Variants of the Consecutive Ones property: Algorithms, Complexity and Applications in Genomics

Ashok Rajaraman

PhD Thesis Proposal (MATH 879)
Department of Mathematics, Simon Fraser University

Summer 2013

Abstract

The Consecutive Ones property (C1P) is a well studied structural property concerning binary matrices, with important applications in genome assembly. While the original decision problem of detecting C1P matrices is solvable in polynomial time, decision and optimization problems on generalizations of the property are usually intractable. These generalizations are extremely important concepts used as models in genomics, and the limits of tractability of both decision and optimization problems concerning them is a pivotal question.

The following document examines generalizations of the C1P and tractability results for the same. These include fixed-parameter tractability, optimization problems on these variants, and some proposed new approaches to the original C1P problem, which we hope may lead to interesting results. It also discusses the application of the algorithms developed to genomic data sets.

Chapter 1

Introduction

In this chapter, we seek to motivate the topic of this proposal. To do so, we examine an important problem in genomics and genome biology, namely genome assembly, and briefly describe the mathematical model we use to address this problem.

1.1 Background

A fundamental contribution of mathematics and computer science to genomics and genome biology has been the development of methods to compute or infer the organization of genomes. Such methods range from physical map reconstruction algorithms [AKNW95, AKWZ95], to whole genome assembly algorithms [NP13].

In general, genome assembly[1] starts out with a highly fragmented assembly of a genome of interest, in the form of DNA segments such as contigs[2], genes, probes[3], evolutionary conserved blocks, etc. In this document, for the sake of exposition, we use the generic term of *markers* to denote such segments. An assembly can be extended (i.e. the fragmentation is reduced) using available contiguity information involving these segments, i.e. we are given subsets of these markers which encode the information that the markers are co-localized along a chromosome; such co-localization information, called *intervals* in this document, can originate from sequencing experiments (paired-reads, mate-pairs or long reads [NP13]) or comparative analysis [Far07, RMB$^+$09, JRTC12]. In some cases, we may also have copy number information for the markers, which is the estimated number of times a marker may appear over all chromosomes of the considered genome. The goal of genome assembly is to use this information to compute a linear (or, in the case of bacterial and organelle genomes, circular) order of the markers, which we call an *assembly*[4].

We can then distinguish two distinct processes: data acquisition, aimed at obtaining an initial set of fragmented data, through sequencing or bioinformatics approaches, and data processing, namely genome assembly, which is the main motivation of the research presented in this proposal.

From a computational point of view, we are interested in a family of genome assembly models that can account for the complex nature of the data at hand, while having some desirable properties in terms

[1] In this document, we use "genome assembly" as a generic term for the problem of ordering markers along the chromosome of a genome.

[2] Contigs are gapless DNA sequences obtained by extending short sequencing reads by overlapping.

[3] Probes are short DNA segments known to be present in a genome.

[4] There is a slight abuse of notation here. If the markers correspond to contigs, the order we seek to find is indeed called an assembly. If the markers correspond to probes or genes, the order is called a *physical map*.

of computational tractability. In the family of models we consider, the data can be represented as a binary matrix or a hypergraph, and an ordering (a permutation or a sequence) of the columns of the matrix, or of the vertices of the hypergraph, corresponds to an assembly.

To illustrate the general process described above, let us introduce the simplest model that we will consider, that was central in several physical mapping algorithms developed in the 1990s. We make the simplifying assumption that every marker occurs exactly once in the final assembly (*unique probes*, in physical mapping terminology). Under this assumption, we encode our data as a binary matrix M as follows.

1. Each column represents a marker.

2. Each row encodes one interval; if a marker appears in an interval, the entry corresponding to that row and column is set to 1, and is set to 0 otherwise.

Assuming that we want to assemble our markers into a linear genome, we want to compute a linear order of the markers, i.e. a permutation of the columns in the matrix, with each interval appearing as a consecutive substring in this permutation, i.e. the 1's in each row will occur consecutively. The existence of such an order corresponds to a special structural property on the matrix, which is called the *Consecutive Ones Property* (C1P). This model and property were first introduced by Fulkerson and Gross [FG64] for studying the fine structure of a gene, based on work by Benzer [Ben59]. Since then, it has been used in algorithms to compute physical maps[AKNW95, AKWZ95] and, more recently, in palaeogenomics[5] and the reconstruction of ancestral genome maps [CT08], which is the main applied motivation of the research described in this proposal, and will be described in more detail in Chapter 5.

The basis of the palaeogenome assembly problems we consider in the applied part of our work is that the order of ancestral markers can be deduced from the order of the corresponding markers in extant genomes. In a nutshell, we group ancestral and extant markers in families and detect groups of markers whose co-localization is conserved in pairs of extant species, pointing towards potential ancestral intervals. This data acquisition phase is followed by an assembly phase that follows the principles described above. We refer you to [CT08, RTC13] for more details.

1.2 Overview of mathematical and computational problems

Under the assumption that every marker occurs exactly once, the C1P has been used for physical mapping and ancestral genome map reconstruction. As we will describe in more detail in chapter 2, limited tractability results exist regarding the C1P, basically limited to deciding if a binary matrix satisfies the C1P. Most natural combinatorial optimization problems related to the C1P are known to be NP-hard [Dom09, Dom08].

While the unique markers assumption does not prevent the reconstruction of the organization of palaeogenomes when relying on a limited number of extant species [CT08, CGOT10], this becomes a problem when more extant genomes are considered [GCST11], or more generally, when the resolution of markers is lowered, which leads to the inclusion of *repeats families* in the markers [RTC13]. However, accounting for repeated markers, i.e. markers that can appear more then once in an assembly, is well known for being one of the major problems in genome assembly [TS12].

We are interested in how far we can extend the basic C1P model in order to incorporate the notion of repeated segments, striking a balance between working within a model that incorporates this notion and designing algorithms within this model that are efficient enough to process real data.

[5] Palaeogenomics is the study of ancient genomes.

The type of computational problems we will investigate fall into two broad categories.

1. (Decision results) Given an instance (a matrix, with other input, such as copy numbers, if necessary), can we decide efficiently whether it is C1P, or more generally, if it satisfies a variant of the C1P?

2. (Optimization results) Given an instance, which is known to be not C1P (or a variant), can we discard at most $k > 0$ rows in the matrix so as to get an instance of the C1P variant in question?

Until recently, these problems, when considering non-unique markers, had not been widely investigated, whether in relation with physical mapping algorithms [MS99, BI99], contig assembly problems, where the most popular theoretical models is centered on de Bruijn graphs [KT13], or scaffold assembly problems[6], where repeated contigs are usually discarded prior to the scaffolding [GSN11].

From a mathematical/computational point of view, our goals are to survey known results on both these problems, and extend them to more general cases that are relevant for practical genome assembly problems. We are particularly interested in tractability results in the form of efficient algorithms for both the decision and optimization problems that we shall define. In the absence of tractability results for a problem, we would like to show that no efficient algorithm exists for the problem at hand, followed by possible solutions to this, in the form of exponential algorithms, approximations, limited tractability results or heuristics.

From an applied point of view, we will develop two related lines of research:

- the extension of the software ANGES [JRTC12], for reconstructing ancestral genome maps, and FP-SAC [RTC13], for scaffolding ancient contigs;

- the application of these software for reconstructing ancestral genomes of important extinct organisms (fishes, *Yersinia pestis* bacteria, the plague causative agent), that will be described in more detail in Chapter 5.

[6]Scaffold assembly aims at ordering contigs and estimating the length of the gaps between consecutive contigs.

Chapter 2

Consecutive Ones- Variants and Results

This chapter lays out the mathematical formulation of the Consecutive Ones property, the variants we will be examining, and the problems that we are interested in. This includes some classical results of note, and recent developments regarding the variants. We also provide a short section within the definitions of the variants which motivates their study through the field of genomics. This chapter also serves to lay out the mathematical notation that will be used throughout this document.

2.1 Preliminaries and notation

The primary objects we shall work with are binary matrices. We use various languages to represent these matrices, hypergraphs and sequences being the most important ones. In the context of the binary matrices we study, we will usually find it convenient to denote the set of row indices and column indices of an $m \times n$ binary matrix M by $\mathcal{R}_M = \{0, \ldots, m-1\}$ and $\mathcal{C}_M = \{0, \ldots, n-1\}$ respectively. If the matrix under consideration is clear as per the context of the situation, we drop the subscripts, and refer to the set of row and column indices as $\mathcal{R}$ and $\mathcal{C}$ respectively. In general, we use the notation $[n]$ to denote the integers $\{0, 1, \ldots, n-1\}$. The (i, j)-entry of a matrix M is denoted by m_{ij}. Similarly, for any general matrix A (not necessarily binary), we use the lower case letter indexed by the row and column to denote individual entries, i.e. a_{ij}. Given a binary matrix M, we say the column j *supports* row i if $m_{ij} \neq 0$, and the *supported set* of the column j, denoted by $supp\,(j)$, is the set $\{i \in \mathcal{R} : m_{ij} = 1\}$. Given a set of columns $S \subseteq \mathcal{C}$, the supported set of S, is the set $supp\,(S) = \bigcup_{j \in S} supp\,(j)$. For a set of columns $S \subseteq \mathcal{C}$, the set of *common rows* of S is the set $\bigcap_{j \in S} supp\,(j)$.

Permutations of the column set of binary matrices will be objects of interest for us. In this context, we usually refer to permutations as elements of the symmetric group S_n, and we use π, σ to denote permutations. The notation M_π or M_σ denotes the matrix obtained by permuting the columns of M by π or σ respectively.

The following combinatorial interpretation of a binary matrix will be useful.

Definition 2.1. Given an $m \times n$ binary matrix M, we can define a hypergraph $H = (V, E)$, where the vertex set is $\{v_i \in V : i \in \mathcal{C}\}$, and a hyperedge $e_i = \{v_j \in V : m_{ij} = 1\}$ is associated to each row for $0 \leq i \leq m-1$.

Put another way, we have a base set on the alphabet $[n]$, and m subsets of this base set, which are exactly the rows of the matrix, represented as the set of columns in which they have an entry 1.

0	1	1	0	0
1	1	0	0	0
0	0	1	1	1
0	1	1	1	0

(a) Matrix with the C1P

0	1	1	0	0
1	1	0	0	0
0	0	1	1	1
1	0	1	1	0

(b) Matrix without the C1P

Figure 2.1: Examples illustrating the Consecutive Ones Property(C1P)

We now introduce some common terminology on hypergraphs that will be use in the document. Let $H = (V, E)$ be a hypergraph, and let $c\colon V \to \mathbb{N}$ be a function mapping the vertices to positive integers. Such a function is called a *multiplicity function*. We denote the maximum value of the function c over all vertices $v \in V$, i.e. $\max_{v \in V} c(v)$, by μ, and call this value the *maximum multiplicity*. A vertex $v \in V$ such that $c(v) = 1$ is called a *unique vertex*. A vertex that is not unique is called a *repeat vertex* or just a *repeat*. The set of repeats in the vertex set V is denoted by V_R, and $\rho = |V_R|$.

We use Δ to denote the value $\max_{e \in E} |e|$, the maximum size of a hyperedge (alternately, the maximum number of 1's over all the rows). In order to refer to the degree of a single vertex v in the hypergraph H, i.e. the number of edges that it is contained in, we use the notation $deg_H(v)$. If the hypergraph (graph) under consideration is unambiguous, we drop the subscript, and denote the degree of the vertex v by $deg(v)$. By δ, we denote the *maximum degree* of a vertex, i.e. $\delta = \max_{v \in V} deg(v)$. An edge of size 2 (size greater than 2) is called an *adjacency* (*interval*). An interval of size 3 is called a *triple*. The set of all adjacencies (intervals) in the hypergraph is denoted by E_A (E_I), and the graph induced by E_A is denoted by $H_A = (V, E_A)$.

The restriction of any function on the set of edges, $f\colon E \to R$, where R is an arbitrary range, to a subset $S \subseteq E$, is denoted by f_S. For example, the restriction of a weight function $w\colon E \to \mathbb{R}$ restricted to a set $S \subseteq E$ is given by w_S. The restriction of a function f on the edges to the set of adjacencies (intervals) is denoted by f_A (f_I), and the restriction of the function f to the set $E_A \cup S$ ($E_I \cup S$), where S is some subset of E, is denoted by $f_{A \cup S}$ ($f_{I \cup S}$).

2.2 The Consecutive Ones property

The original problem which we discuss is the Consecutive Ones property. As we will see, there are various relaxations of this definitions, which will form the bulk of the problems that we encounter.

Definition 2.2. [FG64] A binary matrix M is said to have the *Consecutive Ones property* (C1P), or alternately, is said to be C1P, if there exists a permutation π of the column set of M such that the 1 entries on each row appear consecutively under this permutation.

For example, consider Figure 2.1a. We see here a binary matrix whose columns are permuted in a manner that causes all the rows in the matrix to have consecutive 1's. On the other hand, Figure 2.1b is an example of a matrix for which no such permutation exists. As a rule, we do not differentiate properties of a binary matrix M with those of the matrices formed by permuting the columns of M.

The topics we discuss relate to variants of this property. Usually, we shall deal with generalizations, and tractability results for the same. We will also discuss certain optimization problems relating to the property. One of the better studied variants is the *circular ones property*.

Definition 2.3. An $m \times n$ binary matrix M is said to have the *circular ones property* (Ci1P) if there exists a permutation of the columns such that either the 1's or the 0's are consecutive in every row.

A Ci1P matrix can be visualized as a cylinder- we can wrap around the matrix so that the first and last column meet for every row, and then we can find consecutive blocks of 1's on every row.

2.2.1 Decision problems

In this section, we recap the known results on the classical C1P problem, including some new developments. It will be noticed that there is a striking dichotomy between decision and optimization problems.

Deciding the C1P was proved to be in P by Fulkerson and Gross [FG64], and later proved to be solvable in linear time by Booth and Leuker.

Theorem 2.1. *[BL76] Given an $m \times n$ binary matrix M, it is possible to test if M is C1P in time and space $O(n + m + s)$, where s is the number of non-zero entries in the matrix.*

Since then, there have been other polynomial time and space algorithms for the C1P [McC04, HMPV00, HG02, ABH98]. Booth and Leuker's algorithm is constructive, and creates a data structure which encodes all permutations of the columns that yield a C1P ordering. This data structure, called the PQ-tree, is now a standard in most algorithms for the problem.

Of future interest to us is a decision result for the C1P presented by Atkins' et al, in which they order the columns of the matrix using the eigenvectors of the associated Laplacian matrix (defined in Chapter 4).

Theorem 2.2. *[ABH98] Given an $m \times n$ binary matrix M, it is possible to test if M is C1P in time $O(T(n)(n + n \log n))$, where $T(n)$ is the complexity of computing the eigenvectors of a symmetric positive semi-definite matrix of size n.*

While the time complexity is considerably greater than Booth and Leuker's algorithm, the merit of this algorithm lies in that even if the matrix is not C1P, it outputs a PQ-tree. We shall take a closer look at this algorithm later in Chapter 4.

It would also be useful to have an efficient way to characterize non-C1P matrices. Formally, we want to solve the complement of the C1P problem, and decide which matrices are not C1P, with a short certificate proving this claim for a YES instance. The first step towards this was taken by Tucker, who provided a forbidden substructure characterization of C1P matrices [Tuc72]. Tucker did not provide an algorithm for finding these submatrices, though some algorithms have been devised since [CST12, DGN10]. A more useful and elegant certificate was devised by McConnell [McC04], who provided an $O(n)$ size certificate for non-C1P matrices.

The C1P decision problem, in its classical formulation, has been solved many times over, by different methods. In fact, many of these results also hold for the Ci1P.

Theorem 2.3. *[Tuc70] Given an $m \times n$ binary matrix M, M is Ci1P if and only if the matrix obtained by complementing all rows i such that $m_{i0} = 1$ is C1P.*

Once we have reduced the problem to a C1P testing instance, we can proceed via any of the various algorithms stated before.

2.2.2 Optimization problems

We discuss here the most obvious optimization problem for binary matrices.

Problem 2.1. Consecutive Ones Submatrix by Row deletions (MIN-ROW-C1P)

Let M be an $m \times n$ binary matrix which does not have the C1P, and let k be a positive integer. Is there a subset S of at most k rows such that the submatrix $M \setminus S$ has the C1P?

Optimization problems are of particular importance to us, since we rarely encounter problem instances that already have the C1P. Instead, it is more critical to optimize the data we have so that we end up with a C1P instance, without throwing away too much of the information obtained. Classically, the notion of 'too much' is either given by the cardinality of the set we are throwing away, or by a weight function on the set of rows. It is obviously preferable to deal with a weight function, since it is a more general instance of the problem.

The MIN-ROW-C1P problem was proved to be NP-hard, even for sparse matrices [Tuc72]. In fact, it is NP-hard even if we relax the C1P condition to the Ci1P condition. This can be proved by reducing the Hamiltonian cycle problem to it.

Dom et al. [DGN10] used Tucker's characterization to provide FPT and approximability results for the problem.

Theorem 2.4. *[DGN10] Let M be an $m \times n$ binary matrix.*

1. *If M has at most Δ 1's in a row, the MIN-ROW-C1P problem can be solved in time exponential in the number of row deletions k, with base depending only on Δ.*

2. *If M has at most Δ 1's in a row, the MIN-ROW-C1P problem can be approximated within a factor of at most $\Delta + 4$ if $\Delta \in \{2, 5, 6, \ldots\}$, and within a factor of at most 9 otherwise, in polynomial time.*

This result is a complex but clever use of Tucker's forbidden submatrix characterization of the C1P. The algorithm proceeds through recognizing these forbidden matrices, and then deleting them. In the approximation algorithm, every such matrix is found and deleted. For the fixed parameter result, when a pattern is found, a single row is deleted, and the algorithm branches on the row being deleted.

An important trend to note is that the optimization problems we consider always optimize the number of rows, and not the number of columns. This is also true of the other problems we will encounter in this document. In the context of application, this is because, from a biological standpoint, we usually do not discard markers once we have constructed the binary matrix, but we prefer to discard possibly faulty co-localization information. However, Dom et al.'s result also extends to column deletion scenarios.

2.3 Consecutive Ones with multiplicity

The C1P problem calls for the existence of a *permutation* of the columns such that the 1's on each row are consecutive. We can also look for a *sequence* of the columns that can satisfy the consecutivity condition for the rows. Naturally, we can always construct such a sequence by simply appending the content of the rows into a single, long string. To completely define the problem, we add a constraint in the form of a multiplicity function. Then, we make the following definition.

Definition 2.4. (Compatibility condition)

Let $H = (V, E)$ be a hypergraph, with a multiplicity function $c\colon V \to \mathbb{N}$. Given a set of linear/circular sequences $\mathcal{S}$, we say an edge is *compatible* with $\mathcal{S}$ if there exists a consecutive substring in some sequence $s \in \mathcal{S}$ which consists of exactly the vertices in e.

This is a consecutivity condition, as in the C1P problem. The modified property for sequences, which we will write out in terms of hypergraphs, is stated below.

Definition 2.5. Given a hypergraph $H = (V, E)$, and a multiplicity function $c\colon V \to \mathbb{N}$, we say the instance (H, c) has the *consecutive ones property with multiplicity* (alternately, (H, c) is mC1P) if there exists a set of linear sequences $\mathcal{S}$ on the alphabet V such that

1. for each $v \in V$, v occurs at least in one $s \in \mathcal{S}$, and the total number of occurrences of v does not exceed $c(v)$ over all sequences in $\mathcal{S}$, and

2. for each $e \in E$, e is compatible with $\mathcal{S}$.

If such a set of linear sequences $\mathcal{S}$ exists, this set is called a *certificate* for the instance. Assume that a given instance (H, c) is mC1P. Then, given a certificate $\mathcal{S}$, a convenient way of representing this set is as a 2-graph on the vertex set V. Formally,

Definition 2.6. Given an mC1P instance $(H = (V, E), c)$, and a set of linear sequences $\mathcal{S}$ which is a certificate for the instance, we define a *covering graph* $G = (V, E')$ to be the minimal simple graph such that, for any two consecutive elements uv in a sequence $s \in \mathcal{S}$, $u, v \in V$, there is an edge $(u, v) \in E'$.

By this definition, every sequence $s \in \mathcal{S}$ is a walk on the graph G, and the set of walks satisfies the properties that every vertex $v \in V$ is visited at least once and at most $c(v)$ times, and the vertices of every hyperedge $e \in E$ form a consecutive subwalk. Since we defined each edge as a row of our matrix, by requiring a consecutive subwalk on the vertices of the edge, we are forcing the columns involved in the row to be consecutive, thus satisfying a consecutive ones condition in the corresponding matrix.

As with the C1P, we can relax the mC1P condition to look for a single circular sequence in which every edge occurs as a consecutive substring. This is called the mCi1P, analogous to the Ci1P in Section 2.2. We also define the following relaxation of the mC1P and the mCi1P.

Definition 2.7. Given a hypergraph $H = (V, E)$, and a multiplicity function $c\colon V \to \mathbb{N}$, we say the instance (H, c) has the *component circular ones property with multiplicity* (alternately, (H, c) is COMP-mCi1P) if there exists a set of circular sequences $\mathcal{S}$ on the alphabet V such that

1. for each $v \in V$, v occurs at least in one $s \in \mathcal{S}$, and the total number of occurrences of v does not exceed $c(v)$, and

2. for each $e \in E$, e is compatible with $\mathcal{S}$.

The COMP-mCi1P is a somewhat weaker analogue of the Ci1P. Where the Ci1P specifies that there needs to exist a *single* circular permutation, the COMP-mCi1P asks for a set of circular sequences instead. The concept of a covering graph also extends to the COMP-mCi1P. If an instance (H, c) is mC1P, then, since we can concatenate the linear segments into a circular sequence without exceeding the multiplicity of any vertex, the instance is also COMP-mCi1P. This will prove to be a useful relaxation of the mC1P, especially in optimization problems.

The generalization to sequences is not arbitrary. Analyzing problems on sequences is a mainstay of computational complexity and algorithms, since they are such general objects.

A motivation through genomics While the C1P model is useful in ordering genes on a genome, it cannot take duplications into account. Generalizing it to the mC1P offers the relaxation needed to do this in a natural way, providing a mathematical formalization to the problem of assembling genomes with duplications. It is, thus, of substantial importance to define the limits of tractability for this problem. Furthermore, certain organisms (such as bacteria) may have circular chromosomes, as well as small DNA molecules separate from the chromosome, called plasmids. This motivates the definition of the COMP-mCi1P.

As we shall see in the succeeding subsections, there are a number of recent results concerning it.

2.3.1 Decision problems

Having defined the mC1P, the obvious problem would be to recognize instances that have the property. This motivates the problem stated below.

Problem 2.2. (mC1P)
Given a hypergraph $H = (V, E)$, and a multiplicity function $c\colon V \to \mathbb{N}$, determine if the instance (H, c) has the mC1P property.

As in the case of the C1P, an instance (H, c) having the mC1P also has the COMP-mCi1P. However, the following result, by Wittler et al., proves that, unlike the C1P, determining the mC1P for arbitrary instances is NP-hard, but it is tractable when all the hyperedges are adjacencies.

Theorem 2.5. *[WMPS11]*

1. *The problem $mC1P$ is solvable in polynomial (linear) time and space when $\Delta = 2$.*
2. *The problem mC1P is NP-hard in general for instances where $\mu \geq 2$ and $\Delta \geq 3$.*

Note that this is the strongest possible result. If $\mu = 1$, the case reduces to the classical C1P problem, which is solvable in linear time and space, as seen in Section 2.2. On the other hand, if $\Delta = 2$, the first part of the theorem proves that this case is solvable in linear time and space.

The proof of the first part of the theorem comes from a reduction of the problem to the problem of determining if a given graph is Eulerian. The second part of the theorem is proved by a polynomial time many-one reduction from an esoteric version of 3SAT to mC1P. This is done by introducing a number of vertices and hyperedges as gadgets for each clause and variable, with some multiplicity associated to the vertices. Then, it is proved that the CNF is satisfiable if and only if there exists an mC1P assembly of the resulting instance. In general, the model for intractability results consists of similar reductions, and they are usually non-trivial. A similar result, using different methods, was proved by Battaglia et al. [BGS12].

There are some special cases where the mC1P problem is still tractable on hypergraphs. The following theorem of Chauve et al. [CMPW11] is particularly interesting.

Theorem 2.6. *[CMPW11] Given a hypergraph $H = (V, E)$, and a multiplicity function $c\colon V \to \mathbb{N}$, the mC1P can be decided for the instance (H, c) if every $e \in E$ can be classified into one of the following categories.*

1. *For every $v \in e$, $c(v) = 1$.*
2. *e is an adjacency between two vertices u, v, and $c(u), c(v) > 1$.*
3. *There exists $u \in e$ such that $c(u) > 1$, for every $v \in e \setminus \{u\}$, $c(v) = 1$, and $e \setminus \{u\} \in E$ (matched multiedge condition).*

2.3.2 Optimization problems

We already noted that the problem MIN-ROW-C1P is NP-hard (Section 2.2.2). It is only natural that the problem MIN-ROW-mC1P is also NP-hard, since even the decision problem mC1P is NP-hard. In fact, the problem is NP-hard even when we restrict ourselves to a 2-graph with multiplicity 1.

One way to relax the condition in order to end up with a tractable optimization problem is to look for a set of circular sequences of the vertices instead of a set of linear sequences or a single circular sequence. This is where the notion of COMP-Ci1P comes into play, since the property enforces exactly this condition. Formally, we state the following problem.

Problem 2.3. (MAX-COMP-mCi1P)

Given a (weighted) hypergraph $H = (V, E)$ ($H = (V, E), w\colon E \to \mathbb{R}_{\geq 0}$) and a multiplicity function $c\colon V \to \mathbb{N}$, find a maximum cardinality (weight) subset $S \subseteq E$ such that the instance $(H' = (V, S), c)$ is COMP-mCi1P.

This is obviously a very important problem- given an instance that does not have the property, we need to find a maximum size (weight) subset of the instance which does exhibit the property. The hope was that this relaxation gives us enough room to design algorithms that solve it. A recent result by Manŭch et al. proves the following theorem, which, summarized, says that the relaxation only works for a graph consisting of adjacencies.

Theorem 2.7. *[MPW+13]*

1. *MAX-COMP-mCi1P can be decided on an instance (H, c) in polynomial time and space if $\Delta = 2$.*
2. *MAX-COMP-mCi1P is NP-hard for $\Delta \geq 3$ and $\mu \geq 1$.*

The polynomial time algorithm for the first result is an elegant reduction to a matching problem on a larger graph [DLG94]. Since maximum matching can be solved in $O\left(|V|^{1/2}|E|\right)$ for a general graph $G = (V, E)$ (possibly with weighted edges) [MV80], MAX-COMP-mCi1P can be decided on 2-graphs in time $O\left(n + m\right)^{3/2}$. The intractability result is again proved through a reduction from a 3SAT variant to a hypergraph problem. Most strikingly, this is a very tight result- the problem becomes NP-hard for $\Delta = 3$, even if the maximum multiplicity is 1.

The tractability result is of some significance where application is concerned. In the context of biology, this is a very useful tool for discarding possibly uninformative co-localization information. In fact, it has been applied to real data sets, with encouraging results [RTC13], and it is a key result that we shall use in Chapter 3. But it is significant that at the moment, we do not even have a heuristic algorithm for the intractable case. Dom's work [DGN10], as mentioned in Section 2.2, provides both approximation and fixed parameter algorithms for the case where $\mu = 1$, but for higher multiplicity, since checking for COMP-mCi1P is NP-hard in general, we have no known method to address the problem.

Finally, we note the following result of Cygan et al. [CMP+11], which proves that deleting edges to make a graph Eulerian can be done in time exponential only in the number of edges being deleted.

Theorem 2.8. *[CMP+11] Let $G = (V, E)$ be a graph, and let k be a positive integer. It is possible to decide if a set $S \subset E$ of cardinality k exists, such that the graph $G' = (V, E \setminus S)$ is Eulerian, in time $O\left(2^{k \log k} m^6 n^3 \log\left(k + k^2\right) \log m\right)$, where $|V| = n$ and $|E| = m$.*

This theorem important since we know that an instance (H, c), where H is a 2-graph, is mC1P if and only if we can construct a graph having an Eulerian cycle from this instance. While the theorem does not

have a multiplicity function to bound the number of times a vertex can be visited, it is still a significant direction to consider. This is also a result which sparks interest in *fixed parameter tractable* algorithms for mC1P type problems, which shall be discussed in Chapter 3.

2.4 Gapped Consecutive Ones

Another question one could ask is if there is a good measure to decide the 'C1Pness' of a given binary matrix. A good metric for this is the number of gaps separating consecutive series of 1's in the rows, and the size of these gaps. Let us first formally define gaps in binary matrices.

Definition 2.8. Let M be an $m \times n$ binary matrix. A *gap* in a row i is a contiguous set of columns $\mathcal{J} = \{j, \ldots, j+l-1\}$ such that $m_{ik} = 0$ for all $k \in \mathcal{J}$, and $m_{i,j-1} = m_{i,j+l} = 1$. The size of a gap $\mathcal{J}$ is the cardinality of the set $\mathcal{J}$, i.e. $|\mathcal{J}| = l$. The collection of all such sets $\mathcal{J}$ for a given row i is denoted by $\mathfrak{J}_i$, and its cardinality is called the *number of gaps* in the row i, denoted by $gaps_M(i)$. The cardinality of the largest set in $\mathfrak{J}_i$ for any row i is denoted by $size_M(i)$.

We can now define the following generalization of the C1P.

Definition 2.9. An $m \times n$ binary matrix M is (k, ℓ)-C1P for integers $k \geq 1$ and $\ell \geq 0$ if there exists a permutation π of the columns such that $max_{i \in \mathcal{R}}(gaps_{M_\pi}(i)) \leq k-1$ and $max_{i \in \mathcal{R}}(size_{M_\pi}(i)) \leq \ell$.

By this definition, a C1P matrix is $(1,0)$-C1P. The columns of a matrix which is $(2,1)$-C1P can be permuted in such a way that every row has at most 1 gap (or alternately, there are at most 2 consecutive runs of 1's in every row), and a gap consists of at most 1 column. The notation $(*, \ell)$-C1P ($(k, *)$-C1P) denotes that the columns of the matrix under consideration can be permuted such that the maximum size of a gap over all rows is at most ℓ, without any bound on the number of gaps (the maximum number of gaps is at most k, without any bound on the size of the gaps).

We can then define the following problem.

Problem 2.4. ((k, ℓ)-C1P)
Given an $m \times n$ binary matrix M, determine if M is (k, ℓ)-C1P for fixed k and ℓ.

A different, but related problem is the optimization problem of finding a permutation π of a matrix M which minimizes the total number of gaps over all the rows of the matrix.

Motivating the variant Optimizing the C1P or the mC1P is usually seen as an edge deletion problem, as mentioned before. This assumes that the edges discarded truly encode no information. However, the gapped C1P offers an alternate viewpoint, in which every edge is considered equally informative, but not necessarily accurate in encoding the containment information. This is particularly useful from an application point of view- the colocalization data we obtain for genome assembly may have small errors which prevent assembly. But discarding this data altogether might be too extreme a step, since most of it might be correct. In this respect, the notion of gaps in our matrix comes as a handy way to model the scenario.

2.4.1 Results

The most obvious polynomial time solvable case for the (k, ℓ)-C1P problem is the case $(1,0)$-C1P, which is the same as checking if the matrix is C1P. But past that point, this problem is quite hard to solve.

The first known result for the gapped problem was given by Goldberg et al. [GGKS95]. They proved that deciding if a matrix has a column permutation π such that it is $(k, *)$-C1P is NP-hard for all values of k. Later, Manŭch, Patterson and Chauve proved the following result.

Theorem 2.9. *[MPC12]*

1. *Given a binary matrix M, deciding if M is $(2, \ell)$-C1P is NP-hard for all values of $\ell \geq 2$.*
2. *Given a binary matrix M, deciding if M is $(k, 1)$-C1P is NP-hard for all values of $k \geq 3$.*

Of the two excluded cases, we mentioned before that $(1, 0)$-C1P is the C1P decision problem. However, deciding if a matrix is $(2, 1)$-C1P is a major open problem.

Question 2.1. [MPC12] What is the complexity of deciding the $(2, 1)$-C1P problem?

It is notable that the decision problems are NP-hard for fixed k and ℓ, which implies that all known algorithms are at least exponential in n. This means that the problem is not fixed parameter tractable in both these parameters. Also, to the best of the author's knowledge, no approximation algorithm for the problem is known.

However, Atkins et al.'s spectral algorithm [ABH98] for the C1P outputs a set of permutations when the matrix is not C1P. These permutations are not very well understood, since they do not represent C1P orderings of the columns (which do not exist). Vuokko proved that the output of the algorithm in the case of non-C1P hypergraphs is related to the total number of gaps. The precise result is stated as follows.

Theorem 2.10. *[Vuo10] Let M be an $m \times n$ binary matrix. Let P be a permutation matrix, and $\mathbf{S}$ be the discrete differential operator, and let $\hat{M}$ be the augmented matrix $\mathbf{0}M\mathbf{0}$, where $\mathbf{0}$ is a column of 0's. Let $\mathcal{L}$ be the Laplacian matrix associated to M, and let $\mathcal{L} = R\Lambda R^T$ be its eigenvalue decomposition, where $R = (\mathbf{r}_1 \dots \mathbf{r}_k)$, and Λ is a diagonal matrix with distinct eigenvalues of $\mathcal{L}$, $\lambda_1 < \dots < \lambda_k$ as entries. The Frobenius norm of the matrix $\|\mathbf{S}P\hat{M}^T\|_F$, which is given by*

$$\|\mathbf{S}P\hat{M}^T\|_F^2 = \sum_{i=1}^{k} (\alpha - \lambda_i) \|\mathbf{S}P\mathbf{r}_i\|^2,$$

where $\alpha > \max_i \lambda_i$ is a positive real number, is the total number of gaps in the matrix $\hat{M}$ when ordered by the permutation matrix P. Then, Atkins et al.'s spectral algorithm outputs a set of permutations that minimize the second term of this formula.

Roughly speaking, the spectral algorithm is a heuristic for minimizing the total number of gaps. From an algorithmic point of view, while the general framework of the algorithm is well understood, the algorithm has not been analyzed as an approximation for minimizing gaps in the matrix. We motivate this in Chapter 4.

2.5 A prefatory to the coming chapters

We just encountered some basic variants of the C1P. The rest of this proposal deals which the problems discussed in this chapter. There is no catch-all method that we use to address these problems; the results that we discuss and those that we hope to get fall into various categories- deterministic results, parameterized complexity, approximation algorithms etc. The breadth of the techniques being used is essential, since we are not merely dealing with a theoretical problem. The applications of C1P variants make any contribution to the theory behind it very interesting.

Chapter 3

Current developments on the mC1P

In this chapter, we shall recap some recent work on the mC1P. This consists of original work that will be included in the thesis, as well as the currently open avenues of research that will be taken up during the progress of the thesis.

Apart from the problems addressed in the previous chapter, one major question we try to answer here is the problem of getting a unique assembly in the presence of repeats. By casting it into an mC1P framework, we show limited tractability results for this problem. The methods have recently been used to assemble the genome of the plague bacterium which is believed to have caused the Black Death pandemic in Europe [RTC13].

3.1 Definitions and concepts

We introduced most of the main concepts for the mC1P in Sections 2.1 and 2.3. It will, however, be necessary to go through some more definitions before we can progress to the rest of the chapter. We also encourage you to refer to Appendix A for a primer on parameterized complexity and fixed parameter tractability.

Through this chapter, we shall be dealing with instances of the mC1P, given as a two tuple $(H = (V, E), c)$, where H is a hypergraph and $c\colon V \to \mathbb{N}$ is a multiplicity function. Depending on the problem (mC1P/COMP-mCi1P), we specify a *model*, asking for sets of linear or circular sequences. We shall also use the following functions.

1. A *weight function* $w\colon E \to \mathbb{R}_{\geq 0}$ on the set of edges. This shall be used in optimization problems. Instances accompanied by a weight function are said to be *weighted* instances.

2. An *order function* $o\colon E \to V^*$, where V^* is the set of all finite strings on the alphabet V. This is a novel concept that we shall use to obtain tractable instances for the mC1P problem. The order function is defined as follows: given an instance $(H = (V, E), c)$, an order function maps an edge $e \in E$ to one of the following.

   (a) $o(e) = \epsilon$, where ϵ is the empty string, or

   (b) $o(e) = v_0 \ \ldots \ v_{k-1}$, where $v_i \in e$ for all $0 \leq i \leq k-1$, and each $v \in e$ appears at least once and at most $c(v)$ times in the string $o(e)$.

   If e is an adjacency (i.e. $e \in E_A$), then we define $o(e) = \epsilon$. An instance equipped with an order function is said to be *ordered*.

The weight function is pretty standard in graph/hypergraph models. The order function is unique to the problems we address: it is an additional restriction on the sequences we are looking for. Formally, we can redefine the mC1P and COMP-mCi1P properties by defining a compatibility condition for ordered edges.

Definition 3.1. (Compatibility condition for ordered edges)

Let $H = (V, E)$ be a hypergraph, with a multiplicity function $c\colon V \to \mathbb{N}$ and an order function $o\colon E \to V^*$. Given a set of linear/circular sequences $\mathcal{S}$, we say an edge e, $o(e) \neq \epsilon$, is *compatible* with $\mathcal{S}$ if there exists a consecutive substring in some sequence $s \in \mathcal{S}$ which is equal to either $o(e)$ or its reverse.

Then, the mC1P (COMP-mCi1P) property for an instance $(H = (V, E), c, o)$ can be defined as before, by asking for a set of linear (circular) sequences in which every vertex $v \in V$ appears at least once and at most $c(v)$ times, and with which every edge is compatible. Thus, the order function specifies that the hyperedge should occur in some pre-specified order. Unless otherwise mentioned, all instances shall be taken to be unweighted and unordered.

We also define the following term.

Definition 3.2. Let $H = (V, E)$ be a hypergraph, and $c\colon V \to \mathbb{N}$ be a multiplicity function. A *maximal repeat cluster* is a connected component of the subhypergraph on the vertex set V_R, and edge set $\{e \cap V_R : e \in E\}$.

This shall be useful in one of the results we mention later.

3.2 Current progress

In this section we discuss three problems and partial results to these problems. Suggested extensions to these problems are also provided at the end of each subsection.

3.2.1 Extended decision results for mC1P

The first result addresses a particular case where we can decide the mC1P on instances provided with ordered intervals. We shall need the following definition to present this result.

Definition 3.3. (Repeat spanning interval)

Given a hypergraph $H = (V, E)$, a multiplicity function $c\colon V \to \mathbb{N}$, and an order function $o\colon E \to V^*$, an interval $e \in E_I$ is an ordered *repeat spanning interval* for a maximal repeat cluster R if $e = \{u, v, r_0, \ldots, r_{k-1}\}$, with $c(u) = c(v) = 1$, $\{r_0, \ldots, r_{k-1}\} \subseteq R$, and $o(e) = u.\ s\ .v$, where s is a sequence on the set $\{r_0, \ldots, r_{k-1}\}$. The subset of repeat spanning intervals contained in E_I is denoted by E_{rsi}.

The following result follows for the mC1P decision problem when the hyperedges are conditioned to have a certain structure.

Theorem 3.1. *[CPR13] (RSI-mC1P)*

Let $H = (V, E)$ be a hypergraph, with a multiplicity function $c\colon V \to \mathbb{N}$ and an order function $o\colon E \to V^*$. Assume that every edge $e \in E$ which contains a repeat falls into one of the following categories.

1. *e is an adjacency,*
2. *e is a matched multiedge (as defined in Theorem 2.6), or*

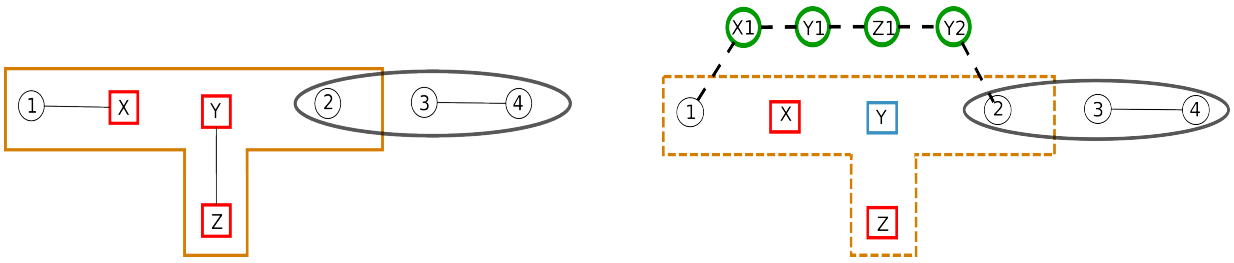

Figure 3.1: Dealing with ordered edges in Theorem 3.1: $e = \{1, X, Y, Z, 2\}, o(e) = 1\,X\,Y\,Z\,Y\,2$, with repeats X, Y, Z. A path is introduced for the edge, and the multiplicities of X and Z are decreased by 1, while that of Y is decreased by 2. The edge e itself is deleted.

3. *e is an ordered repeat spanning interval for some maximal repeat cluster in H.*

The mC1P problem can be decided for the instance (H, c, o) in polynomial time and space.

The proof of this result involves 'expanding' each repeat spanning interval $e \in E_I$ into a path consisting of copies of the vertex as they occur in $o(e)$, with each copy having unit multiplicity. This is illustrated in Figure 3.1. As a copy of the vertex v_i is introduced, the multiplicity of v_i is decreased by 1. Finally, we end up with an instance containing repeats in matched multiedges and adjacencies, which is the case addressed by Theorem 2.6, and can be decided in polynomial time and space.

A simple but important extension can be framed using the following concept.

Definition 3.4. Let $H = (V, E)$ be a hypergraph equipped with multiplicity and order functions $c\colon V \to \mathbb{N}$ and $o\colon E \to V^*$ respectively. An edge $e \in E$ is called a *repeat overlapping interval* if $o(e) = v_0\,.v_1\,\ldots\,.v_{k-1}$, where all v_i either belong to some maximal repeat cluster R, or they are unique vertices that share an edge with a vertex in R.

We can now ask if an instance consisting of repeat overlapping intervals can be decided to have the mC1P or not. Even a partial result for this instance would be very useful in genome assembly applications. Regarding this, we have the following conjecture.

Conjecture 3.1. *Let $H = (V, E)$ be a hypergraph equipped with multiplicity and order functions $c\colon V \to \mathbb{N}$ and $o\colon E \to V^*$ respectively. Assume that every edge $e \in E$ containing a repeat falls into one of the following categories.*

1. *e is an adjacency,*
2. *e is a matched multiedge, or*
3. *e is a repeat overlapping interval for some maximal cluster in H.*

The mC1P problem is decidable on the instance (H, c, o) in polynomial time.

This is a generalization of the previous theorem, and we suspect that it should not be too hard to extend the methods used to prove that to the present case. It remains to be seen if a stronger generalization exists, which would probably consist of relaxing the matched multiedge requirement, or allowing some intervals to be unordered.

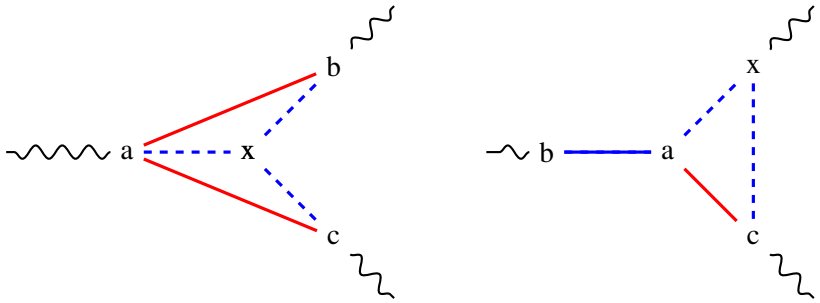

Figure 3.2: Graph modifications made by the algorithm for Theorem 3.2 on encountering triples $\{a, b, x\}$ and $\{a, c, x\}$. Here, x is a repeat, and a, b, c are unique vertices. The red edges are added in place of the triples, and the dotted blue edges are removed. In the second case, no red edge is added for $\{a, b, x\}$, since the edge (a, b) is already present.

3.2.2 Partial optimization

In Chapter 2, we saw that problem MAX-COMP-mCi1P was tractable for the case of 2-graphs. The algorithm proceeds by producing a matching in an auxiliary graph which defines a maximum weight COMP-mCi1P instance. Now we turn to the following problem.

Problem 3.1. (MAX-(k)INT-mCi1P)

Given a weighted hypergraph $H = (V, E), w\colon E \to \mathbb{R}_{\geq 0}$ and a multiplicity function $c\colon V \to \mathbb{N}$, such that the induced adjacency graph instance $(H_A = (V, E_A), c)$ is COMP-mCi1P, and $\Delta = k$, find a maximum weight subset $S \subseteq E_I$ such that the instance $(H' = (V, E_A \cup S), c, w_{A \cup S})$ is COMP-mCi1P.

Note here that we can easily find an adjacency graph $(H_A = (V, E_A), c)$ through Manŭch et al.'s algorithm [MPW$^+$13]. The question here is if we can add a set of larger hyperedges, and optimize solely on this set to get a COMP-mCi1P instance. This problem addresses a very important problem motivated through genome assembly: assuming that we have an instance which we know is COMP-mCi1P, but which may have more than one certificate, how do we determine which certificate is best suited for our needs? We refer you back to Chapter 1, in which we briefly stated how repeats have confused assembly methods. The presence of more than 1 certificate is a reason why current genome assembly techniques often fail. The problem above asks if we can choose a good set of large intervals to reduce the number of candidate certificates.

To start off, we shall define a notion of *almost compatible* hyperedges.

Definition 3.5. (Relaxed compatibility)

Let $H = (V, E)$ be a hypergraph, with a multiplicity function $c\colon V \to \mathbb{N}$. Then, an interval $e \in E_I$ is said to be *almost compatible* with the induced adjacency graph H_A if the vertices in e induce a connected subgraph in H_A.

The basis of this definition is that we shall use the COMP-mCi1P adjacency graph to provide a set of solutions, and we shall only consider intervals that are consistent with this set of solutions. If an interval is not almost compatible with the induced adjacency graph, then, irrespective of which certificate for the COMP-mCi1P we choose, we will not find this interval on the certificate. Thus, the assumption we make is that a COMP-mCi1P certificate for the output we desire must be a walk on the induced adjacency graph. Under this condition, we have the following partial result for this problem.

Theorem 3.2. *[CPR13] Let $H = (V, E), w\colon E \to \mathbb{R}_{\geq 0}$ be a weighted hypergraph, with a multiplicity function $c\colon V \to \mathbb{N}$, such that the induced adjacency graph instance $(H_A = (V, E_A), c)$ is COMP-mCi1P. Assume that, for all $e \in E_I$, e is an almost compatible interval of size 3 (a triple), and no triple contains more than 1 repeat. MAX-(3)INT-mCi1P can be solved on the instance (H, c, w) in time $O\left((n+m)^{3/2}\right)$.*

This theorem poses a restriction on the types of intervals that we are allowed. However, if we have a bound on the size of maximal repeat clusters, we can do away with this restriction, which gives us the following corollary.

Corollary 3.1. *[CPR13] Let $H = (V, E), w\colon E \to \mathbb{R}_{\geq 0}$ be a weighted hypergraph, with a multiplicity function $c\colon V \to \mathbb{N}$, such that the induced adjacency graph instance $(H_A = (V, E_A), c)$ is COMP-mCi1P. Assume that the size of a maximal repeat cluster in H_A is at most 1, and for all $e \in E_I$, e is an almost compatible triple. MAX-(3)INT-mCi1P can be solved on the instance (H, c, w) in time $O\left((n+m)^{3/2}\right)$.*

The techniques for proving both these results follow a two-step process. First, we prove that certain triples are always included in a maximum weight subset S. This is done by using the fact that the induced adjacency graph is COMP-mCi1P, and so, it must have a valid certificate $\mathcal{S}$ of circular sequences, on which every edge of H_A appears as a consecutive substring. For Theorem 3.2, there are two such triples: (i) those that contain no repeat vertices, and (ii) those that contain exactly 1 repeat, and there is an adjacency between the two unique vertices in the triple. For the corollary, the triple with exactly 2 repeats is also added to this list. Then, we construct an adjacency graph by introducing an adjacency for each remaining triple which connects the two unique vertices in it. In this graph, after reweighting the edges suitably, we can delete a minimum cardinality (weight) subset of the newly introduced adjacencies using Manŭch et al.'s algorithm from Theorem 2.7.

While this is a good result to start with, the MAX-(k)INT-mCi1P problem is still open for the general case.

Question 3.1. Given a weighted hypergraph $H = (V, E), w\colon E \to \mathbb{R}_{\geq 0}$, with a multiplicity function $c\colon V \to \mathbb{N}$, such that (H_A, c) is COMP-mCi1P, and $\Delta = k$, what is the complexity of MAX-(k)INT-COMP-mCi1P on the instance (H, c, w) for $k \geq 3$, if there is no bound on the size of a maximal cluster?

The next problem asks if we can use an order function $o\colon E \to V^*$ to increase the tractability of MAX-(k)INT-mCi1P.

Question 3.2. (MAX-RSI-mCi1P)

Given an ordered, weighted hypergraph $H = (V, E), w\colon E \to \mathbb{R}_{\geq 0}$, with multiplicity function $c\colon V \to \mathbb{N}$ and order function $o\colon E \to V^*$, such that (H_A, c) is COMP-mCi1P, find a maximum weight subset $S \subseteq E_I$ such that the hypergraph $(H' = (V, E_A \cup S), c)$ is COMP-mCi1P.

None of the existing methods developed seem to be suitable for this case. The most useful optimization technique available to us is the maximum matching algorithm, but in the case of repeat spanning intervals, there is no clear signal as to how we can extend this. We conjecture that this problem is algorithmically hard.

3.2.3 Fixed parameter tractability of the mC1P decision problem

Assume we are given a hypergraph $H = (V, E)$ and a multiplicity function $c\colon V \to \mathbb{N}$. Recall the following notation: Δ is the size of the largest edge in the hypergraph, δ is the maximum degree over all vertices in

V, μ is the maximum multiplicity of over all the vertices, and ρ is the cardinality of the set $|V_R|$, i.e. the number of repeats. We have the following result for the mC1P.

Theorem 3.3. *[CPR13] The mC1P problem is FPT with respect to the parameterization (ρ, μ), provided δ and Δ are constants, and can be decided in time $O\left((\delta(\Delta + \rho\mu))^{2\rho\mu}(n + m + s + \rho\mu)\right)$ and space $O(n + m + s + \rho\mu)$, where $s = \sum_{e \in E} |e|$.*

The algorithm for this result works as follows. Given an instance $(H = (V, E), c)$, we construct a set of new hypergraphs with multiplicity function mapping to 1, such that (H, c) is mC1P if and only if at least one of these hypergraphs is C1P. These hypergraphs are created by making $c(v)$ copies for each repeat $v \in V_R$, and then choosing neighbours for each of these new vertices. The choice of neighbours is where the exponential term arises. After this, we introduce new edges in the constructed hypergraph depending on the choice of neighbours and the original edges in E.

The problem with this result is that almost every aspect of the instance has been parameterized or fixed. The algorithm is exponential in $\rho \cdot \mu$, and both Δ and δ occur in the base for this exponent. Thus, the non-polynomial component in the runtime grows very fast. So, while the algorithm is technically FPT in the given parameters, it is not of much practical significance. Thus, we still have the following questions to answer.

Question 3.3.
1. Is there a better FPT algorithm for mC1P in the parameters (ρ, μ), keeping Δ and δ constant?
2. What is the parameterized complexity of mC1P for other parameterizations, such as the maximum size of a maximal repeat cluster?

At this point, there is also no known parameterized intractability result for the mC1P problem in other parameters, which means that the limits of the parameterization technique is still unknown for this problem.

3.2.4 Extending the results to signed hypergraphs

In terms of application to genome assembly and physical mapping, we often have to deal with directed markers. This direction is not naturally captured by the classical notions of C1P or mC1P.

The question of directed markers arises from the fact that genomic segments are oriented along a chromosome if a genome is double stranded. Until now, each segment was represented by a single vertex in the hypergraph. To handle oriented markers, we represent a genomic segment by two vertices instead- one for each extremity of the segment, with an adjacency between these two vertices, called a *required* adjacency, as opposed to the rest of the adjacencies, which are *inferred*. In order to get an assembly of the genome under consideration, we extend the notion of an mC1P/COMP-mCi1P certificate to require that the walk on the covering graph alternate between required adjacencies and inferred adjacencies.

Both decision and optimization problems can be addressed using this technique. The decision algorithm for the mC1P on 2-graphs remains almost the same [WMPS11], and Manŭch et al.'s algorithm for MAX-COMP-mCi1P can be suitably modified to deal with signed hypergraphs [MPW$^+$13]. There are subtle differences in the treatment of signed hypergraphs and normal hypergraphs. For example, where we originally looked for an Eulerian cycle in the decision algorithm for 2-graphs, we now need to look for an Eulerian cycle that alternates between required and inferred edges. For the optimization result, the construction of the auxiliary graph changes slightly.

Our results can be likewise extended to signed genomes. For the problem MAX-(3)INT-mCi1P, the signed analog takes into account compatible *quadruples* of vertices, rather than triples. The basic concept of

the algorithm remains the same, and the corollary can be extended as well. For the problem RSI-mCi1P, the order function is defined for a signed hypergraph, and the decision algorithm proceeds as before, except that the paths that we create out of every repeat spanning interval now alternate between inferred and required edges. The FPT algorithm for the mC1P remains much the same, with the only difference being that we choose 1 neighbour for each new vertex introduced, rather than 2 neighbours (since the required edge always accounts for 1 neighbour).

3.3 Research avenues

The algorithms developed in this chapter, as stated before, have been recently used in the reconstruction of the genome map of the ancestral Black Death plague bacterium. There are many open question remaining, however.

Extending MAX-COMP-mCi1P Manŭch et al.'s optimization result [MPW$^+$13] is limited to 2-graphs, and is the strongest possible result for optimizing the COMP-mCi1P. It is not known what sort of restrictions or relaxations would lead to a larger class of tractable cases. Since the optimization problem is so important, one of the main problems faced by us is to extend this tractability result. We still cannot find a larger class of instances (H, c), with $\Delta > 2$ and $\mu \geq 1$, for which there is a polynomial time algorithm for MAX-COMP-mCi1P. The problem lies in classifying the types of intervals which permit a polynomial time algorithm for MAX-COMP-mCi1P. This includes checking the property when we have ordered (possibly repeat spanning or repeat overlapping) intervals, small almost-compatible intervals etc.

As of now, the condition $\Delta > 2$ implies NP-completeness. A restriction on the type of hyperedges in H, such as matched multiedges perhaps, or a bounded size for maximal repeat clusters, might yield tractable results. However, assuming such attempts do not yield polynomial time algorithms, we can look at the following problem.

Question 3.4. What is the parameterized complexity of the problem MAX-COMP-mCi1P, parameterized by the number of hyperedges to delete?

Recall that MAX-ROW-C1P is FPT with respect to the number rows being deleted [DGN10]. It seems plausible that MAX-COMP-Ci1P could also be FPT. However, the inclusion of multiplicities could greatly complicate the problem. Furthermore, the apparent disjoint between classical complexity and parameterized complexity makes it hard to conjecture on this point.

More problems in partial optimization In parallel with the MAX-COMP-mCi1P problem, we have no known method to decide MAX-(k)INT-mCi1P and MAX-RSI-mCi1P. The first and most important problem is to figure out the complexity of the classical, unparameterized problems. Following a negative answer to that, we can address the following questions.

Question 3.5. What is the parameterized complexity of the following edge deletion problems, according to some 'nice' parameterization?

1. MAX-(k)INT-mCi1P.
2. MAX-RSI-mCi1P

The natural parameterization is the number of edges to delete. It is recommended to look into other parameterizations, though. One particularly important one may be the maximum size of a maximal repeat cluster. Corollary 3.1 makes use of this to find a tractable result for the MAX-(3)INT-mCi1P problem, and the RSI-mC1P problem uses repeat spanning intervals to 'resolve' such clusters. This seems to indicate that the size of these clusters plays an important role in the tractability of both the decision and the optimization problems.

Even if there is no FPT algorithm for any of the problems mentioned in this section (i.e. if they are at least $W[1]$-hard), we can still look into exponential time algorithms for them, which often significantly improve on the brute force algorithm of simply trying all possible linear orders of the vertices. Techniques for finding such algorithms will probably build on the dynamic programming exact algorithm for the Travelling Salesman Problem (TSP).

Sampling matchings for MAX-COMP-mCi1P The current algorithm for MAX-COMP-mCi1P for adjacencies works by constructing an auxiliary graph and then finding a matching in this graph. The edges retained by the matching are known to be compatible with some COMP-mCi1P ordering of the vertices. The maximum matching algorithm itself is deterministic and runs in polynomial time.

However, choosing a maximum weight matching also means that we are discarding many other possible COMP-mCi1P certificates. Ideally, one would hope that one could examine each matching, and choose one based on its fit with other data. For example, if we have repeat spanning intervals or repeat overlapping intervals available, we might want a matching that can give an unambiguous ordering when these intervals are included. But counting matchings is a #P-complete problem [Jer87, Val79], so we cannot hope to iterate through them all.

Given a matching M on the 2-graph $G = (V, E)$, with weight function $w\colon E \to \mathbb{R}_{\geq 0}$, we say that the weight of the matching M is given by $w(M)$. We can define a Boltzmann distribution on the set of all matchings, with the probability of choosing matching M from this distribution being given by:

$$Pr(M) = \frac{e^{-w(M)/T}}{\sum_{M \in \mathcal{M}} e^{-w(M)/T}},$$

where $\mathcal{M}$ is the set of all matchings in G, and T is a temperature parameter, which can be used to vary the distribution.

An algorithm for sampling matchings from this distribution would be a useful addition to the treatment of COMP-mCi1P instances. The current standard for such algorithms are Markov Chain Monte Carlo based methods [PW98].

In a similar vein, we can talk about sampling orderings for an mC1P instance. The problem of counting all possible certificates was studied in detail by Battaglia et al. [BGS12], who used this work to prove that mC1P is NP-complete. Their approach is of interest because they encode certificates as generalized PQ-trees. However, they prove that the problem of counting such certificates is #P-complete in general. It remains to be seen if one can adopt an approach that can sample from the set of possible certificates.

Finally, we have the question of either counting or sampling from the space of certificates for the tractable instances of the mC1P/COMP-mCi1P that we have encountered. If we are given a Yes instance for the mC1P problem, or a Yes instance for COMP-mCi1P on adjacencies, it is not known if we can count all possible certificates. If counting these certificates if algorithmically hard, it would be desirable to have a sampling algorithm to choose a valid certificate from a probability distribution.

Question 3.6. 1. Is there a polynomial time algorithm to count the number of certificates for an mC1P/COMP-mCi1P instance, given certain restrictions (such as repeat spanning intervals, or almost compatible

triples)?

2. Is there an efficient method to randomly sample certificates for an mC1P/COMP-mCi1P instance from a probability distribution?

The problem of counting the number of Eulerian tours (and thus, the number of certificates for a Yes instance for the mC1P on adjacencies), is #P-complete [BW04], but limited results exist for randomly sampling from the list of tours [TV01]. Exploring these avenues, especially in the case of good data that we already have, would be a very interesting future project.

Chapter 4

C1P with gaps: Alternative approaches

As we saw in Chapter 2, Atkins et al. forwarded an algorithm that solves the C1P decision problem exactly for matrices having the property [ABH98]. The algorithm is unique in that it is based on the spectrum of the Laplacian matrix associated to a graph on the set of columns, and outputs a PQ-tree even when the matrix is not C1P. The permutations encoded by it in this case, however, seem to have no direct combinatorial interpretation.

In recent years, the field of graph partitioning through Laplacian spectra has gathered much attention. There are many algorithms that use the Laplacian spectrum to find sparse cuts and multicuts. This gives us the notion that if we can separate the columns of a matrix into sets of dense components, with few rows in common between these components, the number of gaps created cannot be too large.

This chapter is an exploratory discussion on the spectral algorithm of Atkins et al., and on connections with graph partitioning. We hope results from that field will prove to be useful in addressing the problem of C1P with gaps.

4.1 Definitions and concepts

In this chapter, we shall revert back to the binary matrix representation that was originally used to introduce the C1P. We may have to deal with weighted graphs, and in that context, we introduce a few definitions. If $G = (V, E)$ is a weighted graph with weights $w \colon E \to \mathbb{R}_{\geq 0}$, the degree of a vertex v (still denoted by $deg\,(v)$) is said to be the quantity $\sum_{e \in E, e \text{ incident to } v} w\,(e)$. Given a subset $S \subseteq V$ of the vertices, the volume of the set S is the quantity $\sum_{v \in S} deg\,(v)$, and it is denoted by $vol\,(S)$.

We also need the definition of the Laplacian matrix of a graph.

Definition 4.1. Let $G = (V, E)$ be a simple graph, with vertex set $V = \{v_0, \ldots, v_{n-1}\}$. The *Laplacian matrix* of G, denoted by $\mathcal{L}_G$ (and denoted by $\mathcal{L}$ if the graph under consideration is clear by the context) is defined as follows.

$$l_{ij} = \begin{cases} -1 & \text{if } (v_i, v_j) \in E, \\ 0 & \text{if } (v_i, v_j) \notin E \text{ and } i \neq j, \\ deg\,(v_i) & \text{if } i = j, \end{cases}$$

where l_{ij} is the entry on the i^{th} row and j^{th} column of $\mathcal{L}_G$, the rows and columns being indexed by $[n]$.

We can also defined the normalized Laplacian matrix by multiplying the Laplacian on both sides by the matrix $D^{-1/2}$, which is a diagonal matrix indexed by the vertices, with diagonal entry $d_{ii} = 1/\sqrt{deg\,(v_i)}$

for each vertex $v_i \in V$. The graph Laplacian is a very well studied object [Moh92, MA91, Chu97, PSL90], and of great interest in the design of graph algorithms, particularly in graph partitioning [Tre12, LOGT12, KLL$^+$13].

One concept that will be a recurring topic in this chapter is that of an *edge cut* in a graph.

Definition 4.2. Let $G = (V, E)$ be a simple graph. A *partition* of G is a subset $S \subseteq V$ of the vertices, and the *edge cut* corresponding to S is the set of edges which have exactly one end point in S. The set of cut edges is denoted by $E\left(S, \overline{S}\right)$, where $\overline{S}$ represents the complement of S, i.e. the set $V \setminus S$. The *size* of the edge cut $E\left(S, \overline{S}\right)$ in an unweighted graph is the cardinality of the edge cut. In a weighted graph, with edge weights $w \colon E \to \mathbb{R}_{\geq 0}$, the size of an edge cut $E\left(S, \overline{S}\right)$ is the quantity $\sum_{e \in E\left(S,\overline{S}\right)} w\left(e\right)$. The size of an edge cut $E\left(S, \overline{S}\right)$ in both the unweighted and the weighted case is denoted by $|E\left(S, \overline{S}\right)|$.

A quantity of great interest in the computer science community, which is related to flows and connectivity, is the *edge expansion* of a graph.

Definition 4.3. Let $G = (V, E)$ be a d-regular graph, and let $S \subset V$ be a partition in this graph. The *edge expansion* of S is the following quantity.

$$\phi\left(S\right) = \frac{|E\left(S, \overline{S}\right)|}{d \min\left\{|S|, |\overline{S}|\right\}}. \tag{4.1}$$

The *edge expansion of* G is the minimum expansion over all possible proper subsets S of V.

$$\phi\left(G\right) = \min_{S \subset V} \phi\left(S\right). \tag{4.2}$$

While edge expansion is defined for regular graphs, a closely related quantity, the *conductance* of a graph, is defined for general graphs. For a subset $S \subset V$ of a general graph, its conductance is defined as

$$\phi\left(S\right) = \frac{|E\left(S, \overline{S}\right)|}{\min\left\{vol\left(S\right), vol\left(\overline{S}\right)\right\}}. \tag{4.3}$$

Results that hold for edge expansion usually also hold for conductance. Furthermore, they can be generalized to weighted graphs. Since we shall only be dealing with edge expansion in this document, we shall just use the term *expansion* to indicate edge expansion.

Computing the expansion of a graph is NP-hard. However, there are very clever results that use spectral graph theory and semi-definite programming to obtain approximation algorithms for this quantity [ARV09, LR99].

Treewidth and pathwidth The concepts of treewidth and pathwidth were first defined by Robertson and Seymour [RS84].

Definition 4.4. (Treewidth and pathwidth)

1. Let $G = (V, E)$ be a simple graph. A *tree decomposition* of G is a tuple $(\mathcal{X}, \mathcal{T} = (\mathcal{X}, \mathcal{E}))$, where $\mathcal{X} = \{X_0, \ldots, X_{r-1}\}$ is a collection of subsets of V, called *bags*, and $\mathcal{T}$ is a tree with nodes X_i, such that

   (a) $\bigcup_i X_i = V$,

(b) for every $e \in E$, $e = (u, v)$, there is a bag X_i such that $u, v \in X_i$, and

(c) for any vertex $v \in V$, the set $\{X_i \in \mathcal{X} : v \in X_i\}$ induces a subtree in $\mathcal{T}$.

The *width* of a tree decomposition $(\mathcal{X}, \mathcal{T})$ of a graph G is the size of the largest bag in $\mathcal{X}$. The *treewidth* of G is the minimum width taken over all tree decompositions of G.

2. A *path decomposition* of a graph G is a tree decomposition $(\mathcal{X}, \mathcal{T})$ where the tree $\mathcal{T}$ is a path. The *pathwidth* of a graph G is the minimum width taken over all path decompositions of G.

The treewidth and pathwidth of graphs are NP-hard to compute in general.

4.2 Connecting graph partitioning and the C1P

In Section 2.3, we briefly mentioned that Atkins et al.'s algorithm [ABH98] was proved by Vuokko [Vuo10] to output a set of permutations that minimize the principal component of the Frobenius norm of a certain matrix. The sketch of the algorithm is as follows. Given a binary matrix M, we construct the matrix $A = M^T M$ from it. This matrix, after setting the diagonal elements a_{ii} to 0, is used as a weighted adjacency matrix for a simple graph G_A. The algorithm then calculates the second smallest eigenvalue λ_1 of the Laplacian of the graph G_A, sorts the eigenvector v_1 corresponding to this eigenvalue, and recurses on identical entries in v_1. Atkins et al. proved results that bounded the multiplicity of λ_1, and imposed restrictions on the structure of v_1, given that the matrix M is C1P.

The problem, as mentioned before, is that the algorithm returns a PQ-tree even if the matrix is not C1P. While Vuokko's work on the significance of the permutations encoded in this tree is indicative of some gap optimization criterion, it does not give us a clear description of the quantity being optimized. Here, we introduce a classical result, first proved for Reimannian manifolds by Cheeger [Che70], and later proved by Alon and Milman [AM85, Alo86] for regular graphs. As stated, the result generalizes to non-regular graphs and weighted graphs.

Theorem 4.1. *[Alo86] (Cheeger's inequalities)*
Let $G = (V, E)$ be a d-regular graph. Then, there is a constructive algorithm which proves the following result.

$$\frac{\phi^2(G)}{2} \leq \lambda_1 \leq 2\phi(G), \tag{4.4}$$

where λ_1 is the second smallest eigenvalue of the normalized Laplacian matrix of G.

Better results have been found since [KLL$^+$13, LOGT12], but from our perspective, the interest stems from the fact that the algorithm used to prove the upper bound on $\phi(G)$ sorts the vertices of the graph according to the eigenvector of λ_1, and then a probabilistic argument shows that there is an edge cut according to this sorted order which has an expansion of at most $\sqrt{2\lambda_1}$. The ordering scheme used is the same as the one used in Atkins et al.'s algorithm. The only difference is that while we look for a bisection in the case of the minimum expanding set algorithm, the C1P algorithm looks for a multipartition based on collections of equal entries in the eigenvector.

4.2.1 The theory behind the C1P spectral algorithm

We stated that the algorithm works on the matrix $A = M^T M$. Let us analyze this matrix.

Property 4.1. Let M be an $m \times n$ binary matrix. Then, in the matrix $A = M^T M$,

1. the diagonal entry a_{ii} is the number of rows k such that $m_{ki} = 1$ in M,
2. a_{ij} is the number of rows k such that $m_{ki} = m_{kj} = 1$ in M, i.e. the number of *common rows*,

When we ask for a C1P ordering, or a nearly-C1P ordering (defined by, say, the number of gaps) of a matrix M, we are asking for a permutation of the columns such that no (or few) gaps appear in the rows of M. A simple relaxation of this is to ask instead for an embedding of the columns in $\mathbb{R}$. Ideally, we would want this embedding to be one-to-one. The matrix A provides us with a way of quantifying this. Assume that two columns i and j have a large number of common rows, but i shares very few rows with another column k. Then, if we have an embedding $\pi \colon \mathcal{C} \to \mathbb{R}$ such that $\pi(i) < \pi(k) < \pi(j)$, then we shall be creating gaps in some of the rows that i and j share, but which do not appear in k. Formally, the matrix A encodes a cost function, where a_{ij} is the cost of separating i and j. The exact cost minimized by the spectral algorithm is given by the Rayleigh quotient of the Laplacian $\mathcal{L}_A$, denoted by $R(\cdot)$.

$$\begin{aligned} R(x) &= \frac{x^T \mathcal{L}_A x}{\|x\|^2} \\ &= \frac{\sum_{\{i,j\}} a_{ij} |x_i - x_j|^2}{\sum_i x_i^2}, \end{aligned} \tag{4.5}$$

where $x \in \mathbb{R}^n$, x_i is the i^{th} entry in the vector x, and $\|x\|$ denotes the Euclidean norm of the vector x. This quadratic function has value 0 when x is the all 1's vector (denoted by $\mathbf{1}$), and this corresponds to the smallest eigenvalue of the Laplacian, i.e. 0. To get the second smallest eigenvalue, we note that the Laplacian is a real symmetric matrix, and has an orthonormal basis of eigenvectors. Thus, we get

$$\lambda_1 = \min_{x \in \mathbb{R}^n, x^T \mathbf{1} = 0} R(x),$$

and the eigenvector gives us an embedding that attains this cost.

4.2.2 Graph expansion and the C1P

The argument we seek to make is that graph expansion is a useful parameter for studying the distribution of gaps in a non-C1P matrix. Consider the set of columns in a matrix M corresponding to the vertex set of minimum expansion S in the graph $G_A = (V, E)$ formed from $A = M^T M$. We shall use S to denote both this set of vertices and the set of columns in M corresponding to these vertices. Since it has low expansion, we can infer that the quantity $|E\left(S, \overline{S}\right)|$ is small compared to $vol(S)$. Since each edge in the graph is weighted by the number of common rows between the columns corresponding to the end points of the edge, we can say the following.

$$\begin{aligned} |supp(S) \cap supp\left(\overline{S}\right)| &\leq |E\left(S, \overline{S}\right)|, \\ |supp(S)| &\leq vol(S). \end{aligned}$$

In the matrix M, if we keep all columns corresponding to S together in a permutation of the columns:

1. moving the columns corresponding to S cannot create any gaps in rows supported only by S,

2. moving the columns corresponding to S can create at most 1 gap in the rows that are not supported by S, since none of the columns in S have a 1 in these rows, and

3. moving S can create an arbitrary number of gaps in rows supported by at least 1 column in S and at least 1 column in $\overline{S}$, but there are not 'too many' such rows.

This intuition provides a direction for exploring the C1P problem as a graph partitioning problem: find a minimum expanding set, collapse the corresponding set of columns into a single column and find an almost C1P permutation on the rest of the matrix, then locally optimize within the set.

This is a very loose approximation, since we have no information about 3 or more columns supporting the same row. For most cases, this is a gross overestimate. However, this approach may give us a bound on the minimum total number of gaps that can be created. The questions raised by this are summarized below.

Question 4.1. 1. Does Atkins et al.'s algorithm find a set S_{opt} which approximates the following quantity for a matrix M?

$$\psi(M) = \min_{S \subset \mathcal{C}_M} \frac{|supp(S) \cap supp\left(\overline{S}\right)|}{|supp(S)|}.$$

2. What is the relation between the quantity $\psi(M)$ and the problem of finding the smallest k such that the matrix M is $(k, *)$-C1P? Is there a Cheeger-type inequality that approximates k?

3. Assuming Part (2) is answered, can we use techniques for approximating $\phi(G_A)$ to approximate $\psi(M)$?

We must also remember that the underlying question is one of finding a permutation of the columns (i.e. a bijective mapping from $\mathcal{C}$ to itself), which we relaxed to embedding the columns in $\mathbb{R}$. Leighton and Rao [LR99] formed a relaxation of this to arbitrary metric spaces and used that to devise an $O(\log n)$ factor approximation algorithm for minimum expansion. This lends us the idea of further relaxing the current embedding into $\mathbb{R}$. Semi-definite relaxations often give the strongest possible approximation result (under certain conjectures), buts are also the hardest to devise and analyze.

The basic framework of a result along these lines should probably proceed as given below.

1. Define the *expansion* of a binary matrix. This definition should quantify a set of columns which can be moved together in the matrix without creating 'too many' gaps.

2. Prove that the expansion of the matrix is bounded (possibly in both directions) by a function of λ_1, where λ_1 is the second smallest eigenvalue of the Laplacian/normalized Laplacian. This should be a constructive proof, building on the algorithm of Atkins et al.

3. Use existing tools in the theory of sparse cuts and graph expansion to improve the result.

4.3 Pathwidth and gaps

A different way to make a matrix 'almost C1P' is to look at the graph theoretic analog of C1P matrices.

Definition 4.5. A graph $G = (V, E)$ is an interval graph if we can associate an interval on the real line to each vertex $v \in V$, and two vertices $u, v \in V$ form an edge if and only if the corresponding intervals have a non-empty intersection.

We will just state here that interval graphs can be associated to C1P matrices, and testing algorithms for the C1P can be used to recognize interval graphs [BL76, HMPV00]. We have the following theorem on the pathwidth of interval graphs.

Theorem 4.2. *[Möh90]*

1. *The pathwidth of an interval graph $G = (V, E)$ is one less than the size of the maximum clique in G.*
2. *The pathwidth of a graph $G = (V, E)$ is one less than the size of the smallest maximum clique in interval graphs on the vertex set V which contains G as a subgraph (*interval supergraphs*).*

This theorem allows us to take a different view on approximating a C1P matrix- find the smallest interval supergraph of the intersection graph of the rows. We associate a graph to the matrix M, with vertices indexed by the rows, and two vertices forming an edge if and only if there is a column that supports both corresponding rows. Every permutation of the columns uniquely defines a path decomposition of this graph. Assume that the columns are numbered by $[n]$, in the order that they appear in M, and the rows are numbered by $[m]$, and create $(\mathcal{X} = \emptyset, \mathcal{T} = (\mathcal{X}, \mathcal{E}))$.

1. For each column $i \in [n]$, add a bag $X_i \in \mathcal{X}$. Add (X_i, X_{i+1}) to $\mathcal{E}$ for all $i \in [n-1]$.
2. For each row $r \in [m]$, if $m_{ri} = 1$, add r to the bag X_i.
3. For every row $r \in [m]$, if $r \in X_i$ and $r \in X_j$, $i < j$, add r to all X_k, $i < k < j$.

This is a valid path decomposition of an interval supergraph of the graph associated to M, but not necessarily the smallest such supergraph. If M is not C1P, there are bags which contain a row even if the entry corresponding to that row and column is a 0. A gap in a row $r \in \mathcal{R}$ is a maximal subpath P in $\mathcal{T}$ such that $r \in X_i$ for all $X_i \in V(P)$, and $m_{ri} = 0$. In such a construction, the cardinality of a bag $X_i \in \mathcal{X}$ will be equal to the number of rows having a 1 in column i plus the number of rows in which column i is involved in a gap, and the pathwidth is the size of the largest bag minimized over all possible permutations of the columns. A minimum width path decomposition of the intersection graph of the binary matrix will try to minimize the number of gaps introduced in a single column, i.e. not too many rows will have a gap in any column, and we associate the subpaths in $\mathcal{T}$ defined by each row to an interval to get the corresponding interval graph. Kaplan and Shamir studied the relation between pathwidth and bandwidth with interval graph completion in detail [KS96].

Many exact exponential algorithms rely on the tree and path decompositions of graph, and FPT algorithms often use a parametrization by pathwidth or treewidth. This makes such a definition attractive in the sense of finding tractable algorithms for the (k, ℓ)-C1P problem, or closely related C1P maximization problems, since the problem of deciding if a given intersection graph has a pathwidth of at most p is linear time solvable for fixed p [Bod96].

Question 4.2. Is any variation of (k, ℓ)-C1P FPT under parametrization by pathwidth? If not, does there exist finite $t \in \mathbb{N}$ such that the problem is $W[t]$-complete?

Chapter 5

Applications

In Chapters 3 and 4, we discussed a number of theoretical problems related to C1P variants, and we discussed several recently obtained algorithmic results of practical relevance for genome assembly problems. In this chapter, we review the applied projects that will make use of these positive results, as well as of some of the extensions to these we expect to obtain.

5.1 Software development

In order to apply our new algorithms on real data, we will integrate them into recently developed software in the Computational Palaeogenomics group[1]: ANGES (reconstructing ANcestral GEnomeS maps, [JRTC12]) aimed at inferring the organization of ancestral genomes from the comparison of extant related genomes, and FPSAC (Fast Phylogenetic Scaffolding of Ancient Contigs, [RTC13]), aimed at inferring the organization of sequenced ancient contigs using a comparative approach.

Both software are organized as a suite of Python scripts, in order to ease the integration of parts of them into pipelines, and rely on the same underlying methodological principles (in fact most FPSAC code is taken from the first version of ANGES). They differ mostly in the data acquisition phase, as FPSAC computes marker families from the alignment of sequenced ancient contigs onto related extant genomes, while ANGES relies on whole-genome alignments or gene families to define marker families.

The main developments to these software will be, aside from adding newly developed algorithms, to integrate them both into a single unified suite of python scripts, complemented with different graphical interfaces specific to the different tasks they perform (maps versus scaffolds).

This is intended to be a first step toward a general multi-genome scaffolding/mapping system that could consider several extant and ancient genomes and heterogeneous data (extant genome sequences, ancient contigs, gene families, whole genome alignments) at the same time, although the design of a first prototype of such a system is out of the scope of this proposal.

5.2 Pathogen evolution, and the *Yersinia pestis* genus

The bubonic plague, historically one of the most feared human diseases, is caused by bacteria from the *Yersinia pestis* genus, and is thought to be the cause of dramatic pandemics, among which the Black Death

[1] `http://paleogenomics.irmacs.sfu.ca/`

pandemic wiped out more than half of the European population in the late Middle-Ages. In a recent breakthrough, Bos et al. [BSG+11] proposed a draft genome of the bacterium by sequencing a metagenome obtained from a mass graveyard for victims of the Black Death, and comparing it against extant species of *Yersinia Pestis*. They provided strong evidence that the Black Death causative agent was a *Yersinia pestis* strain, thus ending a long controversy about the etiology of the Black Death. Moreover, they showed that this strain was likely to be the ancestor of most extant plague-causing bacteria. However, due to the decay of ancient DNA, they were not able to complete the assembly of the chromosome of this bacterial strain, and obtained a highly fragmented draft, composed of above $130,000$ contigs, of which only $2,105$ were of length above 500bp, thus leaving the possibility of analysing the evolution of synteny with the *Yersinia Pestis* clade out of reach.

The data obtained by Bos et al. is of particular interest for us, for several reasons: first, the sequenced contigs belong to an extinct species that is ancestral to existing sequenced and assembled *Yersinia Pestis* strains, and second, *Yersinia Pestis* genomes have a very conserved sequence (i.e. point mutations are rare), contain many repeats, and their synteny is significantly unconserved due to an estimated high genome rearrangement rate [DMR08]. This prompted us to develop a scaffolding method, FPSAC, based on the methodological principles implemented in ANGES but including specific elements allowing us to process ancient contig data, and to apply it to the Black Death data set. Our preliminary results are interesting, as we were able to organize all contigs into a unique scaffold, as well as estimate the genome sequence of the inter-contig gaps, thus providing a much refined (in fact, an almost finished) draft of the chromosome of the causative agent of the Black Death pandemic [RTC13]. From an algorithmic point of view, the key elements that opened the way to such a result were the algorithms developed by Manŭch et al. [MPW+13] and by us [CPR13] to integrate the notion of repeated markers within the C1P framework.

The next stage in this project will be to reconstruct the organization of all internal nodes of the *Yersinia Pestis* phylogeny. While only one internal node is currently provided with sequencing data (the Black Death agent), the extreme sequence conservation within the *Yersinia Pestis* clade allows us to reasonably consider that these sequenced contigs can be used as proxies for the ancestral sequences of the other extinct strain genomes. The results obtained so far suggest that the syntenic signal within this clade is strong enough that complete or almost complete assembly drafts can be obtained for all considered ancient genomes. In parallel, we will also reconstruct ancestral genome maps based on gene families computed by the Bioinformatics and Evolutionary Genomics Group at the LBBE lab in Lyon[2], in order to assess the robustness of the inferred genome drafts to data sources. This will also provide a first data set to consider for developing integrated assembly methods combining heterogeneous data sources. Within the context of this proposal, the emphasis will be on the assembly draft generation, while others in the Computational Paleogenomics group and in the collaborating labs will analyze the obtained results.

Altogether, we expect this preliminary applied work to be a significant methodological advance in the field of computational evolutionary pathogenomics, that could be applied to other important pathogens where several extant strains have been sequenced and assembled, such as *Vibrio cholera* or *Mycobacterium tuberculosis*.

5.3 Eukaryotic genomes

Data for eukaryotic genomes present a greater challenge for our methods, due to the more complex nature of their genomes (especially in terms of repeats) and their much larger size. Nevertheless, recent works have

[2] `http://lbbe.univ-lyon1.fr/-Equipe-Bioinformatique-et-.html?lang=en`

shown that, at least at the level of ancestral genome maps, our algorithms and software can handle such data [OTC11].

One project is motivated by a recent work in our lab on the reconstruction of the ancestral amniote genome [OTC11], and will concentrate on two data sets, composed respectively of four assembled teleost fishes and of four assembled bird/reptilian genomes. For both datasets, both whole genome alignments and gene family trees are available from the public Ensembl database[3], that will provide the initial data for computing markers. We are well aware of the pitfalls of using gene trees for such data sets, due to the many well documented errors in the available uncurated trees obtained through high-throughput automatic methods; our group, in collaboration with the LBBE and Nadia El-Mabrouk's group in Montréal, is currently developing gene tree correction methods specifically targeted at being used in reconstructing ancestral genomes. The final goal of this part of the proposed research is, again, the development and preliminary implementation of a pipeline that will be well suited to such large-scale projects.

We might also explore a second project regarding eukaryotic genomes, centered on horse genomics. In a recent paper, Orlando *et al.* have sequenced the genomes of several ancient and modern horses [Orl13]. As expected, the assemblies of the sequenced data are highly fragmented. We are currently having preliminary discussions with the Orlando's lab on the application of our software FPSAC to improve the assembly of the ancient horse genomes.

[3] `http://uswest.ensembl.org/`

Chapter 6

Conclusions

The C1P problem has been studied since the 1960's, and the various questions arising from this scrutiny are of important theoretical and practical importance. In particular, variants of the C1P (and not just the ones examined in this report) are usually good models for a variety of different problems, but at the same time, they are almost always computationally intractable.

We now give a brief summary of the problems we have seen in this report, the avenues of attack available to us, and the application of these techniques to genome assembly.

The mC1P The mC1P and COMP-mCi1P are very well suited to genome assembly applications, and we saw some recent tractable results regarding these problems. The general decision and optimization problems seem to be, at the moment, intractable. However, the existence of FPT algorithms for the mC1P decision problem [CPR13] and the Eulerian deletion optimization problem [CMP$^+$11] provide us with the starting point for examining exact exponential algorithms for both problems. The RSI-mC1P problem, which was proved to be tractable [CPR13], seems to have an easy generalization to include repeat overlapping intervals. The proof for this problem is still in progress.

When talking of partial optimization techniques, the main problem of interest was the MAX-(k)INT-mCi1P, which was showed to have limited tractability. The general case, even for $k \geq 3$, does not seem to be tractable. The problem MAX-RSI-mCi1P, may provide some limited tractability instances for such optimization problems. It remains to be seen if we can use fixed parameter techniques for either of these problems.

Since the mC1P is so important for genome assembly applications, efficient sampling of matchings to get a unique assembly is a research theme that needs to be explored. This offers an alternative heuristic technique to obtain a unique assembly, compared to the possibly intractable optimization problems we have already discussed.

C1P with gaps The tractability of the (k, ℓ)-C1P problem is almost settled [MPC12, MP11], with the exception of $(2, 1)$-C1P. Still, the problem offers an intriguing way to characterize the notion of 'almost-C1P', and the fact that it does not involve discarding any of the information provided to us makes this a problem worth pursuing.

The connection of the spectral algorithm for C1P [ABH98] to minimizing the number of gaps is only vaguely understood as a combinatorial problem [Vuo10], but the plethora of research done in the field of metric embeddings and spectral algorithms for graph partitioning suggests that a more substantial connection

exists. The main issue in the direction of this research would be the mathematical tools involved, which are usually quite complicated, and the loss of information when we translate the problem into a familiar graph theoretic setting. The author believes that this setting can be bypassed to obtain a more complete picture, which should then be approachable through more powerful methods, such as semidefinite programming.

As before, finding FPT algorithms for (k, ℓ)-C1P is a useful way to address some of the issues regarding this variant. The notion of pathwidth (treewidth), which we discussed in detail in Chapter 4, may prove to be a key concept where this is concerned. Many exponential algorithms, especially FPT algorithms, are parameterized by pathwidth or treewidth [FK10], and since the pathwidth is a fundamentally important quantity in interval graphs, one would expect finding almost C1P orderings to rely on this fact.

One aspect of the C1P with gaps which has not been discussed is the extension of the notion to the mC1P problem. This is a branch of research that is currently dormant, but a suitable definition of gaps in the presence of a multiplicity function should not be too hard. However, knowing the status of the gapped C1P problem in the classical case, it is not known how far the limits of tractability for the gapped mC1P will extend.

Applications There is already a developing framework for applying most tractable results for the C1P/mC1P to genome assembly. The ANGES [JRTC12] and FPSAC [RTC13] software packages are Python-based pipelines for ancestral genome reconstruction and scaffolding genomes respectively, and the modular structure adopted in both make them suitable for future expansion. They have been used in a limited capacity for assembling genomes, and have shown some success. Further theoretical developments are expected to be incorporated into them through the course of the research, with application to large-scale data sets.

Appendix A

Parameterized complexity and fixed parameter tractability

Parameterized complexity and the design and analysis of fixed parameter algorithms is a comparatively new field in the theory of computational complexity. We give some basic definitions here, so that we can address problems in this new framework. For more details on this topic, we refer you to [FG06, Nie06].

In this section, we shall talk about decision problems. A *problem* is a set $Q \subseteq \Sigma^*$, where Σ is a finite alphabet. For any string $x \in Q$, we define $|x|$ to be the length of x. The question we ask is how we can recognize a string $x \in Q$, preferably using an efficient algorithm. A parameterized problem is defined as follows.

Definition A.1. Let $Q \subseteq \Sigma^*$ be a problem.

1. A *parameterization* of Σ^* is a polynomial time computable function $\kappa\colon \Sigma^* \to \mathbb{N}$.
2. A *parameterized problem* over Σ is a pair (Q, κ), where $Q \subseteq \Sigma^*$ is a set of strings (a problem) over Σ, and κ is a parameterization of Σ^*.

Having given this definition, we have a well defined notion of *fixed parameter tractability*.

Definition A.2.
1. An algorithm with an input alphabet Σ is said to be *fixed parameter tractable* with respect to a parameterization $\kappa\colon \Sigma^* \to \mathbb{N}$ if there exists a computable function $f\colon \mathbb{N} \to \mathbb{N}$, and there exists a polynomial $p \in \mathbb{Z}[X]$, such that, for every string $x \in \Sigma^*$ in Q given as input, the algorithm terminates in time $O\left(f\left(\kappa\left(x\right)\right) p\left(|x|\right)\right)$.
2. A problem $Q \subseteq \Sigma^*$ is said to be *fixed parameter tractable* (FPT) with respect to the parameterization $\kappa\colon \Sigma^* \to \mathbb{N}$ if there is a fixed parameter tractable algorithm with respect to κ that decides Q.

As a counterpoint, we have the concept of *parameterized complexity*. This is defined through a hierarchy of complexity classes, called the W-hierarchy of complexity classes. The hierarchy is given as follows:

$$FPT = W\left[0\right] \subseteq W\left[1\right] \subseteq W\left[2\right] \subseteq \ldots \subseteq W\left[P\right],$$

where the inclusions are conjectured to be strict. The W-hierarchy is defined using circuit complexity, and we do not mention the exact definition here. It suffices here to know that problems in $W\left[1\right]$ itself are considered parameterized intractable.

Bibliography

[ABH98] Jonathan E. Atkins, Erik G. Boman, and Bruce Hendrickson. A spectral algorithm for seriation and the consecutive ones problem. *SIAM J. Comput.*, 28(1):297–310, 1998.

[AKNW95] Farid Alizadeh, Richard M. Karp, Lee Aaron Newberg, and Deborah K. Weisser. Physical mapping of chromosomes: A combinatorial problem in molecular biology. *Algorithmica*, 13(1/2):52–76, 1995.

[AKWZ95] Farid Alizadeh, Richard M. Karp, Deborah K. Weisser, and Geoffrey Zweig. Physical mapping of chromosomes using unique probes. *Journal of Computational Biology*, 2(2):159–184, 1995.

[Alo86] Noga Alon. Eigenvalues and expanders. *Combinatorica*, 6(2):83–96, 1986.

[AM85] Noga Alon and Vitali D Milman. λ_1, isoperimetric inequalities for graphs, and superconcentrators. *Journal of Combinatorial Theory, Series B*, 38(1):73–88, 1985.

[ARV09] Sanjeev Arora, Satish Rao, and Umesh Vazirani. Expander flows, geometric embeddings and graph partitioning. *Journal of the ACM (JACM)*, 56(2):5, 2009.

[Ben59] Seymour Benzer. On the topology of the genetic fine structure. *Proceedings of the National Academy of Sciences of the United States of America*, 45(11):1607, 1959.

[BGS12] Giovanni Battaglia, Roberto Grossi, and Noemi Scutellà. Consecutive ones property and PQ-trees for multisets: Hardness of counting their orderings. *Information and Computation*, 219(0):58–70, 2012.

[BI99] S. Batzoglou and S. Istrail. Physical mapping with repeated probes: The hypergraph superstring problem. In *CPM*, volume 1645 of *LNCS*, pages 66–77, 1999.

[BL76] K.S. Booth and G.S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *Journal of Computer and System Sciences*, 13(3):335–379, 1976.

[Bod96] Hans L Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on computing*, 25(6):1305–1317, 1996.

[BSG+11] Kirsten I Bos, Verena J Schuenemann, G Brian Golding, Hernán A Burbano, Nicholas Waglechner, Brian K Coombes, Joseph B McPhee, Sharon N DeWitte, Matthias Meyer, Sarah Schmedes, et al. A draft genome of yersinia pestis from victims of the black death. *Nature*, 478(7370):506–510, 2011.

[BW04] Graham R Brightwell and Peter Winkler. Note on counting eulerian circuits. *lanl. arXiv. org*, 2004.

[CGOT10] Cedric Chauve, Haris Gavranovic, Aïda Ouangraoua, and Eric Tannier. Yeast ancestral genome reconstructions: The possibilities of computational methods ii. *Journal of Computational Biology*, 17(9):1097–1112, 2010.

[Che70] Jeff Cheeger. A lower bound for the smallest eigenvalue of the Laplacian. In *Problems in analysis (Papers dedicated to Salomon Bochner, 1969)*, pages 195–199. Princeton Univ. Press, Princeton, N. J., 1970.

[Chu97] Fan RK Chung. *Spectral graph theory*, volume 92. AMS Bookstore, 1997.

[CMP+11] M. Cygan, D. Marx, M. Pilipczuk, M. Pilipczuk, and I. Schlotter. Parameterized complexity of eulerian deletion problems. In *Graph-Theoretic Concepts in Computer Science*, pages 131–142. Springer, 2011.

[CMPW11] C. Chauve, J. Maňuch, M. Patterson, and R. Wittler. Tractability results for the consecutive-ones property with multiplicity. In *Combinatorial Pattern Matching*, pages 90–103. Springer, 2011.

[CPR13] Cedric Chauve, Murray Patterson, and Ashok Rajaraman. Hypergraph covering problems motivated by genome assembly questions. *CoRR*, abs/1306.4353, 2013.

[CST12] Cedric Chauve, Tamon Stephen, and Maria Tamayo. Efficient algorithms for finding tucker patterns. *arXiv preprint arXiv:1206.1837*, 2012.

[CT08] Cedric Chauve and Eric Tannier. A methodological framework for the reconstruction of contiguous regions of ancestral genomes and its application to mammalian genomes. *PLoS computational biology*, 4(11):e1000234, 2008.

[DGN10] Michael Dom, Jiong Guo, and Rolf Niedermeier. Approximation and fixed-parameter algorithms for consecutive ones submatrix problems. *J. Comput. Syst. Sci.*, 76(3-4):204–221, 2010.

[DLG94] A. Dessmark, A. Lingas, and O. Garrido. On parallel complexity of maximum f-matching and the degree sequence problem. *Mathematical Foundations of Computer Science 1994*, pages 316–325, 1994.

[DMR08] Aaron E Darling, István Miklós, and Mark A Ragan. Dynamics of genome rearrangement in bacterial populations. *PLoS Genet*, 4(7):e1000128, 2008.

[Dom08] Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property*. PhD thesis, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

[Dom09] Michael Dom. Algorithimic aspects of the consecutive-ones property. *Bulletin of the EATCS*, 98:27–59, 2009.

[Far07] T. Faraut *et al.* A comparative genome approach to marker ordering. *Bioinformatics*, 23(2):e50–e56, 2007.

[FG64] DR Fulkerson and Oliver Alfred Gross. Incidence matrices with the consecutive 1's property. Technical report, DTIC Document, 1964.

[FG06] Jörg Flum and Martin Grohe. *Parameterized complexity theory*, volume 3. Springer Heidelberg, 2006.

[FK10] Fedor V Fomin and Dieter Kratsch. *Exact exponential algorithms*. Springer, 2010.

[GCST11] Haris Gavranovic, Cedric Chauve, Jérôme Salse, and Eric Tannier. Mapping ancestral genomes with massive gene loss: A matrix sandwich problem. *Bioinformatics [ISMB/ECCB]*, 27(13):257–265, 2011.

[GGKS95] Paul W Goldberg, Martin C Golumbic, Haim Kaplan, and Ron Shamir. Four strikes against physical mapping of dna. *Journal of Computational Biology*, 2(1):139–152, 1995.

[GSN11] Song Gao, Wing-Kin Sung, and Niranjan Nagarajan. Opera: Reconstructing optimal genomic scaffolds with high-throughput paired-end sequences. *Journal of Computational Biology*, 18(11):1681–1691, 2011.

[HG02] Mohammad Taghi Hajiaghayi and Yashar Ganjali. A note on the consecutive ones submatrix problem. *Inf. Process. Lett.*, 83(3):163–166, 2002.

[HMPV00] Michel Habib, Ross M. McConnell, Christophe Paul, and Laurent Viennot. Lex-bfs and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. *Theor. Comput. Sci.*, 234(1-2):59–84, 2000.

[Jer87] Mark Jerrum. Two-dimensional monomer-dimer systems are computationally intractable. *Journal of Statistical Physics*, 48(1-2):121–134, 1987.

[JRTC12] Bradley R. Jones, Ashok Rajaraman, Eric Tannier, and Cedric Chauve. Anges: reconstructing ancestral genomes maps. *Bioinformatics*, 28(18):2388–2390, 2012.

[KLL$^+$13] Tsz Chiu Kwok, Lap Chi Lau, Yin Tat Lee, Shayan Oveis Gharan, and Luca Trevisan. Improved cheeger's inequality: analysis of spectral partitioning algorithms through higher order spectral gap. In *Proceedings of the 45th annual ACM symposium on Symposium on theory of computing*, pages 11–20. ACM, 2013.

[KS96] Haim Kaplan and Ron Shamir. Pathwidth, bandwidth, and completion problems to proper interval graphs with small cliques. *SIAM Journal on Computing*, 25(3):540–561, 1996.

[KT13] Evgeny Kapun and Fedor Tsarev. De bruijn superwalk with multiplicities problem is np-hard. *BMC Bioinformatics*, 14(S-5):S7, 2013.

[LOGT12] James R Lee, Shayan Oveis Gharan, and Luca Trevisan. Multi-way spectral partitioning and higher-order cheeger inequalities. In *Proceedings of the 44th symposium on Theory of Computing*, pages 1117–1130. ACM, 2012.

[LR99] Tom Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM (JACM)*, 46(6):787–832, 1999.

[MA91] Bojan Mohar and Y Alavi. The laplacian spectrum of graphs. *Graph theory, combinatorics, and applications*, 2:871–898, 1991.

[McC04] Ross M. McConnell. A certifying algorithm for the consecutive-ones property. In *SODA*, pages 768–777, 2004.

[Möh90] Rolf H Möhring. *Graph problems related to gate matrix layout and PLA folding*. Springer, 1990.

[Moh92] Bojan Mohar. Laplace eigenvalues of graphs - a survey. *Discrete Mathematics*, 109(1-3):171–183, 1992.

[MP11] Ján Manŭch and Murray Patterson. The complexity of the gapped consecutive-ones property problem for matrices of bounded maximum degree. *Journal of Computational Biology*, 18(9):1243–1253, 2011.

[MPC12] Ján Manŭch, Murray Patterson, and Cedric Chauve. Hardness results on the gapped consecutive-ones property problem. *Discrete Applied Mathematics*, 160(18):2760–2768, 2012.

[MPW$^+$13] Ján Manŭch, Murray Patterson, Roland Wittler, Cedric Chauve, and Eric Tannier. Linearization of ancestral multichromosomal genomes. In *CTW*, pages 169–173, 2013.

[MS99] Guy Mayraz and Ron Shamir. Construction of physical maps from oligonucleotide fingerprints data. *Journal of Computational Biology*, 6(2):237–252, 1999.

[MV80] S. Micali and V.V. Vazirani. An O $\left(\sqrt{|V|}|\mathrm{E}|\right)$ algorithm for finding maximum matching in general graphs. In *Foundations of Computer Science, 1980., 21st Annual Symposium on*, pages 17–27. IEEE, 1980.

[Nie06] Rolf Niedermeier. *Invitation to fixed-parameter algorithms*, volume 3. Oxford University Press Oxford, 2006.

[NP13] N. Nagarajan and M. Pop. Sequence assembly demystified. *Nature Review on Genetics*, 14(3):157–167, 2013.

[Orl13] Ludovic Orlando *et al.* Recalibrating *equus* evolution using the genome sequence of an early middle pleistocene horse. *Nature*, 499(7456):74–78, 2013.

[OTC11] Aïda Ouangraoua, Eric Tannier, and Cedric Chauve. Reconstructing the architecture of the ancestral amniote genome. *Bioinformatics*, 27(19):2664–2671, 2011.

[PSL90] Alex Pothen, Horst D Simon, and Kang-Pu Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal on Matrix Analysis and Applications*, 11(3):430–452, 1990.

[PW98] James Propp and David Wilson. Coupling from the past: a users guide. *Microsurveys in Discrete Probability*, 41:181–192, 1998.

[RMB$^+$09] Anna I. Rissman, Bob Mau, Bryan S. Biehl, Aaron E. Darling, Jeremy D. Glasner, and Nicole T. Perna. Reordering contigs of draft genomes using the mauve aligner. *Bioinformatics*, 25(16):2071–2073, 2009.

[RS84] Neil Robertson and P.D Seymour. Graph minors. iii. planar tree-width. *Journal of Combinatorial Theory, Series B*, 36(1):49–64, 1984.

[RTC13] Ashok Rajaraman, Eric Tannier, and Cedric Chauve. The genome of the medieval black death agent (extended abstract). *ArXiV*, abs/1307.7642, 2013. Full version in revision for *Bioinformatics*.

[Tre12] Luca Trevisan. Max cut and the smallest eigenvalue. *SIAM Journal on Computing*, 41(6):1769–1786, 2012.

[TS12] T. J. Treangen and S. L. Salzberg. Repetitive DNA and next-generation sequencing: computational challenges and solutions. *Nature. Rev. Genet.*, 13:36–46, 2012.

[Tuc70] Alan Tucker. Matrix characterizations of circular-arc graphs. Technical report, DTIC Document, 1970.

[Tuc72] A. Tucker. A structure theorem for the consecutive 1's property. *Journal of Combinatorial Theory, Series B*, 12(2):153–162, 1972.

[TV01] Prasad Tetali and Santosh Vempala. Random sampling of euler tours. *Algorithmica*, 30(3):376–385, 2001.

[Val79] Leslie G Valiant. The complexity of computing the permanent. *Theoretical computer science*, 8(2):189–201, 1979.

[Vuo10] Niko Vuokko. Consecutive ones property and spectral ordering. In *SDM*, pages 350–360, 2010.

[WMPS11] R. Wittler, J. Maňuch, M. Patterson, and J. Stoye. Consistency of sequence-based gene clusters. *Journal of Computational Biology*, 18(9):1023–1039, 2011.