. *Systematic Biology*, 20(4):406–416, 1971.

[57] P. Fouilhoux and A. Mahjoub. Solving VLSI design and DNA sequencing problems using bipartization of graphs. *Computational Optimization and Applications*, 51(2): 749–781, 2012.

[58] L. R. Foulds and R. L. Graham. The Steiner problem in phylogeny is NP-complete. *Adv Appl Math*, 3:43–49, 1982.

[59] D. R. Fulkerson. Note on Dilworth's decomposition theorem for partially ordered sets. *Proc. American Mathematical Society*, 7:701–702, 1956.

[60] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[61] N. Garg, V. V. Vazirani, and M. Yannakakis. Approximate max-flow min-(multi) cut theorems and their applications. *SIAM Journal on Computing*, 25(2):235–251, 1996.

[62] F. Geraci. A comparison of several algorithms for the single individual SNP haplotyping reconstruction problem. *Bioinformatics*, 26(18):2217–2225, 2010.

[63] M. Gerlinger et al. Intratumor heterogeneity and branched evolution revealed by multiregion sequencing. *N Engl J Med*, 366(10):883–92, Mar 2012.

[64] C. Gonzaga-Jauregui, J. R. Lupski, and R. A. Gibbs. Human genome sequencing in health and disease. *Annual review of medicine*, 63:35, 2012.

[65] H. J. Greenberg, W. E. Hart, and G. Lancia. Opportunities for combinatorial optimization in computational biology. *INFORMS J. on Computing*, 16(3):211–231, 2004.

[66] S. Guindon and O. Gascuel. A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. *Systematic biology*, 52(5):696–704, 2003.

[67] G. Gundem et al. The evolutionary history of lethal metastatic prostate cancer. *Nature*, 520(7547):353–357, 2015.

[68] J. Guo, J. Gramm, F. Hüffner, R. Niedermeier, and S. Wernicke. Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. *J. Comput. Syst. Sci.*, 72(8):1386–1396, 2006.

[69] Gurobi Optimization team. Gurobi optimizer. www.gurobi.com, 2016.

[70] D. Gusfield. *ReCombinatorics: The Algorithmics of Ancestral Recombination Graphs and Explicit Phylogenetic Networks*. MIT Press, 2014.

[71] G. Ha, A. Roth, J. Khattra, J. Ho, D. Yap, L. M. Prentice, N. Melnyk, A. McPherson, A. Bashashati, E. Laks, J. Biele, J. Ding, A. Le, J. Rosner, K. Shumansky, M. A. Marra, C. B. Gilks, D. G. Huntsman, J. N. McAlpine, S. Aparicio, and S. P. Shah. Titan: Inference of copy number architectures in clonal cell populations from tumor whole genome sequence data. *Genome Research*, 2014.

[72] B. V. Halldórsson, D. Aguiar, and S. Istrail. Haplotype phasing by multi-assembly of shared haplotypes: Phase-dependent interactions between rare variants. In *PSB*, pages 88–99. World Scientific Publishing, 2011.

[73] D. He, A. Choi, K. Pipatsrisawat, A. Darwiche, and E. Eskin. Optimal algorithms for haplotype assembly from whole-genome sequence data. *Bioinformatics*, 26(12): i183–i190, 2010.

[74] D. He, B. Han, and E. Eskin. Hap-seq: an optimal algorithm for haplotype phasing with imputation using sequencing data. *Journal of Computational Biology*, 20(2): 80–92, 2013.

[75] IBM Corp. IBM ILOG CPLEX optimization studio. www.cplex.com, 2016.

[76] M. Jain, I. T. Fiddes, K. H. Miga, H. E. Olsen, B. Paten, and M. Akeson. Improved data analysis for the minion nanopore sequencer. *Nature methods*, 12:351–356, 2015.

[77] M. Järvisalo, D. Le Berre, O. Roussel, and L. Simon. The international sat solver competitions. *AI Magazine*, 33(1):89–92, 2012.

[78] Y. Jiang, Y. Qiu, A. J. Minn, and N. R. Zhang. Assessing intratumor heterogeneity and tracking longitudinal and spatial clonal evolutionary history by next-generation sequencing. *Proceedings of the National Academy of Sciences*, 113(37):E5528–E5537, 2016.

[79] W. Jiao et al. Inferring clonal evolution of tumors from single nucleotide somatic mutations. *BMC Bioinformatics*, 15, 2014.

[80] Y. Jiao, J. Xu, and M. Li. On the k-closest substring and k-consensus pattern problems. In *CPM*, volume 3109 of *LNCS*, pages 130–144, 2004.

[81] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[82] W. J. Kent. Blat—the blast-like alignment tool. *Genome Research*, 12(4):656–664, 2002.

[83] S. Khot. On the power of unique 2-prover 1-round games. In *STOC*, pages 767–775. ACM, 2002.

[84] E. Khurana, Y. Fu, D. Chakravarty, F. Demichelis, M. A. Rubin, and M. Gerstein. Role of non-coding sequence variants in cancer. *Nature Reviews Genetics*, 17(2):93–108, 2016.

[85] J. Kleinberg, C. Papadimitriou, and P. Raghavan. Segmentation problems. In *STOC*, pages 473–482. ACM, 1998.

[86] D. E. Knuth. *The Art of Computer Programming*, volume 4. Addison-Wesley, 2005.

[87] V. Kuleshov. Probabilistic single-individual haplotyping. *Bioinformatics*, 30(17): i379–i385, 2014.

[88] V. Kuleshov et al. Whole-genome haplotyping using long reads and statistical methods. *Nature Biotechnology*, 32(3):261, 266, 2014.

[89] G. Lancia, V. Bafna, S. Istrail, R. Lippert, and R. Schwartz. SNPs problems, complexity, and algorithms. In *ESA*, volume 2161 of *LNCS*, pages 182–193, 2001.

[90] E. Landau. *Handbuch der Lehre von der Verteilung der Primzahlen*, volume 1. Teubner, 2000.

[91] T.-H. Lee, H. Guo, X. Wang, C. Kim, and A. H. Paterson. Snphylo: a pipeline to construct a phylogenetic tree from huge snp data. *BMC Genomics*, 15(1):1, 2014.

[92] P. Leekitcharoenphon, R. S. Kaas, M. C. F. Thomsen, C. Friis, S. Rasmussen, and F. M. Aarestrup. snptree-a web-server to identify and construct snp trees from whole genome sequence data. *BMC genomics*, 13(7):1, 2012.

[93] L. A. Levin. Universal sequential search problems. *Problemy Peredachi Informatsii*, 9(3):115–116, 1973.

[94] H. Li and N. Homer. A survey of sequence alignment algorithms for next-generation sequencing. *Briefings in bioinformatics*, 11(5):473–483, 2010.

[95] Y. Li, S. Zhou, D. C. Schwartz, and J. Ma. Allele-Specific Quantification of Structural Variations in Cancer Genomes. *Cell Systems*, 3(1):21–34, July 2016.

[96] J. T. Linderoth and T. K. Ralphs. Noncommercial software for mixed-integer linear programming. *Integer programming: theory and practice*, 3:253–303, 2005.

[97] R. Lippert, R. Schwartz, G. Lancia, and S. Istrail. Algorithmic strategies for the single nucleotide polymorphism haplotype assembly problem. *Briefings in Bioinformatics*, 3(1):23–31, 2002.

[98] J. Ma, A. Ratan, B. J. Raney, B. B. Suh, W. Miller, and D. Haussler. The infinite sites model of genome evolution. *Proceedings of the National Academy of Sciences of the United States of America*, 105(38):14254–14261, Sept. 2008.

[99] S. Malikic, A. W. McPherson, N. Donmez, and C. S. Sahinalp. Clonality inference in multiple tumor samples using phylogeny. *Bioinformatics*, 31(9):1349–1356, 2015.

[100] S. Malikic et al. Clonality inference in multiple tumor samples using phylogeny. *Bioinformatics*, 2015.

[101] S. Mangiola, M. K. Hong, M. Cmero, N. Kurganovs, A. Ryan, A. J. Costello, N. M. Corcoran, G. Macintyre, and C. M. Hovens. Comparing nodal versus bony metastatic spread using tumour phylogenies. *Scientific Reports*, 6, 2016.

[102] E. R. Mardis. The impact of next-generation sequencing technology on genetics. *Trends in genetics*, 24(3):133–141, 2008.

[103] A. McPherson, A. Roth, C. Chauve, and S. C. Sahinalp. Joint inference of genome structure and content in heterogeneous tumor samples. In *International Conference on Research in Computational Molecular Biology*, pages 256–258. Springer, 2015.

[104] A. McPherson, A. Roth, E. Laks, T. Masud, A. Bashashati, A. W. Zhang, G. Ha, J. Biele, D. Yap, A. Wan, et al. Divergent modes of clonal spread and intraperitoneal mixing in high-grade serous ovarian cancer. *Nature Genetics*, 2016.

[105] J. a. Meidanis, O. Porto, and G. P. Telles. On the consecutive ones property. *Discrete Applied Mathematics*, 88(1—3):325–354, 1998.

[106] M. L. Metzker. Sequencing technologies—the next generation. *Nature reviews genetics*, 11(1):31–46, 2010.

[107] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient sat solver. In *Proceedings of the 38th annual Design Automation Conference*, pages 530–535. ACM, 2001.

[108] N. Navin, J. Kendall, J. Troge, P. Andrews, L. Rodgers, J. McIndoo, K. Cook, A. Stepansky, D. Levy, D. Esposito, et al. Tumour evolution inferred by single-cell sequencing. *Nature*, 472(7341):90–94, 2011.

[109] N. E. Navin. Delineating cancer evolution with single-cell sequencing. *Science Translational Medicine*, 7(296), 2015.

[110] N. E. Navin. The first five years of single-cell cancer genomics and beyond. *Genome Research*, 25(10):1499–1507, 2015.

[111] T. Nawy. Single-cell sequencing. *Nature methods*, 11(1):18–18, 2014.

[112] R. Nielsen, J. S. Paul, A. Albrechtsen, and Y. S. Song. Genotype and SNP calling from next-generation sequencing data. *Nature Reviews Genetics*, 12(6):443–451, 2011.

[113] S. Nik-Zainal et al. The life history of 21 breast cancers. *Cell*, 149(5):994–1007, May 2012.

[114] P. C. Nowell. The clonal evolution of tumor cell populations. *Science*, 194, 1976.

[115] L. Oesper, A. Ritz, S. J. Aerni, R. Drebin, and B. J. Raphael. Reconstructing cancer genomes from paired-end sequencing data. *BMC Bioinformatics*, 13(6):S10, 2012.

[116] L. Oesper, A. Mahmoody, and B. J. Raphael. Theta: inferring intra-tumor heterogeneity from high-throughput dna sequencing data. *Genome Biology*, 14(7):R80, 2013.

[117] R. Ostrovsky and Y. Rabani. Polynomial-time approximation schemes for geometric min-sum median clustering. *J. ACM*, 49(2):139–156, 2002.

[118] C. H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1982.

[119] C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *J. Comput. Syst. Sci.*, 43(3):425–440, 1991.

[120] M. Patterson, T. Marschall, N. Pisanti, L. van Iersel, L. Stougie, G. W. Klau, and A. Schönhuth. WhatsHap: Haplotype assembly for future-generation sequencing reads. In *Research in Computational Molecular Biology (RECOMB)*, volume 8394 of *LNCS*, pages 237–249, 2014.

[121] M. Patterson, T. Marschall, N. Pisanti, L. van Iersel, L. Stougie, G. W. Klau, and A. Schönhuth. WhatsHap: Weighted haplotype assembly for future-generation sequencing reads. *J. of Computational Biology*, 6(1):498–509, 2015.

[122] G. A. Pavlopoulos, A. Oulas, E. Iacucci, A. Sifrim, Y. Moreau, R. Schneider, J. Aerts, and I. Iliopoulos. Unraveling genomic variation from next generation sequencing data. *BioData Mining*, 6(1):13, 2013.

[123] Y. Pirola, P. Bonizzoni, and T. Jiang. An efficient algorithm for haplotype inference on pedigrees with recombinations and mutations. *IEEE/ACM Trans. Comput. Biology Bioinform.*, 9(1):12–25, 2012.

[124] Y. Pirola, G. Della Vedova, P. Bonizzoni, A. Stella, and F. Biscarini. Haplotype-based prediction of gene alleles using pedigrees and SNP genotypes. In *ACM-BCB*, pages 33–41, 2013.

[125] Y. Pirola, S. Zaccaria, R. Dondi, G. W. Klau, N. Pisanti, and P. Bonizzoni. HapCol: accurate and memory-efficient haplotype assembly from long reads. *Bioinformatics*, 32(11):1610, 2015.

[126] Y. Pirola, S. Zaccaria, R. Dondi, G. W. Klau, N. Pisanti, and P. Bonizzoni. HapCol. http://hapcol.algolab.eu/, 2015.

[127] J. E. Pool, I. Hellmann, J. D. Jensen, and R. Nielsen. Population genetic inference from genomic sequence variation. *Genome Research*, 20(3):291–300, 2010.

[128] V. Popic et al. Fast and scalable inference of multi-sample cancer lineages. *Genome Biol*, 16:91, 2015.

[129] B. J. Raphael, J. R. Dobson, L. Oesper, and F. Vandin. Identifying driver mutations in sequenced cancer genomes: computational approaches to enable precision medicine. *Genome medicine*, 6(1):1, 2014.

[130] B. Reed, K. Smith, and A. Vetta. Finding odd cycle transversals. *Oper. Res. Lett.*, 32 (4):299–301, 2004.

[131] R. Rizzi, A. Tomescu, and V. Mäkinen. On the complexity of minimum path cover with subpath constraints for multi-assembly. *BMC Bioinformatics*, 15(S9):S5, 2014.

[132] R. J. Roberts, M. O. Carneiro, and M. C. Schatz. The advantages of SMRT sequencing. *Genome Biol*, 14(6):405, 2013.

[133] D. F. Robinson and L. R. Foulds. Comparison of phylogenetic trees. *Math Biosci*, 53: 131–147, 1981.

[134] T. Roman, L. Xie, and R. Schwartz. Medoidshift clustering applied to genomic bulk tumor data. *BMC Genomics*, 17(1):6, 2016.

[135] R. Schwarz et al. Phylogenetic quantification of intra-tumour heterogeneity. *PLoS Comput Biol*, 10(4), 2014.

[136] R. F. Schwarz, A. Trinh, B. Sipos, J. D. Brenton, N. Goldman, and F. Markowetz. Phylogenetic quantification of intra-tumour heterogeneity. *PLoS Comput Biol*, 10(4): 1–11, 04 2014.

[137] R. F. Schwarz, A. Trinh, B. Sipos, J. D. Brenton, N. Goldman, and F. Markowetz. Phylogenetic Quantification of Intra-tumour Heterogeneity. *PLoS Computational Biology*, 10(4):e1003535, Apr. 2014.

[138] R. F. Schwarz, C. K. Y. Ng, S. L. Cooke, S. Newman, J. Temple, A. M. Piskorz, D. Gale, K. Sayal, M. Murtaza, P. J. Baldwin, N. Rosenfeld, H. M. Earl, E. Sala, M. Jimenez-Linan, C. A. Parkinson, F. Markowetz, and J. D. Brenton. Spatial and temporal heterogeneity in high-grade serous ovarian cancer: A phylogenetic analysis. *PLoS Med*, 12(2):1–20, 02 2015.

[139] R. Shamir, M. Zehavi, and R. Zeira. A linear-time algorithm for the copy number transformation problem. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 54. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[140] J. Shendure and H. Ji. Next-generation dna sequencing. *Nature biotechnology*, 26 (10):1135–1145, 2008.

[141] C. C. Smith et al. Validation of ITD mutations in FLT3 as a therapeutic target in human acute myeloid leukaemia. *Nature*, 485(7397):260–263, 2012.

[142] L. Song and L. Florea. CLASS: constrained transcript assembly of RNA-seq reads. *BMC Bioinformatics*, 14(5):1–8, 2013.

[143] A. Sottoriva et al. A Big Bang model of human colorectal tumor growth. *Nat Genet*, 47(3):209–216, 2015.

[144] D. T. Suzuki, A. J. Griffiths, J. H. Miller, R. C. Lewontin, et al. *An Introduction to Genetic Analysis, 7th edition*. WH Freeman and Company, 2000.

[145] K. Tamura, J. Dudley, M. Nei, and S. Kumar. Mega4: molecular evolutionary genetics analysis (mega) software version 4.0. *Molecular biology and evolution*, 24(8):1596–1599, 2007.

[146] J. A. Tennessen, A. W. Bigham, et al. Evolution and functional impact of rare coding variation from deep sequencing of human exomes. *Science*, 337(6090):64–69, 2012.

[147] C. Trapnell, B. A. Williams, G. Pertea, A. Mortazavi, G. Kwan, M. J. van Baren, S. L. Salzberg, B. J. Wold, and L. Pachter. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nature Biotechnology*, 28(5):516–520, 2010.

[148] P. Van Loo, S. H. Nordgard, O. C. Lingjærde, H. G. Russnes, I. H. Rye, W. Sun, V. J. Weigman, P. Marynen, A. Zetterberg, B. Naume, C. M. Perou, A.-L. Børresen-Dale, and V. N. Kristensen. Allele-specific copy number analysis of tumors. *Proceedings of the National Academy of Sciences*, 107(39):16910–16915, 2010.

[149] S. Venkatesan and C. Swanton. Tumor evolutionary principles: How intratumor heterogeneity influences cancer treatment and outcome. In *American Society of Clinical Oncology educational book / ASCO. American Society of Clinical Oncology. Meeting*, volume 35, pages e141–9, 2015.

[150] R.-S. Wang, L.-Y. Wu, Z.-P. Li, and X.-S. Zhang. Haplotype reconstruction from snp fragments by minimum error correction. *Bioinformatics*, 21(10):2456–2462, 2005.

[151] Y. Wang et al. Clonal evolution in breast cancer revealed by single nucleus genome sequencing. *Nature*, 512(7513):155–60, 2014.

[152] D. A. Wheeler, M. Srinivasan, M. Egholm, Y. Shen, L. Chen, A. McGuire, W. He, Y.-J. Chen, V. Makhijani, G. T. Roth, et al. The complete genome of an individual by massively parallel dna sequencing. *nature*, 452(7189):872–876, 2008.

[153] R. B. Williams, E. K. Chan, M. J. Cowley, and P. F. Little. The influence of genetic variation on gene expression. *Genome Research*, 17(12):1707–1716, 2007.

[154] M. Yannakakis. Node-and edge-deletion NP-complete problems. In *Proc. of Symp. Theory of Computing (STOC)*, pages 253–264. ACM, 1978.

[155] K. Yuan et al. BitPhylogeny: a probabilistic framework for reconstructing intra-tumor phylogenies. *Genome Biol*, 16, 2015.

[156] T. I. Zack, S. E. Schumacher, S. L. Carter, A. D. Cherniack, G. Saksena, B. Tabak, M. S. Lawrence, C.-Z. Zhang, J. Wala, C. H. Mermel, et al. Pan-cancer patterns of somatic copy number alteration. *Nature genetics*, 45(10):1134–1140, 2013.

[157] C. Z. Zhang et al. Calibrating genomic and allelic coverage bias in single-cell sequencing. *Nature Communications*, 6:6822, 2015.

[158] Y.-Y. Zhao, L.-Y. Wu, J.-H. Zhang, R.-S. Wang, and X.-S. Zhang. Haplotype assembly from aligned weighted SNP fragments. *Computational Biology and Chemistry*, 29: 281–287, 2005.

Acknowledgments

Over the past three years of my PhD program, the passion that I have for research in science grew constantly. Especially because I had the pleasure to meet several brilliant and supportive people who helped me to understand that science is a lot of fun!

I start by thanking my first advisor Paola Bonizzoni. She guided me for many years along my entire academic studies until this point. I would have never been in the exciting world of computational biology without you and I will always keep your teachings with me. In addition to Paola, I am extremely grateful to Yuri and Riccardo who worked with me on several projects in Milan and taught me so much. I was happy to work also with other talented people in the same group, such as Gianluca, Anna Paola, and Murray. Furthermore, I was really happy to share the majority of my time in Milan, between work and relax, with my good friends Marco, Luca, and Stefano. Also, I would like to thank my tutor, Alberto Leporati, that nicely supervised the activities of my PhD program.

I want to thank my second advisor Ben Raphael. He gave to me the wonderful possibility to work with him in a high-level context, first at Brown University and then at Princeton University. He introduced me to the challenging problems in cancer genomics, and he guided me through the exciting study of computational biology from a point of view very close to the biological applications. I will always remember your advice. Next, I want to thank Mohammed. He is both a great friend and a brilliant colleague. It was exciting to work together on many projects sharing the same passion for science, and I am extremely grateful for all your teachings. Lastly, I want to thank all the people with whom I had a great time both in the extraordinary group of Ben, Ashley, Gryte, Rebecca, Matt, Max, Hsin-ta, and Dora, and in my intense period at Princeton, Bianca, Li-Fang, Roberta, Derek, and especially my roommates Karoline and Claire.

Gunnar Klau has not been my official advisor, but I consider him as one of them since we worked together during my entire PhD program, both in Milan and at Brown University. I'll always remember how much fun we had while sharing the same office at Brown. I am very grateful that we worked together on many projects and I will always bring with me your suggestions.

Clearly, my intense commitment to the work was mainly possible thanks to the constant support and the never-ending love I received from my family and my friends. They shared with me times of happiness and difficulty. As such, I want to sincerely thank my parents Antonio and Flavia, my sister Alissa, my girlfriend Anna, and all my friends. No work in this thesis would have certainly existed without all of you, and I hope I will continue to benefit of your support and help in the future.

Publications

E. Willing, S. Zaccaria, M. D. V. Braga, and J. Stoye. On the inversion-indel distance. *BMC Bioinformatics*, 14(S-15):S3, 2013.

P. Bonizzoni, R. Dondi, G. W. Klau, Y. Pirola, N. Pisanti, and S. Zaccaria*. On the fixed parameter tractability and approximability of the minimum error correction problem. In *26th Annual Symposium on Combinatorial Pattern Matching (CPM)*, volume 9133 of *LNCS*, pages 100–113, 2015.

Y. Pirola†, S. Zaccaria†, R. Dondi, G. W. Klau, N. Pisanti, and P. Bonizzoni. HapCol: accurate and memory-efficient haplotype assembly from long reads. *Bioinformatics*, 32(11): 1610, 2015.

P. Bonizzoni, R. Dondi, G. W. Klau, Y. Pirola, N. Pisanti, and S. Zaccaria*. On the minimum error correction problem for haplotype assembly in diploid and polyploid genomes. *Journal of Computational Biology*, 23(9):718–736, 2016.

M. El-Kebir, B. J. Raphael, R. Shamir, R. Sharan, S. Zaccaria, M. Zehavi, and R. Zeira*. Copy-number evolution problems: Complexity and algorithms. In *International Workshop on Algorithms in Bioinformatics (WABI)*, pages 137–149. Springer, 2016.

M. El-Kebir, B. J. Raphael, R. Shamir, R. Sharan, S. Zaccaria, M. Zehavi, and R. Zeira*. Complexity and algorithms for copy-number evolution problems. *Algorithms for Molecular Biology*, 2016, in press.

S. Zaccaria†, M. El-Kebir†, G. Klau, and B. J. Raphael. The copy-number tree mixture deconvolution problem and applications to multi-sample bulk sequencing tumor data. In *Research in Computational Molecular Biology (RECOMB)*, 2017, in press.

†joint first authorship

*The order of authors follows the traditional convention of publications in theoretical computer science where the authors are listed in alphabetical order and does not reflect the contribution

Biography

Simone Zaccaria was born on March 26th, 1989 in Alzano Lombardo, a small town close to Bergamo in Italy. He attended the Liceo Scientifico in Alzano from 2003 to 2008. In 2008, he started to study Computer Science at the University of Milano-Bicocca. He obtained a Bachelor's degree in 2011 with a thesis titled *Inferenza di aplotipi in pedigree multiallelici tramite risolutori SAT (Haplotype inference on multiallelic pedigree via SAT solvers)* under the supervision of prof. Paola Bonizzoni. He continued to study Computer Science at the University of Milano-Bicocca. Two years later in 2013, he obtained a Master's Degree in Computer Science with a thesis titled *Indel Reversal Distance* under the supervision of Prof. Jens Stoye and Prof. Paola Bonizzoni. This thesis resulted from a research period of six months that Simone spent at the Genome Informatics group headed by Prof. Jens Stoye at the Bielefeld University in Germany. Simone obtained both the degrees in Computer Science cum laude. In 2013, Simone started the PhD program in Computer Science at the University of Milano-Bicocca under the supervision of Prof. Paola Bonizzoni and winning a related grant. During this program, Simone was a visiting research fellow for a period of seven months at Brown University (Rhode Island, US) and for a period of two months at Princeton University (New Jersey, US) under the supervision of Prof. Ben Raphael that joined as his second PhD advisor.