Variants of the Consecutive-Ones Property Motivated by the Reconstruction of Ancestral Species

by

Murray Patterson

BSc. (Honours) Computer Science, Acadia University, 2003

MSc. Computing Science, Simon Fraser University, 2006

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

THE FACULTY OF GRADUATE STUDIES
(Computer Science)

The University Of British Columbia
(Vancouver)

January 2012

Abstract

The polynomial-time decidable Consecutive-Ones Property (C1P) of binary matrices, formally introduced in 1965 by Fulkerson and Gross [52], has since found applications in many areas. In this thesis, we propose and study several variants of this property that are motivated by the reconstruction of ancestral species.

We first propose the Gapped C1P, or the (k, δ)-Consecutive-Ones Property ((k, δ)-C1P): a binary matrix M has the (k, δ)-C1P for integers k and δ if the columns of M can be permuted such that each row contains at most k blocks of `1`'s and no two neighboring blocks of `1`'s are separated by a gap of more than δ `0`'s. The C1P is equivalent to the $(1, 0)$-C1P. We show that for every bounded and unbounded $k \geq 2, \delta \geq 1, (k, \delta) \neq (2, 1)$, deciding the (k, δ)-C1P is NP-complete [55]. We also provide an algorithm for a relevant case of the (2,1)-C1P.

We then study the (k, δ)-C1P with a bound d on the maximum number of `1`'s in any row (the maximum *degree*) of M. We show that the (d, k, δ)-Consecutive-Ones Property ((d, k, δ)-C1P) is polynomial-time decidable when all three parameters are fixed constants. Since fixing d also fixes k ($k \leq d$), the only case left to consider is the (d, k, ∞)-C1P (when δ is unbounded). We show that for every $d > k \geq 2$, deciding the (d, k, ∞)-C1P is NP-complete.

We also study the Consecutive-Ones Property with Multiplicity (mC1P), introduced by Wittler and Stoye [151]: a binary matrix M on columns $S = \{1, \ldots, n\}$ has the mC1P for *multiplicity vector* $\mathbf{m} : S \to \mathbb{N}$ if there is a sequence σ on S such that (i) σ contains each $s \in S$ at most $\mathbf{m}(s)$ times, and (ii) for each row r of M, the set of columns that have entry `1` in r form at least one subsequence of σ. We show that deciding the mC1P, and two restricted variants thereof, are NP-complete, for M having maximum degree 3 (6 for one of the variants), and for $\mathbf{m}(s) \leq 2$ for all

$s \in S$. We also give a tractability result for the mC1P that is motivated by handling telomeres in the reconstruction of ancestral species.

Finally, we study the Generalized Cladistic Character Compatibility (GCCC) Problem, a generalization of the Perfect Phylogeny Problem [137] introduced by Benham et al. [12]. We use the structure of the PQ-tree [21] associated with the C1P to give algorithms for several cases of the GCCC Problem.

Preface

This thesis is structured into six chapters. The first chapter gives a general overview of the C1P and the motivation for considering the four variants that we propose and study here. This was written specifically for the thesis by me with help from Cedric Chauve in structuring the content. Each of the four subsequent chapters is then dedicated to a particular variant. These four chapters form the results of this thesis, which have been published in several co-authored publications, as detailed below. The sixth chapter concludes this thesis with open questions and future work.

In Chapter 2, Cedric Chauve identified the (k,δ)-C1P and its motivation for studying this variant. The results of Sections 2.2 and 2.3 were found by Ján Maňuch and I, while Ján Maňuch wrote most of Section 2.2 and I wrote Section 2.3. The ideas of Section 2.4, with exception of Condition 8 were found by me, and this section was also written by me. Finally, the idea of the construction of Section 2.5 was mine, while I wrote most of this with some help from Ján Maňuch. All the results, with exception of Section 2.4 appear in our work Maňuch et al. [101]. Preliminary results on this appear in our published work Chauve et al. [29].

In Chapter 3, Cedric Chauve came up with the idea for the algorithm of Section 3.1, and also wrote most of this, which was expanded later by me. The results of Section 3.2 were then found by Ján Maňuch and I. Ján Maňuch came up with the idea of using a hypergraph covering problem to show NP-completeness of deciding the $(3,2,\infty)$-C1P, and wrote this up as well (Sections 3.2.1, 3.2.2 and 3.2.4). Generalizing this construction (Section 3.2.3) was then found by Ján Maňuch and I, while I wrote it up and Ján Maňuch supplied the figures. The result of Section 3.1 can be found in our work Maňuch et al. [101] (and in our work Chauve et al. [29]). The results of Section 3.2 are the subject of our published work Maňuch and Patterson [100], while preliminary results on this appear in our published work Maňuch and Patterson [99].

In Chapter 4, Wittler and Stoye [151] formally define the notion of the *m*C1P, and propose also the two variants of Section 4.2. All of the results of Sections 4.1 and 4.2 where found by Ján Maňuch and I, with some help from Roland Wittler. The ideas and work for the tractability result of Section 4.3 were then shared with Cedric Chauve, Ján Maňuch, Roland Wittler and I. In particular, Cedric Chauve and Roland Wittler worked on and wrote the subsection titled "The Case of a Single Multicolumn", while Ján Maňuch and I worked on and wrote the subsection titled "Completing the Proof of Theorem 51". All of the results of Sections 4.1 and 4.2 appear in our published work Wittler et al. [152], while the tractability result of Section 4.3 is the subject of our published work [31].

The work of Chapter 5 was an equal contribution of Ján Maňuch and I. The GCCC (at least its form) was first proposed in Benham et al. [12]. The results of Section 5.2 were then found and written by Ján Maňuch and I. The algorithm of Subsection 5.3.1 was found by me, and written with help from Ján Maňuch. In Subsection 5.3.2, Ján Maňuch came up with the idea of Lemma 63, while I came up with the idea of this struture based on PQ-trees [21, 106] for Lemma 65. This Subsection 5.3.2 was then written by me. Ján Maňuch and I then came up with the idea of Subsection 5.3.3, and Ján Maňuch wrote this. The results of Section 5.4 were then found and written by Ján Maňuch and I.

Table of Contents

Abstract . . . . **ii**

Preface . . . . **iv**

Table of Contents . . . . **vi**

List of Tables . . . . **ix**

List of Figures . . . . **x**

Glossary . . . . **xvii**

Acknowledgments . . . . **xix**

Dedication . . . . **xxii**

1 Introduction **1**

1.1 The Consecutive-Ones Property 2

1.1.1 An Introduction of the Consecutive-Ones Property 2

1.1.2 Background: Deciding the Consecutive-Ones Property . . 3

1.1.3 Applications of the Consecutive-Ones Property 6

1.2 The Reconstruction of Ancestral Gene Orders 8

1.2.1 A Basic Overview of the Reconstruction of Ancestral Gene Orders 8

1.2.2 Previous Approaches to Reconstructing Ancestral Gene Orders 11

1.2.3 Binary Matrices, the C1P and the Reconstruction of Ancestral Gene Orders (AGOs) 16

1.3 Computational Solutions for non-C1P Matrices 21

1.3.1 Transforming the Matrix to a C1P Matrix 21

1.3.2 Relaxing the C1P 21

1.3.3 Matrices of Bounded Degree 24

1.3.4 Matrices with Columns of Multiplicity 26

1.4 The Generalized Cladistic Character Compatibility Problem . . . 30

2 The Gapped Consecutive-Ones Property 32

2.1 Notation and Conventions 32

2.2 Fixing the Order of Selected Columns in a Matrix 33

2.3 The Complexity of Deciding the (k,δ)-C1P 34

2.3.1 The Complexity of Deciding the (k,δ)-C1P for every $k,\delta \geq 2$ 34

2.3.2 The Complexity of Deciding the $(k,1)$-C1P for every $k \geq 3$ 38

2.4 The (2,1)-C1P 39

2.4.1 The Algorithm 40

2.5 The Complexity of Deciding the (∞,δ)-C1P 46

2.5.1 The 3SAT(L:2,R:2) Problem 46

2.5.2 The Complexity of Deciding the $(\infty,1)$-C1P 47

2.5.3 The Complexity of Deciding the (∞,δ)-C1P 51

3 The Gapped Consecutive-Ones Property for Matrices of Bounded Maximum Degree 53

3.1 An Algorithm for Matrices of Bounded Maximum Degree 53

3.2 The (d,k,∞)-C1P 61

3.2.1 A Hypergraph Covering Problem 61

3.2.2 The 3-Uniform Hypergraph 1-Covering by Paths Problem 62

3.2.3 The *d*-Uniform Hypergraph *p*-Covering by Paths Problem 65

3.2.4 The Complexity of Deciding the (d,k,∞)-C1P 70

4 The Consecutive-Ones Property with Multiplicity 73

4.1 The Consecutive-Ones Property with Multiplicity (*m*C1P) 73

4.2 Two Variants of the *m*C1P 78
4.2.1 The Consecutive-Ones Property with Multiplicity for Framed Rows (*m*C1P(fr)) Variant 78
4.2.2 The Consecutive-Ones Property with Multiplicity for Nested Rows (*m*C1P(ne)) Variant 82
4.3 A Tractability Result for the Consecutive-Ones Property with Multiplicity 86
4.3.1 Preliminaries 86
4.3.2 A Tractable Case of Deciding the *m*C1P 89
4.3.3 Building a PQ-tree which Describes All Sequences that Satisfy the Consecutivity Requirement 97

5 The Generalized Cladistic Character Compatibility Problem 100
5.1 The Generalized Cladistic Character Compatibility (GCCC) Problem 101
5.2 Ordering Problems 103
5.3 Tractability Results 108
5.3.1 An Algorithm for Cases of the Single-Branch GCCC Problem 108
5.3.2 The Benhan-Kannan-Warnow (BKW) Case of the Single-Branch GCCC-NB (SB-GCCC-NB) Problem is Polynomial-Time Solvable 110
5.3.3 The $\{\{1\},\{2\},\{0,2\}\}$-Path GCCC-NB (P-GCCC-NB) Problem 113
5.4 Hardness Results 114

6 Conclusion 118
6.1 Chapter 2: The (k,δ)-C1P 119
6.2 Chapter 3: The (d,k,δ)-C1P 121
6.3 Chapter 4: The *m*C1P 123
6.4 Chapter 5: The GCCC Problem 124

Bibliography 126

List of Tables

Table 5.1 Complexity of all cases of the GCCC Problem for the character tree $0 \to 1 \to 2$ and set of states chosen from the set $\mathscr{Q} \subseteq \{\{0\},\{1\},\{2\},\{0,2\},\{0,1,2\}\}$. The BKW Case is marked with *. 102

List of Figures

Figure 1.1 (a) A binary matrix M that has the C1P. (b) A consecutive-ones (C1) order of M. (c) A binary matrix that does not have the C1P [145]. 2

Figure 1.2 (a) A binary matrix M that has does not have the C1P. (b) The bipartite graph G_M corresponding to M where the black (resp., white) vertices correspond to the columns (resp., rows) of M. Graph G_M contains the asteroidal triple g, h, j. 4

Figure 1.3 (a) A binary C1P matrix M. (b) The PQ-tree T_M for M. Here, T_M has internal (circular) P-nodes and (rectangular) Q-nodes, and leaf nodes for the set of columns of M. A leaf order of T_M obtained by taking any arbitrary (resp., forward or reverse) permutation of the children of a P-node (resp., Q-node) represents a C1 order of M. Note that the current configuration of T_M represents C1 order bdaecgf of M. (c) Another configuration of PQ-tree T_M representing C1 order dgcaebf of M. Note that $T(M)$ has $4! \cdot 2 \cdot 2 = 96$ configurations, and hence M has 96 C1 orders. 5

Figure 1.4 (a) A binary matrix M that does not have the C1P. Note that M is the matrix of Figure 1.3a with a fourth row added, causing this matrix to not have the C1P. (Note also, that M contains the forbidden submatrix of Figure 1.2a). (b) The PQR-tree T_M for M. Here, T_M has the additional third type of internal (diamond-shaped) R-node. An R-node represents a part of M which do not have the C1P (contains a conflicting set of columns of M). Note that the PQR-tree of the matrix formed by the first three lines of M is equivalent to that shown in Figure 1.3b. 6

Figure 1.5 A Sequence Tagged Site (STS) physical map of the kallikrein gene region. The positions of the markers are depicted along the top, and the clones are shown as horizontal lines. The markers were developed from clone insert ends (red) and kallikrein genes (blue). The unfilled squares on clones 338F22 and 003F08 show markers not analysed. (source: http://westnilevirus.okstate.edu/research/2004rr/13/13.htm) . 9

Figure 1.6 The alignment of a human genome against several mammals and a chicken genome. Here, the regions of each genome that code for the Apolipoprotein A1 gene (a gene that has an important role in lipid metabolism) are highly conserved, and hence similar, for all mammals. (source: http://www.lbl.gov/tt/techs/lbnl1690.html) 10

Figure 1.7 An illustration of the inference of syntenies for the ancestor common to set $S = \{\text{human, mouse, dog}\}$ of species with outgroup species chicken. A synteny is a group of markers that appears together in at least two species whose path goes through the considered ancestor. Here, the first synteny appears in the human and the dog, and the second is inferred from the chicken and the mouse, while the fourth one appears in all three species of S. These syntenies can be weighted according to how often they appear in the existing species, i.e, this fourth synteny would be weighted more heavily than the first two. (animal skeleton reproduced with permission from www.bigstock.com) 11

Figure 1.8 (a) Human-mouse nets [85] with human as the reference. Four mouse intervals are depicted, as ordered and oriented by the orthologous human segments. The second and third mouse intervals are adjacent (and appropriately oriented) on a mouse chromosome, and the intervening bases, if any, do not align to human, and are depicted by a thin line connecting these intervals. (b) The human-mouse, human-rat and human-dog nets for a segment of the human sequence, which illustrates the construction of orthology blocks (OB). (c) The construction of conserved segments (CS) from the fusion of runs of consecutive orthology blocks whenever the order and orientation of these blocks are conserved in each of the existing genomes. (source: Ma et al. [96]) 14

Figure 1.9 The set of Contiguous Ancestral Regions (CARs) for the Boreoeutherian ancestral genome (of human, rat, mouse and dog) constructed from the experiments of Ma et al. [96]. Numbers above bars indicate the corresponding human chromosomes. (source: Ma et al. [96]) 15

Figure 1.10 The binary matrix M corresponding to the set of sytenies inferred in Figure 1.7 and the PQ-tree T_M for M. Note that each CAR is a child of the root P-node r of T_M. 19

Figure 2.1 Possible positions of columns $\overline{2\delta+2}$ and $\overline{2\delta+3}$. 33

Figure 2.2 The structure of M_ϕ and the five rows encoding clause $c_2 = \{v_2 \vee \neg v_3 \vee v_1\}$. 37

Figure 2.3 The structure of the construction for a 3CNF formula ϕ on the set V of variables and C of clauses, along with the 3 rows encoding the clause $c_j = \{v_1 \vee v_2 \vee \neg v_3\}$. The blocks $b_1, \ldots, b_{|V|}$ correspond to the variables of ϕ in exactly the same way as in the construction of Subsection 2.3.1. The blocks $D_1, \ldots, D_{|C|}$ correspond to the clauses. Here, for $i \in \{1,2,3\}$, $\hat{r}_i$ is row r_i restricted to the columns of B_t, and P_t^1, P_t^3 (resp., P_t^2) are sets of permutations that do not place any `0` to the left (resp., right) of any `1` in B_t in rows r_1, r_3 (resp., r_2). It follows that all truth assignments to the literals of c_j are (2,1)-C1 orders except for the case when all 3 literals are false (c_j is not satisfied), since $P_t^1 \cap P_t^2 \cap P_t^3 = \emptyset$. Note that for each $i \in \{1, \ldots, |V|\}$, rows can be added to force the copy of variable block b_i on the left and right of the clause blocks to encode the same truth value. . . . 45
Figure 2.4 The structure of matrix M_ϕ. 49

Figure 3.1 (a) A simple dependency on 1-coverings of two touching hyperedges enforced by a copy of D (depicted as a diamond). (b) The 2-clause and (c) 3-clause gadgets for clause c_i. 63
Figure 3.2 (a) The variable gadget for variable with positive occurrences c_i^p and c_j^q and negated occurrence c_k^r in the clauses. The dashed edge is always picked in any valid 1-covering. (b) Grey edges are picked when this variable is set to *false* in a satisfying assignment of ϕ. (c) Grey edges are picked when the variable is set to *true*. 64
Figure 3.3 Hypergraph $D_{d,p}$: only one of the $\binom{|S|}{d-p}$ hyperedges is shown. 66
Figure 3.4 The path G' through vertex set $S \cup P$ that alternates between subpaths completely in S and completely in P. Some of the shown edges may be virtual. 67
Figure 3.5 Hyperedge h of $D_{d,p}$ which contains less than p edges from G' depicted in Figure 3.4. 67
Figure 3.6 A valid p-covering of $D_{d,p}$ in which vertex v has degree 1. . . 68

Figure 3.7 Vertices and hyperedges added to $\overline{H}$ to simulate the 3-edge $h = \{a,b,c\}$. The grayed diamonds depict copies of $D_{d,p}$. . . . 69

Figure 4.1 Graphical representations of the (a) 2-clause gadget and (b) 3-clause gadget for clause c_i. The multiplicity of the columns (resp., vertices) is indicated by the number of dots. Rows are depicted by ellipses surrounding two vertices or triangles surrounding three vertices, respectively. 75

Figure 4.2 Graphical representation of the variable gadget for variable x_ℓ with positive occurrences c_i^α and c_j^β and negated occurrence c_k^γ in the clauses. 76

Figure 4.3 Graphical representations of the (a) 2-clause gadget and (b) 3-clause gadget for clause c_i in the mC1P(ne) case. 84

Figure 4.4 Graphical representation of the variable gadget for variable x_ℓ with positive occurrences c_i^α and c_j^β and negated occurrence c_k^γ in the clauses in the mC1P(ne) case. 85

Figure 4.5 (a) Binary matrix M, with matched multirows. Let $\mathbf{m}(1) = \cdots = \mathbf{m}(5) = 1$ and $\mathbf{m}(a) = \mathbf{m}(b) = 2$: a and b are multicolumns and r_1, r_3 and r_4 are multirows. Row r_3 is not minimal, because it contains r_4. (b) The corresponding matrix $\hat{M}$. Since in $\hat{M}$, by definition $\hat{r}_i = r_i$ for all multirows r_i, the matched multirows are discarded. 87

Figure 4.6 (a) Binary matrix M, with matched multirows. Let $\mathbf{m}(c') = 2$. (b) PQ-tree belonging to the equivalence class $PQ_{\hat{M}}$. P-nodes are represented by circular nodes and Q-nodes by rectangular nodes. An example of a valid C1 order with multiplicity is $c'\,1\,2\,3\,4\,c'\,7\,8\,9\,5\,6$ which is obtained by taking the equivalent PQ-tree with frontier $1\,2\,3\,4\,7\,8\,9\,5\,6$ and inserting two copies of c' into the corresponding positions. Notice that inserting c' between 2 and 3 would break row r_2.
Illustration of Algorithm 2. LCA$(\hat{r}_1)$ and the respective segments of LCA$(\hat{r}_{3,4})$ are highlighted in gray and the respective paths are depicted by dashed lines. The upper left edge is contained in two paths. Here, $K_1 = 1$ and $K_2 = 1$, thus $K = 2 \leq \mathbf{m}(c') = 2$. 91
Figure 4.7 Augmented PQ-tree T' for the matrix given in Figure 4.6. (In fact, to get an augmented PQ-tree from the original PQ-tree shown in Figure 4.6, no modifications are necessary other than attaching leaf nodes labeled c' at appropriate locations.) Only the trees in the equivalence class of T' where the left side of the right Q-node is placed adjacent to the left Q-node have shortened frontiers that meet the multiplicity constraint ($\mathbf{m}(c') = 2$), for example, $c'\,1\,2\,3\,4\,c'\,7\,8\,9\,5\,6$. 97
Figure 4.8 Transformation rules for the LCAs to construct an augmented PQ-tree. An LCA and its parent node are replaced by the nodes shown on the right. The LCA (or the segment of an LCA, respectively) are highlighted in gray. 98
Figure 4.9 Transformation rules for bottom-up iteration to construct an augmented PQ-tree. A newly created Q-node and its parent node are replaced by the nodes shown on the right. 99
Figure 4.10 Special transformation rules for bottom-up iteration to construct an augmented PQ-tree. A newly created Q-node two levels below the root node and its parent node are replaced by the nodes shown on the right. 99

Figure 5.1 (a) A matrix M with entries from set $\{0, 1, 0^-, 0^+\}$. (b) PQ-tree PQ_M for M where the labels of the special zeros (0^- and 0^+) have been "forgotten". 112

Glossary

C1P Consecutive-Ones Property, a property of binary matricies

AGO Ancestral Gene Order

DNA Deoxyribonucleic Acid, a nucleic acid that contains the genetic instructions used in the development and functioning of all known living organisms (with the exception of RNA viruses)

RNA Ribonucleic Acid, one of the three major macromolecules (along with DNA and proteins) that are essential for all known forms of life

STS Sequence Tagged Site mapping, a type of physical mapping of DNA [108, 119]

CAR Contiguous Ancestral Region, a set of genes that remain together in some (reconstructed) ancestral genome [96]

k-**C1P** *k*-Consecutive-Ones Property

(k,δ)-**C1P** (k,δ)-Consecutive-Ones Property

(d,k,δ)-**C1P** (d,k,δ)-Consecutive-Ones Property

d-**UH**-*p*-**CP** *d*-Uniform Hypergraph *p*-Covering by Paths Problem

*m***C1P** Consecutive-Ones Property with Multiplicity

*m***C1P(fr)** Consecutive-Ones Property with Multiplicity for Framed Rows, a restricted variant of the *m*C1P

***m*C1P(ne)** Consecutive-Ones Property with Multiplicity for Nested Rows, another restricted variant of the *m*C1P

GCCC Generalized Cladistic Character Compatibility Problem, a generalization of the Perfect Phylogeny Problem [137]

GCCC-NB GCCC with non-branching character trees Problem, a special case of the GCCC Problem in which character trees have a single branch, i.e., each character tree T_α is $0 \to 1 \to \cdots \to |T_\alpha| - 1$

SB-GCCC-NB Single-Branch GCCC-NB Problem, the case of the GCCC with non-branching character trees (GCCC-NB) Problem where we restrict the solution (a phylogeny tree) to have only one branch starting at the root

P-GCCC-NB Path GCCC-NB Problem, the case of the GCCC-NB Problem where we restrict the solution (a phylogeny tree) to have only two branches starting at the root

BKW Benhan-Kannan-Warnow Case, a case of the GCCC Problem that is of particular intrest to the biological setting that motivated this problem

FPT Fixed Parameter Tractable

PTC Path Triple Consistency Problem

LEF-PTC Left Element Fixed Path Triple Consistency Problem

REF-PTC Right Element Fixed Path Triple Consistency Problem

OEF-TO One Element Fixed Total Ordering Problem

QC Quartet Consistency Problem

TO Total Ordering Problem

NAE-3SAT Not-All-Equal-3SAT

E-C1P Extended Consecutive-Ones Property, a property of matrices with entries from set $\{0, 1, 0^-, 0^+\}$

Acknowledgments

First, I would like to thank the members of my supervisory committee, Ján Maňuch, Cedric Chauve, Arvind Gupta and Anne Condon.

I am very grateful to Ján Maňuch for spending the time to regularly meet and discuss or work on problems, as well as to help proofread and improve my writing. Ján's dedication and great ideas have provided great motivation and direction through tough problems, where a solution seemed nowhere in sight. Indeed, most of what I know about doing scientific research has come through working with Ján, and without him as a mentor, I would not have been able to write this work, or even obtain its results.

I would like to thank Cedric Chauve for providing a wealth of ideas, and ultimately, problems that are relevant to the area of computational biology. Because Cedric is always on the frontier of important research in computational biology, this work contains results to many problems that are not only interesting, but also relevant to this area. Cedric's involvement in the research community has also provided great networking opportunities, one which has led to the position that I plan to hold after this degree.

I am grateful to Arvind Gupta for securing the majority of the funding for this research, as well as pointing me to viable career, research and funding opportunities. Of the members in my committee, I have been working with Arvind the longest, and during this time he has opened up many opportunities. I hope he considers it as good of an investment as I have.

Finally, of my supervisory committee, I would like to thank Anne Condon for helping to familiarize me with the Beta Lab and community at the University of British Columbia (UBC) after my transfer to UBC with my senior supervisor and his other students halfway through this degree. Anne has also provided very helpful feedback on my thesis.

I also wish to thank my external examiner Binhai Zhu, as well as university examiners William Evans and Paul Pavlidis for their helpful feedback.

I gratefully acknowledge funding from the following sources. First, I would like to thank the Natural Sciences and Engineering Research Council (NSERC) of Canada for providing three years of funding with a PGS-D3 Scholarship. For this, I also thank Arvind Gupta and Eugenia Ternovska (a supervisor of my masters thesis) who wrote letters that have no doubt determined greatly the success of my application for this award. I would like to thank the School of Computing Science at Simon Fraser University (SFU), the university where I obtained my masters and then began this degree, for its NSERC award top-up, as well as for several graduate fellowships. I also thank the Canadian Liquid Air Ltd. for a graduate scholarship. I would like to thank UBC for the remainder of its NSERC award top-up when I transferred there, as well as the tuition scholarship that comes with holding an NSERC award.

Two co-authors whom I have not mentioned above, but were important to several results in this work, are Roland Wittler and Jens Stoye, who introduced the concept of the Consecutive-Ones Property (C1P) with Multiplicity (Chapter 4 of this thesis). I thank them for this, as well as for helpful discussions about research and career opportunities.

I would like to thank my friends and colleagues at, or affiliated with, the Beta Lab at UBC, in particular Chris Thachuk for helping to familiarize me with UBC when I transferred here, as well as for many insightful discussions about research and career opportunities. I would also like to thank Bonnie Kirkpatrick, Jeff Sember and Frank Hutter at the Beta Lab for the helpful discussions, and for making the lab a lively and engaging place to work.

I would also like to thank the many friends and colleagues I have made at SFU, where I did my masters thesis, as well as a good amount of this degree. In particular, I would like to thank Phuong Dao for the many discussions about research and career opportunities as well as for all the social times we had, when we wanted to take a break from all of the work. I would like to thank George Ma for the encouragement and help during the low points of this degree, as well as for the helpful discussions about career and life. I would like to also thank Osama Saleh and Jingyun Chen for the lively and insightful discussions on many subjects; their senses of humour through tough times have been a good morale booster. I would also like to thank Fereydoun Hormozdiari, Iman Hajirasouliha, as well as the many other friends and colleagues at SFU whose discussions helped to shape my research and career plans, as well as who helped to encourage and support a community where such ideas can be discussed.

Finally, I am grateful to the many friends I have met in Vancouver, which does not exclude those mentioned above, who made the time enjoyable as I did this degree. I especially thank my family and friends back home in Nova Scotia for sticking with me and giving me the encouragement to undertake this challenging venture for such a long time, and in such a far away and different place.

Dedication

To my father, Kenzie Patterson, who always pushed for higher education. Since a PhD is the highest form of education one can obtain, I hope that he would be proud.

Chapter 1

Introduction

This thesis concerns variants of the Consecutive-Ones Property (C1P) of binary matrices. In particular, we define and study here four ways of generalizing the C1P in order to better model various scenarios of the reconstruction of ancestral species from a computational point of view. The first three of these are motivated by the reconstruction of Ancestral Gene Orders (AGOs) [27], while the last is motivated by the Generalized Cladistic Character Compatibility (GCCC) Problem [12].

First we give an overview of the C1P, some historical background of the property, and its applications. We then give a detailed overview of the reconstruction of AGOs and show its relation to the C1P. We then show that, among many other applications, the problem with the reconstruction of AGOs is that it often involves handling matrices that do not have the C1P. This then leads to the first contribution of this thesis: to offer three ways of generalizing the C1P in order to address this problem raised in the reconstruction of AGOs. Finally we introduce and motivate the GCCC Problem. We then propose our fourth and final variant of the C1P that leads to an algorithm for a case of this problem.

$$\begin{array}{ccccc} a & b & c & d & e \\ \hline \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & 0 \\ \mathbf{1} & 0 & \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 & \mathbf{1} & \mathbf{1} \\ \hline \end{array} \qquad \begin{array}{ccccc} c & a & b & d & e \\ \hline 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 \\ \mathbf{1} & \mathbf{1} & 0 & 0 & 0 \\ 0 & 0 & \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \hline \end{array} \qquad \begin{array}{ccccc} f & g & h & i & j \\ \hline \mathbf{1} & \mathbf{1} & 0 & \mathbf{1} & 0 \\ \mathbf{1} & 0 & 0 & 0 & \mathbf{1} \\ \mathbf{1} & 0 & \mathbf{1} & \mathbf{1} & 0 \\ \hline \end{array}$$

(a) **(b)** **(c)**

Figure 1.1: (a) A binary matrix M that has the C1P. (b) A C1 order of M. (c) A binary matrix that does not have the C1P [145].

1.1 The Consecutive-Ones Property

1.1.1 An Introduction of the Consecutive-Ones Property

Let M be a binary (`0`,`1`)-matrix with m rows and n columns. A *block* in a row of M is a maximal sequence of consecutive entries containing `1`. A *gap* is a sequence of consecutive `0`’s that separates two blocks, where the size of the gap is the length of this sequence of `0`’s. The *degree* of a row of M is the number of `1`’s in the row. The degree of a matrix M is the largest degree over all rows of M. In the first row of the matrix M of Figure 1.1a, the blocks are ab and d, while a gap of size one separates these two blocks. The degree of the second row of M in Figure 1.1a is 2, while the degree of M is 3.

A matrix M is said to have the C1P (for rows) if its columns can be permuted such that each row contains only one block (there are no gaps in this case). We call a permutation π of the columns of M that witnesses this property a consecutive-ones (C1) *order* of M; that the matrix M' resulting from this permutation is *consecutive*, or that it is *consecutive with respect to* π; and that M has the C1P, or is C1P. Further, we call the problem of deciding whether or not a binary matrix has the C1P the C1P Problem. Observe that the matrix M of Figure 1.1a has the C1P, while permutation cabde of its columns is a C1 order of this M, cf. Figure 1.1b, while the matrix of Figure 1.1c does not have the C1P.

According to Kendall [84], this property was first mentioned by Petrie, an archaeologist, in 1899. In 1951, Robinson [129], also an archaeologist, proposed several heuristic methods for the problem. The first polynomial-time algorithm for deciding the C1P was then introduced by Fulkerson and Gross [52] in 1965. Interestingly, it was a problem in genetics, cf. Benzer [13], that motivated these authors to study the C1P.

1.1.2 Background: Deciding the Consecutive-Ones Property

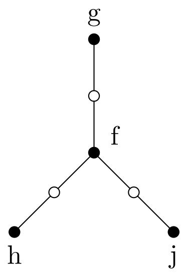
The early attempts at deciding the C1P started with the algorithm of Fulkerson and Gross [52]. In this work, Fulkerson and Gross [52] first compute the *overlap graph* for the set of rows of the binary matrix M. For each component (a tree, otherwise M does not have the C1P) of this graph, they then give a quadratic-time algorithm to incrementally build a permuted form of this component (which corresponds to a set of rows) that has the C1P. Following this, in 1969, Ryser [132] studied this problem and provided a generalization of the result of Fulkerson and Gross [52] for a class of matrices that have the *circular-ones property*.[1] In 1972, Tucker [145] then presented a forbidden submatrix characterization of binary C1P matrices. In this work, Tucker [145] shows that a binary matrix M has the C1P if and only if the bipartite graph G_M corresponding to M contains no asteroidal triple. Here, $G_M = (V_1, V_2, E)$, where V_1 (resp., V_2) is the set of columns (resp., rows) of M, and $(v_1, v_2) \in E$ if and only if column v_1 contains a `1` in row v_2 (cf. Figure 1.2). An asteroidal triple of a graph is a set of three vertices such that there is a path between any two of these vertices which avoids the neighborhood of the third vertex, cf. Figure 1.2b. This set of forbidden submitrices then comes directly from the set of bipartite (sub) graphs which contain an asteroidal triple. For example, the matrix of Figure 1.1c that does not have the C1P contains the submatrix obtained by removing column i, shown in Figure 1.2a, which is a forbidden submatrix because its corresponding bipartite graph, shown in Figure 1.2b, has an asteroidal triple. Until recently [18, 30, 39], the forbidden submatrix approach of Tucker was not seen as computationally useful, which is why people followed other approaches.

In 1976, Booth and Lueker [21] introduced the first linear-time algorithm for deciding this property. In Booth and Lueker [21], the authors introduced also a data structure called the *PQ-tree*, a linear-time constructible structure that encodes all C1 orders of a binary C1P matrix. See Figure 1.3 for an example of a PQ-tree:

[1] While we focus here on generalizations of the C1P other than the circular-ones property, refer to Dom [37] for details on this property.

$$\begin{array}{cccc} f & g & h & j \\ \hline \mathbf{1} & \mathbf{1} & 0 & 0 \\ \mathbf{1} & 0 & 0 & \mathbf{1} \\ \mathbf{1} & 0 & \mathbf{1} & 0 \end{array}$$

(a)



(b)

Figure 1.2: (a) A binary matrix M that has does not have the C1P. (b) The bipartite graph G_M corresponding to M where the black (resp., white) vertices correspond to the columns (resp., rows) of M. Graph G_M contains the asteroidal triple g, h, j.

In Figure 1.3a we have a binary C1P matrix, while Figures 1.3b and 1.3c give two configurations for the PQ-tree of this matrix. This work of Booth and Lueker [21] was a significant achievement in the history of deciding the C1P. In particular, the PQ-tree has since served as a useful tool in using the C1P for modelling problems in many settings. In this thesis, we use the structure of the PQ-tree to obtain several of our algorithmic results. There would then be a break in research on deciding the C1P for more than ten years after this milestone result of Booth and Lueker [21].

Year 1989 showed a renewed interest in research on deciding the C1P with the result of Korte and Möhring [87]. Indeed, while the structure of the PQ-tree is very elegant and simple, the algorithm in Booth and Lueker [21] for constructing it is quite complicated. This motivated Korte and Möhring [87] to introduce MPQ-trees (modified PQ-trees), where the internal (P- and Q-) nodes contain some additional information, which makes these trees simpler to construct. In 1992, Hsu [73] ([76]) also presented a linear-time algorithm to test for the C1P without using PQ-trees, however its implementation is still quite complicated. In 1998, Meidanis et al. [106] proposed a new theory of the C1P which formalizes many concepts alluded to in other works, such as *orthogonality* of two rows in a binary matrix [21, 52, 73, 76, 113]. In addition to this new theory, the authors of Meidanis et al. [106] also introduce a new structure called the PQR-tree, which exists for any instance of a binary matrix; it generalizes the PQ-tree in that a PQR-tree

$$\begin{array}{|ccccccc|}
a & b & c & d & e & f & g \\
\hline
\mathbf{1} & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\
\mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} & 0 & 0 \\
0 & 0 & \mathbf{1} & 0 & 0 & 0 & \mathbf{1} \\
\hline
\end{array}$$

(a)

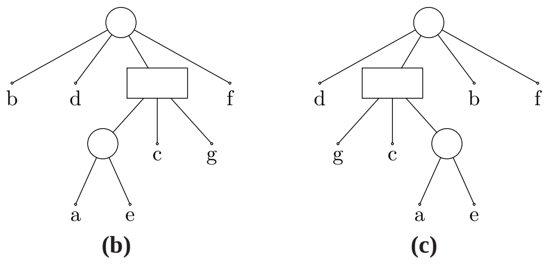


Figure 1.3: (a) A binary C1P matrix M. (b) The PQ-tree T_M for M. Here, T_M has internal (circular) P-nodes and (rectangular) Q-nodes, and leaf nodes for the set of columns of M. A leaf order of T_M obtained by taking any arbitrary (resp., forward or reverse) permutation of the children of a P-node (resp., Q-node) represents a C1 order of M. Note that the current configuration of T_M represents C1 order bdaecgf of M. (c) Another configuration of PQ-tree T_M representing C1 order dgcaebf of M. Note that $T(M)$ has $4! \cdot 2 \cdot 2 = 96$ configurations, and hence M has 96 C1 orders.

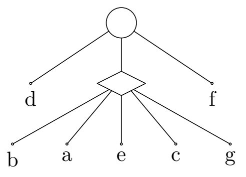
for a C1P matrix is a PQ-tree. See Figure 1.4a for an example of a PQR-tree. In 2000, Habib et al. [64] gave a very simple algorithm for deciding the C1P which is based on partition refinement.

In 2003, Hsu and McConnell [78] introduced a remarkable simplification for building PQ-trees. Here, these authors introduced *PC-trees*, a structure that is much more straightforward to construct, but which encodes all circular-ones orders of a binary matrix that has the circular-ones property. However, a binary matrix M that has the C1P has also the circular-ones property, and moreover, there is an easy way to modify the PC-tree for M so that it yields the PQ-tree for M [37]. In 2004, McConnell [102] proposed the first linear-time certifying algorithm for deciding the C1P, that is, if a matrix M is not C1P, the algorithm outputs a certificate of size linear in M that verifies this.[2] In McConnell [102], the author also provided a slightly different type of structure than the PQR-tree based on partitive families, called the Generalized PQ-tree, which exists for any instance of a binary matrix; again, a generalized PQ-tree is a PQ-tree for a C1P matrix. Most recently, in 2010, Blin et al. [18] developed a faster algorithm for finding the forbidden submatrices

[2]Refer to Kratsch et al. [90] for more details on such certificates.

$$\begin{array}{ccccccc}
a & b & c & d & e & f & g \\
\hline
\mathbf{1} & 0 & 0 & 0 & \mathbf{1} & 0 & 0 \\
\mathbf{1} & 0 & \mathbf{1} & 0 & \mathbf{1} & 0 & 0 \\
0 & 0 & \mathbf{1} & 0 & 0 & 0 & \mathbf{1} \\
0 & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 & 0
\end{array}$$

(a)



(b)

Figure 1.4: (a) A binary matrix *M* that does not have the C1P. Note that *M* is the matrix of Figure 1.3a with a fourth row added, causing this matrix to not have the C1P. (Note also, that *M* contains the forbidden submatrix of Figure 1.2a). (b) The PQR-tree T_M for *M*. Here, T_M has the additional third type of internal (diamond-shaped) R-node. An R-node represents a part of *M* which do not have the C1P (contains a conflicting set of columns of *M*). Note that the PQR-tree of the matrix formed by the first three lines of *M* is equivalent to that shown in Figure 1.3b.

of Tucker [145]. Refer to the works of Michael Dom [36, 37] for a nice survey of the C1P and its algorithmic aspects, respectively.

1.1.3 Applications of the Consecutive-Ones Property

The Consecutive-Ones Property has had a rich set of applications since its introduction. Indeed, according to Kendall [84], Petrie's interest in this property in 1899 was motivated by the application of the seriation of archaeological data [72, 84, 129]. The C1P appears in many other practical applications, such as scheduling [67, 70, 89, 147], information retrieval [54, 88] and circuit and railway design/optimization [46, 104, 105, 131]. Essentially, the C1P finds applications in any problem where one needs to linearly arrange a set of objects subject to the constraint that objects in a given subset must appear consecutively in this order.

Since binary matrices can be represented as graphs and vice versa, the C1P has close connections to graph theory, in particular to interval graphs and their recognition [34, 74, 77, 90]. Indeed, much of the progress in deciding the C1P was a result of research on interval graphs [21, 52, 64, 87]. The C1P also plays an important role in the area of solving (integer) linear programs, in terms of both its direct application to practical linear programming problems [6, 70, 71, 147], or how it relates to linear programming from a more theoretical point of view [115, 116, 136]. From a complexity theoretic point of view, there are many problems on matrices that are in general NP-hard that become polynomial-time solvable when the input has the C1P [33, 112], such as problems in railway optimization and scheduling [105, 147]. This has also been shown in the study of covering problems such as set cover, as well as geometric covering problems such as rectangle stabbing [36, 43, 104, 131].

The C1P has also found applications in quite a few areas of (computational) molecular biology as different technologies developed over time. Since this application is the subject of this thesis, we illustrate this in more detail in the next few paragraphs. One of its first applications to molecular biology was in the study of the composition of genes [13, 52, 92]. By 1926, it was already known from Morgan [109] that genes are arranged linearly on a chromosome. However, by 1959, genetic analysis technology was advanced enough [124] that Benzer [13] was able to perform a series of experiments aimed at verifying whether a gene is also a linear arrangement of its components. While the primitive genetic maps of the set of components that Benzer [13] produced did not altogether exclude non-linear arrangements, the assumption of a linear arrangement seemed to be the most probable fit given this data. This would be the first, very crude, form of physical mapping. It was six years later, in Fulkerson and Gross [52], in the study of this problem by these authors that they formulated the set of these components from the experiments of Benzer [13] as a binary matrix M, where each component represented a row M. The set of components then has a linear arrangement exactly when matrix M has the C1P, hence formally defining this notion of the C1P in Fulkerson and Gross [52] and also introducing the first polynomial-time algorithm for deciding the C1P. Of course, today it is common knowledge that a gene is a linear arrangement of its components, but in Benzer [13], this was an exciting result that provided the first insights into the finer structure of genes.

More recently, when the technology allowed scientists to begin constructing, en masse, highly accurate physical maps [93] of hybridization data, with the aim of sequencing specific DNA strands, it introduced new computational challenges [7, 8], some of which have been overcome by very applied approaches [32, 58, 94], while several theoretical works exist on the subject [4, 5, 55, 149]. Since a DNA strand is too long to study in its entirety (i.e., the human chromosome contains about 10^8 base pairs [5]), it is broken into fragments, or *clones*, and the goal of physical mapping is to reconstruct the DNA strand given a collection of overlapping clones of the strand. A popular approach of the time was Sequence Tagged Site (STS) mapping [108, 119]. In this approach, relatively short substrings called *markers* (or probes) are extracted from the DNA strand itself, but are sufficiently long, however, that it is highly unlikely to occur twice on the same strand. Given the information as to which clones contain which markers, the goal is then to find an order of markers in such a way that subsets of markers that appear on the same clone appear consecutively in this order, i.e., one possible reconstruction of this DNA strand. See Figure 1.5 for an example of an STS physical map. Consider the binary matrix M where we have a column for each marker, and a row for each subset of markers that appear on the same clone (i.e., a row with a `1` in the column corresponding to each marker in this subset). It follows that we can find an order of markers satisfying the above condition if and only if M has the C1P. In the next section, we introduce in detail an application in the area of molecular biology, namely the reconstruction of AGOs, the application that has motivated the definition and study of the several relaxed versions of the C1P that are the subject of this thesis.

1.2 The Reconstruction of Ancestral Gene Orders

1.2.1 A Basic Overview of the Reconstruction of Ancestral Gene Orders

The area of comparative genomics concerns the relationship between the structure and function of genomes across sets of different species. This involves the analysis of the information provided by the signatures of selection in an attempt to understand the evolutionary processes that act on these genomes. Studies in this area have shown that conserved regions between the genomes of a set of species often contain functionally or evolutionarily associated genes [35, 118]. See Figure 1.6 for an example. From this discipline, and the existing data that has been generated, comes the natural question of inferring the structure of ancestral genomes, or Ancestral Gene Orders (AGOs). A set of closely related species, such as mammals in Figure 1.6 have many regions that are common, or at least similar. We can use this

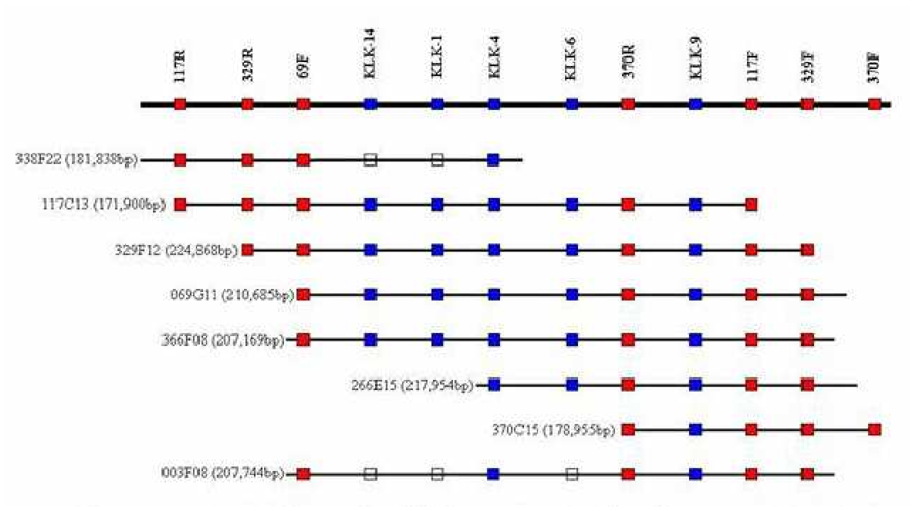


Figure 1.5: A STS physical map of the kallikrein gene region. The positions of the markers are depicted along the top, and the clones are shown as horizontal lines. The markers were developed from clone insert ends (red) and kallikrein genes (blue). The unfilled squares on clones 338F22 and 003F08 show markers not analysed. (source: http://westnilevirus.okstate.edu/research/2004rr/13/13.htm)

commonality to reconstruct the AGOs for this set of species.

Given the genomes for a set S of existing species and a set of genomic markers (such as markers obtained from STS physical mapping, for example, genes), the reconstruction of AGOs is to infer possible orders of these markers in the chromosomes of some ancestor common to S. This assumes that a phylogenetic tree T is given, with the existing species S at the leaves of this tree, and the common ancestor is the extinct (unsequenced) species at the internal node of T that is common to set S. Note, that T may contain some less closely related *outgroup* species (leaves that are not in S), and, in fact, this is a good practice, as the information they provide helps to produce more accurate reconstructions [27, 96]. As an auxiliary step to reconstructing AGOs, we first infer a set of *syntenies*, taking from the terminology of Chauve and Tannier [27], i.e., groups of markers that are believed to appear together in this ancestor, cf. Figure 1.7 for an illustration of this. An AGOs is then any order of the markers such that each group of markers in a synteny appears to-

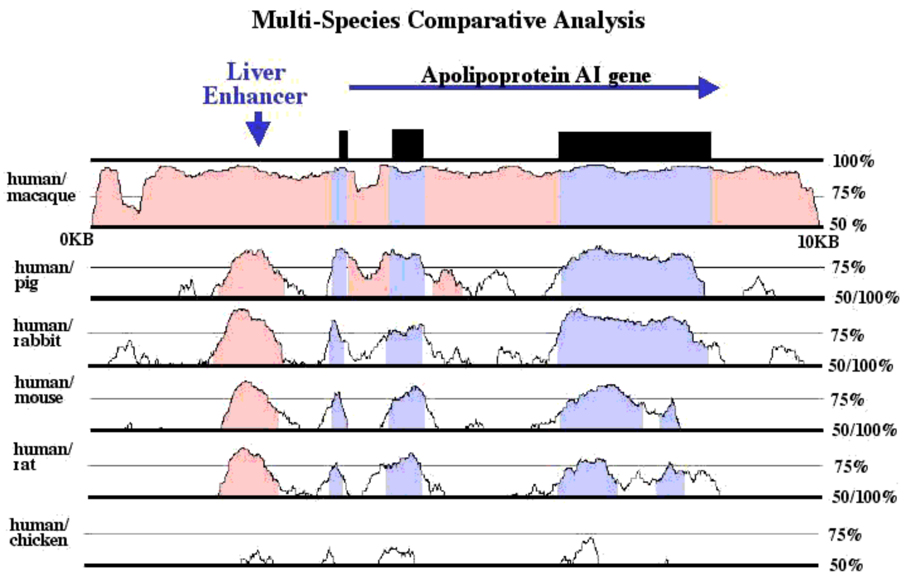


Figure 1.6: The alignment of a human genome against several mammals and a chicken genome. Here, the regions of each genome that code for the Apolipoprotein A1 gene (a gene that has an important role in lipid metabolism) are highly conserved, and hence similar, for all mammals. (source: http://www.lbl.gov/tt/techs/lbnl1690.html)

gether in this order. The value of reconstructing AGOs is that it can give us insights into the biology, ecology, and evolution of extinct species [26, 56]. Experimentally, at least for proteins, the reconstruction of ancestral proteins has led to the discovery of new biochemical functions that have been lost in modern proteins [80, 133]. Since the input to this problem is a phylogeny tree T, this area is closely related to phylogenetics [45] (constructing a phylogeny tree for a set of species, etc.). There are also studies of reconstructing phylogenies for a set of existing species given AGO data as well as computing both simultaneously [1].

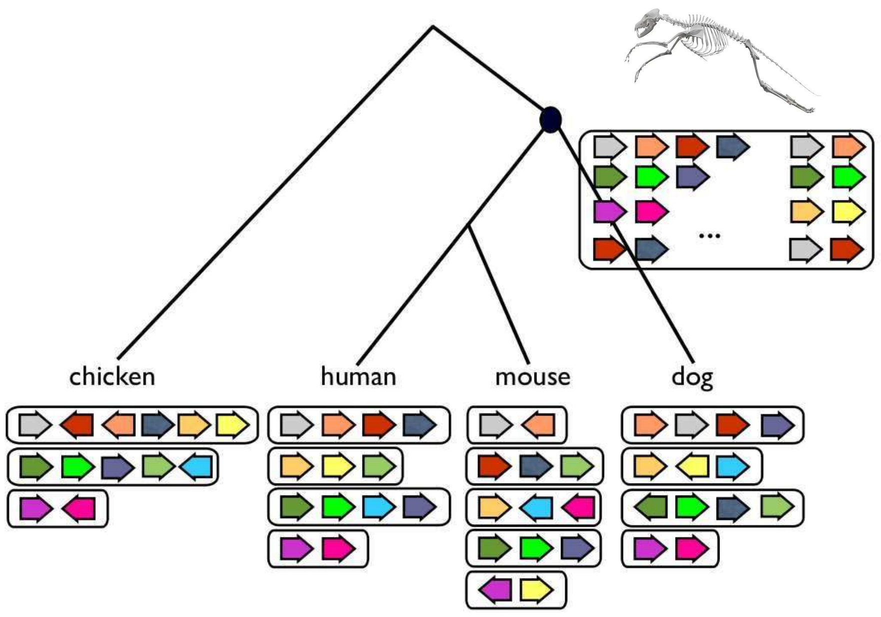


Figure 1.7: An illustration of the inference of syntenies for the ancestor common to set $S = \{\text{human, mouse, dog}\}$ of species with outgroup species chicken. A synteny is a group of markers that appears together in at least two species whose path goes through the considered ancestor. Here, the first synteny appears in the human and the dog, and the second is inferred from the chicken and the mouse, while the fourth one appears in all three species of S. These syntenies can be weighted according to how often they appear in the existing species, i.e, this fourth synteny would be weighted more heavily than the first two. (animal skeleton reproduced with permission from www.bigstock.com)

1.2.2 Previous Approaches to Reconstructing Ancestral Gene Orders

While the problem of reconstructing AGOs has been studied even as early as 1936 for simpler organisms such as insects [140], cytogenetics technology such as chromosome painting allowed scientists to start reconstructing more complex organisms such as mammals [50, 127, 141, 142, 150, 153] in the early to mid 2000's. At roughly the same time, because physical maps for different species became available [5, 108, 119], many bioinformatics methods for reconstructing AGOs from physical mapping data also began to appear [22–24, 111]. The benefit of bioinformatics methods over cytogenetics methods is that they produce AGOs at a much higher resolution. However, since physical mapping is still a young field, there are fewer such existing genome sequences available [44, 110, 126]. Since physical maps continue to be generated at an explosive rate (one reason being the drop in the cost of next generation sequencing technology) it is expected that bioinformatics methods will be the dominating technology for reconstructing AGOs. These bioinformatics methods use various differing approaches in processing data from physical maps. However, scientists started to notice a divergence between some of the bioinformatics methods that use a parsimony approach in terms of evolutionary events (reversals, translocations, fusions and fissions), in particular, the works [22, 111], with cytogenetics studies [51]. However, in 2006, the first bioinformatics approach to this problem appeared in Ma et al. [96] that, when applied to mammalian genomes, gave results that were more in agreement with cytogenetics methods, while exhibiting few points of divergence [130]. We present this important result in more detail in the next paragraph.

Given the genomes for a set S of existing species (in their experiments, S consists of human, mouse, rat and dog, while they use the two outgroup species chicken and opossum), and phylogeny tree T containing S, the approach of Ma et al. [96] is to first segment the multispecies alignment of S with the human genome as a reference (or more precisely, nets, cf. Kent et al. [85]) to build a set of *orthology blocks* [96]. Orthology blocks are essentially regions that are common (regions that are of some minimum size, here 50kb [96], that meet a certain similarity threshold) among all species in S. From these orthology blocks, Ma et al. [96] then compute *conserved segments*, that is, sequences of orthology blocks that remain together and in the same order in all species in S, see Figure 1.8. Finally, from the set of pairs of conserved segments, where each pair appears adjacent in some species of S, they extract a maximal unambiguous subset of adjacencies to construct Contiguous Ancestral Regions (CARs). In order to do this, they employ a method analogous to Fitch [48] to find the most parsimonious scenario for each of these adjacencies. This has the effect of assigning each adjacency a weight between 0 and 1, where the weight is the measure of confidence that this adjacency appears also in the ancestor. The set of outgroup species (here, chicken and opossum) is used to improve the accuracy of this step. Consider now the graph[3] $G = (V, E)$, where the vertex set V is the set of conserved segments, and set E of (weighted) edges is this set of weighted adjacencies. Since the goal to infer a set of AGOs, they construct a graph G' incrementally by selecting edges from E in order of decreasing weight, skipping over any edge in this order that creates either

(a) a vertex of degree larger than two, or

(b) a cycle,

in the current G'. At the end of this process, G' should be a union of disjoint paths, where any layout of these paths on a line represents a potential AGO for this set S of species. Here, it is each disjoint path, or rather its set of conserved segments that represents a CAR. Figure 1.9 represents the set of CARs constructed in the experiments of Ma et al. [96]. The mapping of these CARs (cf. Figure 1.9) onto the chromosomes of the human show quite a similarity, which is expected, as these CARs essentially represent ancestral chromosomal segments.

While this approach of Ma et al. [96] uses a parsimony method to weight each adjacency, there are no assumptions on any evolutionary events, nor is each CAR even guaranteed to be an ancestral whole chromosome, rather their approach is *model-free*, taking from the terminology of Adam et al. [1]. Indeed, the model-free approach avoids computing any global parsimony in terms of evolutionary events such as reversals, translocations, fusions and fissions, which is what all the methods whose results diverge with those of cytogenetics studies [51] rely on. This, and the fact that Ma et al. [96] is the first bioinformatics method to agree well with cytogenetics methods [130], suggests that a model-free approach is a step in the right direction. In the next subsection, we present a model-free framework for reconstructing AGOs based on the C1P of binary matrices. Note that model of adjacencies, used here in Ma et al. [96], is the special case of degree 2 binary matrices. Indeed, with the method of Ma et al. [96], the link between the C1P and the reconstruction of AGOs started to become explicit. The approach we propose generalizes this method of Ma et al. [96] (in one sense, that it concerns matrices

[3]Note that in Ma et al. [96], they consider a directed graph, however the principle is the same. This detail is left out to ease the summary of this method.

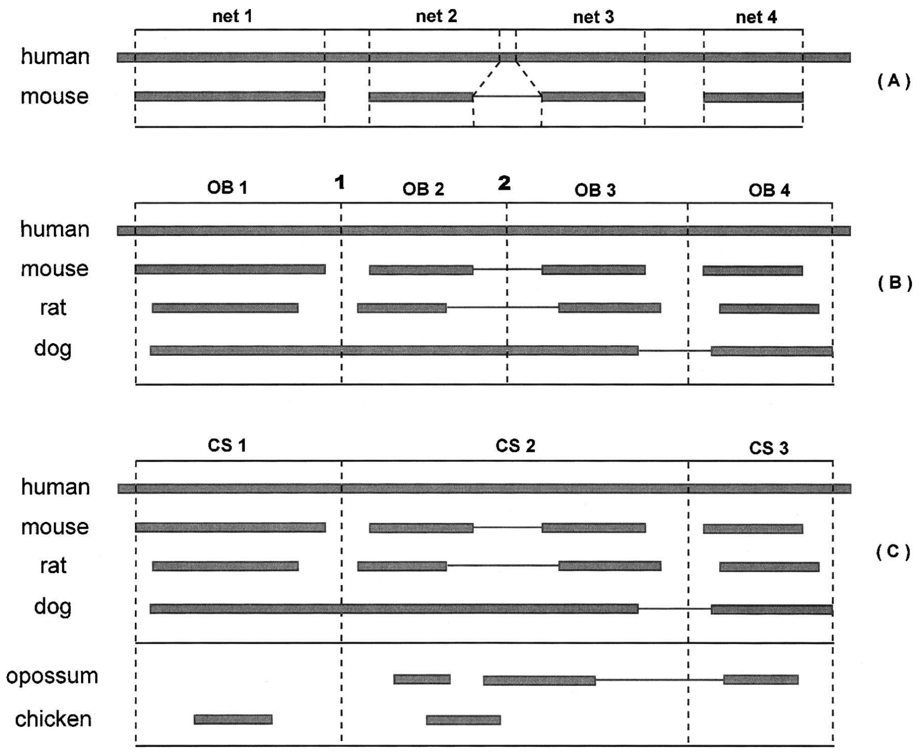


Figure 1.8: (a) Human-mouse nets [85] with human as the reference. Four mouse intervals are depicted, as ordered and oriented by the orthologous human segments. The second and third mouse intervals are adjacent (and appropriately oriented) on a mouse chromosome, and the intervening bases, if any, do not align to human, and are depicted by a thin line connecting these intervals. (b) The human-mouse, human-rat and human-dog nets for a segment of the human sequence, which illustrates the construction of orthology blocks (OB). (c) The construction of conserved segments (CS) from the fusion of runs of consecutive orthology blocks whenever the order and orientation of these blocks are conserved in each of the existing genomes. (source: Ma et al. [96])

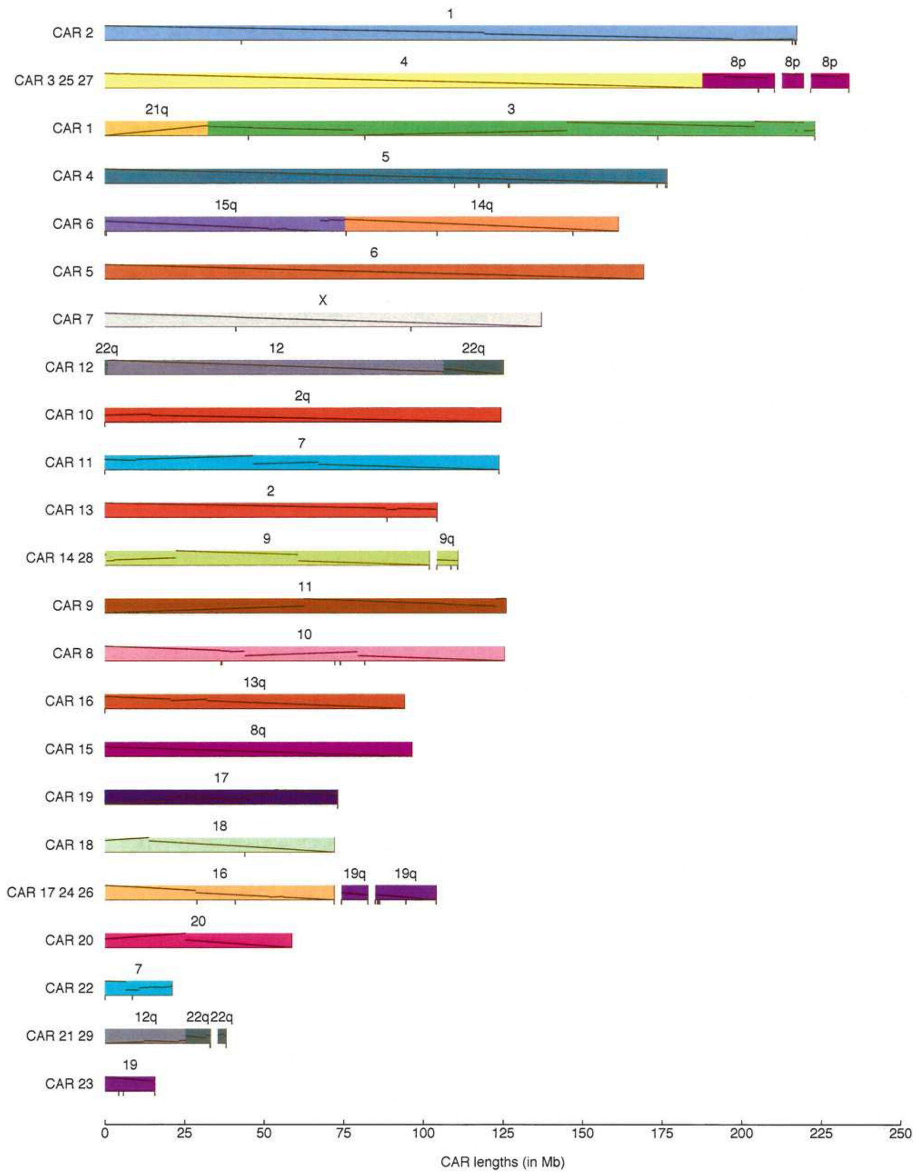


Figure 1.9: The set of CARs for the Boreoeutherian ancestral genome (of human, rat, mouse and dog) constructed from the experiments of Ma et al. [96]. Numbers above bars indicate the corresponding human chromosomes. (source: Ma et al. [96])

of degree larger than 2), and is the state of the art in terms of methodologies for reconstructing AGOs.

1.2.3 Binary Matrices, the C1P and the Reconstruction of AGOs

We now outline the approach for reconstructing AGOs based on the C1P of binary matrices that formalizes and generalizes the principles used in several computational [1, 96] as well as the cytogenetics studies [127, 150, 153]. This approach can be broken down into the following two steps. The first is a data acquisition phase: where we compute from the alignments of these genomes a set (or alphabet) of genomic markers $\mathscr{L} = \{1, \ldots, n\}$. From this set $\mathscr{L}$ of genomic markers we then compute the groups of markers (syntenies) that are believed to be contiguous in the ancestral genome. Here, we represent the set of syntenies with a binary matrix M on the set of columns $\mathscr{L}$ where for each synteny $X \subseteq \mathscr{L}$, we have a row in M with a `1` in every column of X, and `0`'s everywhere else. In general each synteny (row of M) can also be weighted according to the confidence that it appears in the ancestral genome. The second step of this approach consists of transforming this matrix M into a C1P matrix. It is this second step that we concentrate on in this thesis, however, we will see later that the way to approach this second phase depends very much upon the data acquisition phase. Indeed, from a computational point of view, this approach is closely related to physical mapping: if M has (or can be transformed into a matrix that has) the C1P, then we can find an order of markers that represents an AGO. Because syntenies of markers are naturally represented by binary matrices in this way, it also follows that there can be many AGOs that are consistent with M. This set of AGOs can be encoded in a compact way with some uncertainty by the PQ-tree for (the possibly transformed) M, which is another benefit of C1P-based approach.

The first work to represent AGOs with PQ-trees appeared in 2004. Bergeron et al. [15] used a Fitch-like [48] approach to find a most parsimonious scenario for the set of intervals (sytenies) defined by this PQ-tree. This work was quite preliminary however, and the experiments were performed on fairly basic chloroplast genome data. A year later, in 2005, Landau et al. [91] also use PQ-trees for ancestral genomes, but also in a parsimony context. Here, Landau et al. [91]

also suggest a way of representing duplicated genes (genes with multiplicity, which we will cover later in this thesis), but show only how this approach works on some experimental data. Following this, in 2006, Parida [121] improved on the result of Landau et al. [91] by using a PQ-tree where some of the internal nodes are oriented, to help to uniquely construct the orders it encodes, as well as a branch-and-bound scheme for outputting all solutions, rather than just the most parsimonious solution. Again, while the concept of Parida [121] is on the right track, they only give preliminary experimental results to test this concept. In 2007, the work of Adam et al. [1] also considered representing AGOs with PQ-trees. Here, they are concerned with computing the phylogeny and the AGOs, where they frame it as solving the Steiner Tree Problem. While they perform experiments only on fairly basic chloroplast genome data as well, this is the paper that introduces the model-free approach to using bioinformatics methods for reconstructing AGOs. Note that, while this is not made explicit, Ma et al. [96] also represent AGOs with PQ-trees. In Ma et al. [96], $\mathscr{L}$ is their set of conserved segments, and M stores the set of adjacencies (i.e., M has degree 2). This union of disjoint paths that they build is then equivalent to a PQ-tree with a P-node as the root r, where each child of r, containing only Q-nodes (since M has degree 2) corresponds to a path (or CAR). Next we detail the work of Chauve and Tannier [27], where this two step approach for reconstructing AGOs based on the C1P and PQ-trees was first developped.

Here we give some details of the method of Chauve and Tannier [27]. While this approach generalizes the approach of Ma et al. [96] (for one thing, Chauve and Tannier [27] consider matrices of degree larger than 2), these approaches are very similar in spirit. Here, given the markers for set S of species (and possibly some outgroup species) with phylogenetic tree T on S (and the outgroup species), they first compute the set $\mathscr{L}$ of markers. Here they mention that markers can be genes from whole genome alignment methods, orthologous genes, or various others (from comparative maps [111] or virtual hybridization [10] for example). From the input representation of S, Chauve and Tannier [27] compute (maximal) sets of markers, i.e., syntenies, that appear consecutively[4] in at least two

[4]Note that, more precisely, Chauve and Tannier [27] compute a set of gene teams [9, 95]: syntenies, as we have defined them here are gene teams for $\delta = 1$ [27]. We leave these details out to ease the explanation of the principles of this approach of Chauve and Tannier [27].

species from S, where the path in T between these two species goes through the node for the ancestor which we wish to reconstruct. These syntenies are weighted using the same principle in Ma et al. [96] for weighting adjacencies, and outgroup species are also used to improve this step. In fact, the set of syntenies inferred in Figure 1.7 is exactly what the method of Chauve and Tannier [27] would obtain. Note that, since an adjacency is a synteny of size two, this method is more general than that of Ma et al. [96]. One reason for considering these more general syntenies is that it is closer to the methods [127, 150, 153] on cytogenetics data. Indeed, the inference of syntenies in Chauve and Tannier [27] is a bioinformatics version of the hybridization used by cytogeneticists, which explains the convergence between these two approaches. Chauve and Tannier [27] represent the set of syntenies with a binary matrix M on the set of columns $\mathscr{L}$ where for each synteny $X \subseteq \mathscr{L}$, they have a row in M with a `1` in every column of X, and `0`'s everywhere else. We now outline the second step of the approach of Chauve and Tannier [27], transforming M into a C1P matrix.

Given binary matrix M, constructing an AGO for S then corresponds to finding a linear order of $\mathscr{L}$, such that each X appears consecutively in this order, i.e., a C1 order of M. In fact, all AGOs for S can be represented by building the PQ-tree T_M for M. Here, the set of CARs for S will be the children of the root node r of T_M (as it was in Ma et al. [96], however they can contain also P-nodes now, as each row of M has degree larger than 2 in general). It is here that the C1P plays an important role in the reconstruction of AGOs of Chauve and Tannier [27], i.e., that they can represent sets of CARs with a PQ-tree [21]. Indeed, for the set of syntenies inferred in Figure 1.7, the matrix (which is C1P) for this set is given along with the PQ-tree T_M for M in Figure 1.10. However, M rarely has the C1P as we will see later, and so Chauve and Tannier [27] do the following to build this PQ-tree (implicitly transforming M into a C1P matrix). At this point, they could employ the greedy heuristic of Ma et al. [96] of incrementally building a PQ-tree by selecting syntenies in order of decreasing weight, and skipping over any synteny that creates a conflicting set in the collection of currently selected syntenies. Rather than doing this, however, they build first a generalized PQ-tree (a PQR-tree [106], or the generalized PQ-tree from McConnell [102]), and then find a subset of sytenies (rows of M) of maximum cumulative weight, such that the matrix M' of this subset has

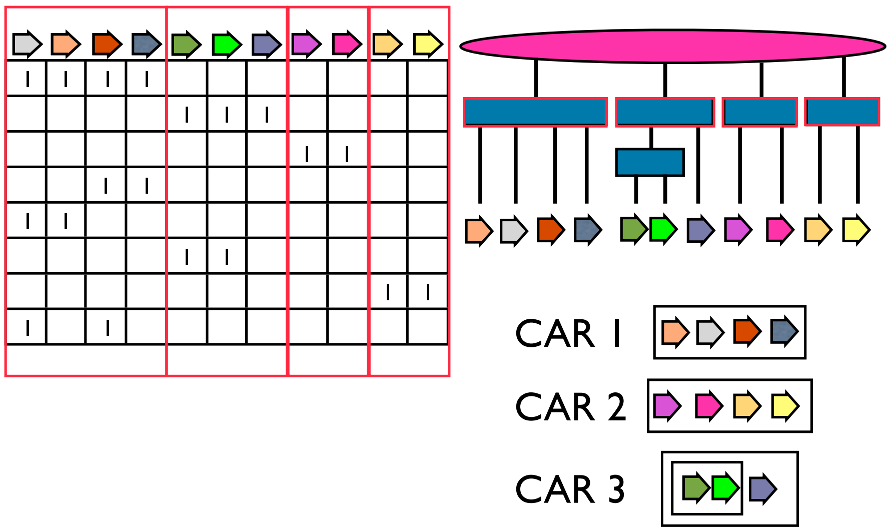


Figure 1.10: The binary matrix M corresponding to the set of sytenies inferred in Figure 1.7 and the PQ-tree T_M for M. Note that each CAR is a child of the root P-node r of T_M.

the C1P, i.e., the generalized PQ-tree for M' is a PQ-tree. While this approach is not greedy, it is the combinatorial optimization problem known as the Consecutive-Ones Submatrix Problem. Here, Chauve and Tannier [27] use the structure of this generalized PQ-tree for M to design an efficient branch-and-bound algorithm for this problem.

In experiments, the method of Chauve and Tannier [27] agrees well with all of the cytogenetics studies [50, 127, 141, 142, 150, 153] as well as with the work of Ma et al. [96], while disagreeing with the same approaches (that are not model-free) that Ma et al. [96] disagrees with. However, different experiments (from data at different levels of resolution, or variations on the input phylogeny T) show that the approach of Chauve and Tannier [27] is more stable in general than that of Ma et al. [96]. One reason for this is due to the fact that, while CARs from syntenies are less well-defined than those of adjacencies (they are degree larger than two), they are better supported because every computed synteny appears in at least two existing species whose path in T goes through the considered ancestor. Another reason is likely due to the fact that in certain cases, the optimization phase of Chauve and Tannier [27], can do much better than the greedy approach of Ma et al. [96]. While both the greedy approach of Ma et al. [96] and the optimization approach of Chauve and Tannier [27] tend to work well in practice (these are the state of the art in bioinformatics methods for reconstructing AGOs), there is much more work to be done in the area of handling a matrix that does not have the C1P. The first step in this effort is to study *why* matrix M does not have the C1P. Indeed, previous works [27, 96] point this out, which we go into more detail in the next paragraph.

Indeed, the second step of this two step approach of reconstructing AGOs based on the C1P of binary matrices, is to transform binary matrix M into one that has the C1P. Ideally, if each synteny was a *true positive* ancestral synteny, then M would be C1P, however matrices from real data are rarely C1P. Rather some of the syntenies are *false positives*, i.e., not contiguous in the true ancestral genome. The reason and nature of these false positives depends highly on the data acquisition method. Depending on the method used, the reasons for this can be errors in constructing the set of markers $\mathscr{L}$, such as errors in the assembly from the whole genome alignments, such as paralogs being mistaken for orthologs in the construction of orthology blocks [96]. Other reasons come from the construction of incomplete syntenies due to the convergent loss of markers, and two syntenies joining together (creating a “chimeric” synteny) due to the convergent fusion of chromosomal segments in several lineages. For example, this second case of chimeric syntenies might happen especially in genomes of yeasts where we generally see many translocations [81, 128]. Indeed, it is unavoidable that we must deal with matrices that do not have the C1P. This is what motivates the work in this thesis. Why these matrices do not have the C1P depends on the nature of the errors in the data acquisition phase. In the next section, we illustrate the several open problems on such matrices, raised by these different types of errors, some of these mentioned in Chauve and Tannier [27], and then propose several relaxations of the C1P to address these problems, which is the contribution of this thesis. In some cases, solving these generalizations is NP-complete, and in other cases, there are algorithms for finding a solution.

1.3 Computational Solutions for non-C1P Matrices

1.3.1 Transforming the Matrix to a C1P Matrix

The first and most direct approach, taken in previous works [27, 96] is to transform the binary matrix *M* into one that has the C1P. Indeed, because of the assumptions made on the nature of the errors expected in their datasets (that the markers, i.e., columns, were inferred correctly), in Chauve and Tannier [27] they consider all computed syntenies, and extract a maximum subset of rows, such that submatrix M' of *M* defined by this set of rows is C1P. However, one could also remove columns from *M* if one was less confident on the correctness of the markers for example, or flip some entries in *M* from `0` to `1`, or from `1` to `0` to account for approximate syntenies. It follows, however that all corresponding optimization problems are NP-complete [36, 38, 66], even for sparse matrices [143]. For the case of extracting a maximum subset from *M* of rows or columns that is C1P, it has been shown in Dom [36] that this is also APX-hard and W[1]-hard. Aside from the work of Chauve and Tannier [27] and the reconstruction of AGOs in general, this problem of transforming a matrix *M* into one that has the C1P, while minimizing the modifications to *M* can be found in other applications [8, 143], as well as physical mapping [7, 55, 94, 149]. The latter comes as no surprise, since, from a computational point of view, physical mapping is also determining the C1P of a binary matrix in the presence of errors (in assembly, computing markers, etc.). We now introduce the contribution of this thesis: in the next three subsections, we outline three variants of the C1P motivated by this problem of reconstructing AGOs that we have proposed and/or studied here.

1.3.2 Relaxing the C1P

Another approach for handling a binary matrix *M* that does not have the C1P is, instead of transforming *M*, to relax the notion of the C1P, and then decide whether *M* has this relaxed property. A natural relaxation of the C1P is to allow gaps in each row of this “relaxed” C1 order of *M*. Indeed Chauve and Tannier [27] they claim that in their reconstructions, certain syntenic features are not captured with the strict nature of the C1P. Rather, if some number of gaps were allowed [16, 122], a significantly larger number of syntenies would be detected. However, allowing gaps could radically change the combinatorial nature of this problem, which means we cannot rely anymore on PQ-trees to encode all solutions, a powerful tool in using an approach based on the C1P for reconstructing AGOs.

Indeed a relaxed form of the C1P with gaps was considered in 1995, motivated by problems in the area of physical mapping [55]. Here Goldberg et al. [55] introduce the notion of the k-Consecutive-Ones Property (k-C1P). A binary matrix M has the k-C1P when its set of columns can be permuted such that each row contains at most k blocks. This is a fairly general form of relaxing the C1P, as it does not put any restriction on the size of the gaps between blocks. Goldberg et al. proposed this relaxation of the C1P to handle the case in physical mapping of chimeric clones: when sets of markers from two distant clones appear as the same clone, an artifact of hybridization [149]. Interestingly, from a computational standpoint, this is identical to the case of Chauve and Tannier [27] when two syntenies join together (creating a "chimeric" synteny) due to the convergent fusion of chromosomal segments in several lineages. The k-C1P models this case well, as there is no restriction on the distance between the two syntenies that join together to form the chimeric clone. However, this relaxation is indeed radically different in combinatorial nature, as Goldberg et al. [55] show that deciding if a binary matrix M has the k-C1P is NP-complete.

Chauve and Tannier [27] state, however, that a decision problem of "consecutive-ones with allowed gaps" is still open, i.e., each row of the matrix must have consecutive-ones, except that between each pair of ones, a fixed number of zeros is allowed. So, in this setting, it makes sense to consider a limit on the maximum size of any allowed gap. This idea has been motivated in other works as well. Indeed, Pasek et al. [122] consider an arbitrary number of fixed-sized gaps and are able to capture interesting conserved syntenic features. Further, Ouangraoua et al. [117], in work on double-conserved syntenies, show that when trying to transform their obtained matrix M into a C1P matrix, they must discard a large number of syntenies, and conclude that the C1P is not the proper model here, and that gaps are needed.

The first variant of this thesis is relaxation of the C1P with a limit on the maximum size of any gap. Here we define the Gapped C1P, or the (k, δ)-Consecutive-

Ones Property.

Property 1 ((k,δ)-Consecutive-Ones Property ((k,δ)-C1P)). *A binary matrix M has the (k,δ)-C1P for the two integers k and δ if the columns of M can be ordered such that each row contains at most k blocks, and no two neighboring blocks of* `1`*'s are separated by a gap of size more than δ.*

Notice that the classical C1P is equivalent to the $(1,0)$-C1P. If any of the two parameters is unbounded, we replace k or δ with ∞. For instance, the k-C1P is equivalent to the (k,∞)-C1P. Note also, then, that in the work of Pasek et al. [122] they consider precisely the (∞,δ)-C1P. Here, we call a permutation π of the columns of M that witnesses the (k,δ)-C1P a (k,δ)-consecutive-ones ((k,δ)-C1) *order* of M; that the matrix resulting from this permutation is (k,δ)-*consecutive*, or that it is (k,δ)-*consecutive with respect to* π; and that M is (k,δ)-C1P, or has the (k,δ)-C1P. Note that, for small k and δ, this is a stricter model than the ones considered before, such as the k-C1P [55] or that of Pasek et al. [122]. A model that is even more strict would be to consider the number of `0`'s in gaps in the entire matrix (in addition to the constraints k and δ) as a third parameter. This remains an interesting open question. Although the (k,δ)-C1P is stricter than previous models, we show in this thesis, however, that deciding this property is computationally hard for the most part.

We give our first set of results, the complexity of deciding the (k,δ)-C1P in Chapter 2 of this thesis. In Section 2.3, we show that for every $k \geq 2, \delta \geq 1, (k,\delta) \neq (2,1)$, deciding the (k,δ)-C1P is NP-complete, leaving open only case of the complexity of the $(2,1)$-C1P. We show that this remains NP-complete even if one of the two parameters is unbounded:

(i) for every $k \geq 2$, deciding the (k,∞)-C1P is just the problem of deciding if matrix M has the k-C1P, and is thus NP-complete by Goldberg et al. [55], and

(ii) for every $\delta \geq 1$, deciding the (∞,δ)-C1P is NP-complete (Section 2.5).

While the complexity of the (2,1)-C1P remains open, we do provide an algorithmic result for a relevant case of the (2,1)-C1P in Section 2.4. We now mention several other versions of the C1P with gaps considered in other works.

Another slightly different version of the C1P with gaps was considered in Haddadi [65], where they show that finding an order of the columns that minimizes the number of gaps in the entire matrix M is NP-complete, even if each row of M has degree at most two. While the works [75, 94] do not deal with the C1P with gaps, they do propose algorithms for recognizing matrices that are "close" to having the C1P in some sense. Aside from this, Dom [36, 37] presents an approximation algorithm as well as a fixed parameter algorithm for instances of the Set Cover Problem that are "close" to having the C1P, which basically means that either the input matrices have been generated by starting with a matrix that has the C1P and replacing randomly a certain percentage of the `1`'s by `0`'s [104], that the average number of blocks of `1`'s per row is much smaller than the number of columns of the matrix [131], or that the maximum number of blocks of `1`'s per row is small [105]. In light of this, approximation schemes remain to be considered for the (k,δ)-C1P, as well as any natural parameter that could lead to a Fixed Parameter Tractable (FPT) result. In the next subsection, we consider the (k,δ)-C1P for matrices of bounded degree which is the second variant of this thesis.

1.3.3 Matrices of Bounded Degree

The NP-completeness results on deciding the (k,δ)-C1P of Chapter 2 involve constructions with many rows of large degree. After examining some data from the experiments of Chauve and Tannier [27], however, we found that this is not always realistic. We considered here the ancestral syntenies dataset for the boreoeutherian ancestor of Chauve and Tannier [27] at a resolution of 200kb, with 1651 markers (i.e., columns) and 2515 syntenies (rows).[5] In this dataset, we observed that 90% of the syntenies have small degree (less than or equal to 16, which is less than 1% of the number of columns of this matrix). In addition to this, each of the remaining 10% of the syntenies (with degrees 17 to 99) contains between 16–144 of these syntenies of degree less than or equal to 16. Indeed this makes sense, as a long common interval that does not contain any other common interval would not be realistic. Hence, if the syntenies with large degree (10%) are discarded, the majority of the information is preserved. Indeed, this has already been shown in

[5]This dataset can be found at
http://www.cecm.sfu.ca/~cchauve/SUPP/ANCESTOR08/BOREO_200_u/index.html.

Chauve and Tannier [27]: when considering only adjacencies (matrices of degree 2), they obtain only slightly more CARs than in the general case of syntenies. This illustrates again that most of the signal is captured in small common intervals. In light of these two analyses, it makes sense to consider versions of the (k,δ)-C1P where the degree is bounded, especially if this could result in algorithms for these versions. Note that this would apply to chimeric syntenies: that we would expect that the individual syntenies that compose them will be detected as well, and then we just need to remove the row corresponding to a chimeric synteny.

To take into account the above observations, we consider here the case of the (k,δ)-C1P for matrices of bounded degree. This forms the second result of this thesis, given in Chapter 3. Formally, we define the (d,k,δ)-Consecutive-Ones Property.

Property 2 ((d,k,δ)-Consecutive-Ones Property ((d,k,δ)-C1P))**.** *A binary matrix M has the (d,k,δ)-C1P when the bound on the maximum degree of any row of M is d, and M has the (k,δ)-C1P.*

We call a permutation π of the columns of M that witnesses this property a (d,k,δ)-consecutive-ones ((d,k,δ)-C1) *order*; that the matrix M' resulting from this permutation is (d,k,δ)-*consecutive*, or that it is (d,k,δ)-*consecutive with respect to* π; and that M is (d,k,δ)-C1P, or has the (d,k,δ)-C1P. In Chapter 3, Section 3.1, we first show that if all three parameters are fixed, deciding the (d,k,δ)-C1P is related to the deciding the bandwidth of a graph, and can be decided in polynomial time by slightly modifying an algorithm of Saxe [135] for recognizing graphs with a fixed constant bandwidth. While this algorithm is only practical for small values of the parameters, this is usually the case in practice (cf. Chauve and Tannier [27] and discussion in previous paragraphs). Currently, an implementation of this algorithm on biological data is in a preliminary stage. We point out that for the case where $d = 2$, we can also take advantage of the faster linear-time algorithm of Caprara et al. [25] for the bandwidth 2 case. An interesting open question here is whether or not the techniques used in Caprara et al. [25] can be extended to matrices of degree (and graphs of bandwidth) larger than two.

After obtaining this algorithmic result for the case of deciding the (d,k,δ)-C1P when all three parameters are fixed, we began to study the complexity of deciding this property when one or more of these parameters is unbounded. The case with d unbounded is just the (k,δ)-C1P, and hence the complexity of deciding everything except for the $(\infty,2,1)$-C1P, or just the $(2,1)$-C1P is known. Since fixing d also fixes k ($k \leq d$), the only case that remains for us to consider is the case when δ is unbounded, or the (d,k,∞)-C1P. The motivation from a practical point of view to consider this case is that it concerns chimeric syntenies (the gap size is unbounded) where we assume that we do not lose too much information by considering only syntenies with low degree as argued above. Here, in Chapter 3, Section 3.2.4, we show that in every non-trivial case, deciding this property is NP-complete, i.e., for every $d > k \geq 2$, deciding the (d,k,∞)-C1P is NP-complete. Note that if $d = 2$, the this becomes the C1P, and if $d \leq k$, then any order of the columns of M is a valid solution, since no row can have more than d blocks of `1`'s. This case is also of importance to physical mapping, since chimerism is a phenomenon that happens here also. In particular, since the setting when clones are short and there is limited coverage of the sequence by the clones is likely to be more realistic (similar to how it is in the reconstruction of AGOs), Goldberg et al. [55] pose the question of deciding the 2-C1P when the number of ones per row and per column is bounded. Interestingly, the construction we use in Section 3.2.4 of Chapter 3 happens also to use a bounded number of ones per column, and hence we answer the above question posed by Goldberg et al. [55]. In the next subsection, we present the third variant of the C1P of binary matrices that we study in this thesis.

1.3.4 Matrices with Columns of Multiplicity

Here, we present the third variant of the C1P of binary matrices that we study in this thesis, namely to allow columns to appear multiple times in a C1 order. While this is technically another relaxation of the C1P, it is very different than the ones considered previously. It also models a very different phenomenon in the reconstruction of AGOs, namely duplicated (or indistinguishable) markers. Indeed, a preliminary approach for handling this was mentioned in Landau et al. [91]. Alternative ways of handling duplicated markers was also a line of future research posed in Chauve and Tannier [27]. The input to C1P based approach mentioned above for reconstructing AGOs is a set of pairwise distinct markers $\mathscr{L} = \{1,\ldots,n\}$. This assumption is needed for the use of the C1P and, in particular, PQ-trees for the reconstruction of AGOs (the columns of the binary matrix M that is the input to deciding the C1P are pairwise distinct). In order to cope with datasets containing duplicate markers (among other things like missing or overlapping markers which are beyond the scope of this discussion), in Chauve and Tannier [27] they use approximate intervals of markers in the detection phase. That is, a set of markers need only be approximately similar (e.g., 80% similar) between two species from S, where the path in T between these two species goes through the ancestor, for it to be considered a synteny (a row in M). This approach in some sense allows the existence of duplications by relaxing the detection of syntenies. An alternate approach suggested in Chauve and Tannier [27] would be to infer some pre-duplication AGO, which has been considered in some rearrangment-based works such as [3, 41, 134].[6] Chauve and Tannier [27] also mention that there exist algorithms for computing syntenies between pairs of genomes with duplicate markers [16], or with duplicate segments followed by losses in both copies [146]. However, because these algorithms account for duplicates, the input is not assumed to be a set of pairwise distinct markers anymore, and hence one cannot use the C1P to model AGOs here.

In 2009, a year after the important result of Chauve and Tannier [27], Stoye and Wittler [139] present a parsimony approach for reconstructing AGOs[7] that uses PQ-trees [139]. Here, they propose a framework based on Bergeron et al. [15], which is what Chauve and Tannier [27] is based on, and provide an efficient method for finding a most parsimonious AGO, which they show works well in practice. In this work of Stoye and Wittler [139], they propose extending their models to allow markers to appear multiple times (to account for duplications). A year after this, in Wittler and Stoye [151], the authors then formally define a model that incorporates markers with multiplicity. This model is equivalent to deciding the following property of binary matrices.

Property 3 (Consecutive-Ones Property with Multiplicity (mC1P))**.** *Given a binary matrix M on columns $S = \{1,\dots,n\}$ and a function $\mathbf{m} : S \to \mathbb{N}$, is there a*

[6]Refer to Ma et al. [97] for a solution to handling duplicate markers in the case of physical mapping.

[7]More accurately, their work concerns models of *gene clusters*, of which a set of syntenies used to reconstruct an AGO is one such model. We only discuss their work in the scope of reconstructing AGOs to remain within the subject of this thesis.

sequence σ *over alphabet S that*

(i) σ *contains each column $s \in S$ at most $\mathbf{m}(s)$ times, and*

(ii) for each row r of M, the set of columns that have entry `1` *in r form at least one* subsequence *of σ.*

Note that deciding this property becomes trivial if we allow any column to have arbitrary multiplicity, i.e., we could take σ to be the concatenation of all rows of M. Of course, such a long AGO would be dubious, and hence a threshold on the multiplicity of each marker is reasonable. This is why Wittler and Stoye [151] introduce this *multiplicity constraint* (i). This property generalizes the C1P: indeed the C1P is the case when $\mathbf{m}(s) = 1$ for all $s \in S$, i.e., that there simply is a *permutation* π over the alphabet S such that (ii) holds. Here, we call this the mC1P. Of course, now that this problem has moved outside the domain of permutations into sequences, the classical C1P and the associated PQ-tree do not apply anymore. A natural question to ask then is the complexity of deciding the mC1P.

In Wittler and Stoye [151], they show that deciding the mC1P can be done in polynomial time if each row of M has degree at most 2 (which is the model of adjacencies) by showing that this problem is equivalent to deciding if a graph is Eulerian. The authors of Wittler and Stoye [151] also show that if each row has degree at most 5, then the mC1P, as well as two restricted variants motivated by biological settings, is NP-complete. We mention that one of these restricted variants, the case of framed common intervals on permutation, was the first model used to formally state the problem of reconstructing AGOs using PQ-trees [15].

In this thesis, we improve these NP-completeness results to each row having degree at most 3 (resp., at most 6 in the case of the framed common intervals variant), while $\mathbf{m}(s) \leq 2$ for each $s \in S$, where S is the set of columns of M. We give these results in Section 4.1 and 4.2 of Chapter 4. The techniques used here to improve these NP-completeness results are based on those introduced in Chapter 3 for showing NP-completeness of deciding the (d, k, ∞)-Consecutive-Ones Property ((d, k, ∞)-C1P).

Finally, in Section 4.3 of Chapter 4, we then present a tractability result which is motivated in the following. The C1P based approach for reconstructing AGOs introduced here (for example, by Chauve and Tannier [27]) involves computing a set of ancestral syntenies represented by binary matrix M, and then building a PQ-tree for M (by possibly transforming M to a C1P matrix). Here, each subtree rooted at a child of the root of this PQ-tree represents a CAR. A CAR is an ancestral chromosomal segment, but it is not guaranteed to be a complete ancestral chromosome. In fact, it is common that the number of CARs obtained is larger than the expected number of ancestral chromosomes. This raises the following natural question: which CARs are believed to form complete ancestral chromosomes, or more generally, to contain an extremity of an ancestral chromosome (an ancestral telomere)? Indeed, a CAR with two ancestral telomeres is in fact a complete ancestral chromosome. Moreover, when CARs are grouped into syntenic sets, that is, sets of CARs that are believed to belong to the same ancestral chromosome, each such syntenic set of CARs can contain only two ancestral telomeres. We address this question as follows. A column c' with multiplicity (bounded, for example, by twice the maximum expected number of ancestral chromosomes, or more generally with infinite multiplicity) can then be used to represent telomeres, that is, virtual extremities of ancestral chromosomes. Then any ancestral synteny that contains putatively a marker that is an extremity of an ancestral chromosome (for example because the ancestral synteny is telomeric in two existing descendants of the considered ancestor) can be represented by two rows in M: a row representing the ancestral synteny, plus a copy of this row with an additional entry `1` in column c'. This structure ensures that if M has the mC1P, then the occurrences of c' are located at the extremities of the CARs. Otherwise (M does not have the mC1P), some rows can be discarded to result in a matrix M' that has the mC1P, with the same property. This assumption on the structure of M is fundamental to leave open the possibility for any ancestral synteny to be at the extremity of a CAR or to be embedded inside a CAR. It follows that the tractable family of matrices considered here meets precisely this assumption.

Formally, in Section 4.3 of Chapter 4, we present a tractability result for a family of matrices where every row of M has (i) at most one entry `1` in columns with multiplicity greater than one, or (ii) exactly two entries `1` in columns with multiplicity greater than one and no other entries. Our proofs rely on the two classical concepts of PQ-trees and Eulerian graphs. The final section of this thesis outlines our study of the GCCC Problem, where we present our fourth and final variant of the C1P that we use to develop an algorithm for a special case of this problem.

1.4 The Generalized Cladistic Character Compatibility Problem

In Chapter 5 we present our fourth and final variant of the C1P in order to develop an algorithm for a case of a phylogeny problem that we consider here. We now briefly motivate our study of this type of phylogeny problem.

Here we study the problem of constructing a phylogenetic tree for a set of species [45]. A *qualitative character* assigns to each species a *state* from a set of states, e.g., "is a vertebrate", or "number of legs". When the evolution of the states of the character is known, e.g., evolution from invertebrate to vertebrate is only forward, the character is called *cladistic*. This evolution of the states is usually represented by a rooted tree, called a *character tree*, on the set of states. The *Qualitative Character Compatibility* Problem, or *Perfect Phylogeny* Problem, is NP-complete [20, 138], while it is polynomial-time solvable when any of the associated parameters is fixed [2, 82, 83, 103]. When characters are cladistic, the problem, called the *Cladistic Character Compatibility* Problem, is the problem of finding a perfect phylogeny tree on the set of species such that it can be contracted to a subtree of each character tree. This problem is polynomial-time solvable [42, 62, 148].

Experimental research in molecular biology [47, 79, 86, 144] shows that traits can disappear and then reappear during the evolution of a species, suggesting that genes contain information about traits that are not always expressed. In Benham et al. [11, 12], the authors argue that a new model for characters is needed in order for the resultant phylogenetic trees to capture this phenomenon. The authors thus devise the *generalized character*, which assigns to each species a *subset* of a set of states, where we only know that the expressed trait (state) is in this subset. The GCCC Problem is then the Cladistic Character Compatibility Problem on a set of species with generalized characters where we first have to pick one state from the subset for each character. Interestingly, generalized characters capture also the case of qualitative characters with missing data (the "Incomplete Perfect Phylogeny" Problem). Here, missing data can be replaced by a "wildcard" generalized state containing all possible states of the character. This problem was shown to be NP-complete even if the number of states is constant in [63].

In Chapter 5 we study the complexity of several cases of the GCCC Problem that are motivated by the previous works of Benham et al. [11, 12]. In Subsection 5.3.2, we introduce a variant of the C1P which gives us an algorithm for a case of this problem.

Chapter 2

The Gapped Consecutive-Ones Property

In this chapter we show that for every bounded and unbounded $k \geq 2, \delta \geq 1, (k,\delta) \neq (2,1)$, deciding the (k,δ)-C1P is NP-complete. Section 2.1 outlines the notation used in this chapter. Section 2.2 provides a theorem that is central to the results of Section 2.3: that for every bounded $k \geq 2, \delta \geq 1, (k,\delta) \neq (2,1)$, deciding the (k,δ)-C1P is NP-complete. In Section 2.4, we then give an algorithm for a case of the (2,1)-C1P that is motivated by the type of construction used to obtain the results of Section 2.3. In the final Section 2.5 of this chapter we show that for every $\delta \geq 1$, deciding the (∞,δ)-C1P is NP-complete.

2.1 Notation and Conventions

First we introduce all notation and conventions used throughout this chapter. Given integers a,b, where $a \leq b$, $\langle a,b \rangle$ denotes the set $\{a, a+1, \ldots, b\}$. Let M be a binary $m \times n$ matrix (on `0`'s and `1`'s) with columns labelled by $\langle 1,n \rangle$. In the constructions used to show NP-completeness of deciding the (k,δ)-C1P, we will divide columns of M into ordered sequences of blocks $B_1,\ldots,B_p$ by designing rows enforcing the columns of each block to appear together and the blocks to appear in the order $B_1,\ldots,B_p$ (resp., in the reversed order), i.e., for any $i < j$, column $c \in B_i$ and $d \in B_j$, c appears before (resp., after) d in any (k,δ)-C1 order of M. The columns

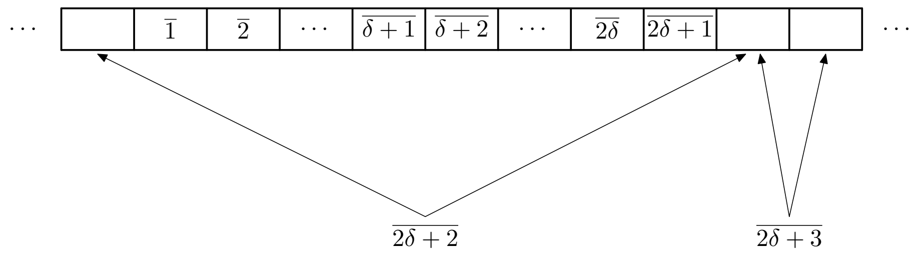


Figure 2.1: Possible positions of columns $\overline{2\delta+2}$ and $\overline{2\delta+3}$.

of a block B_i will be denoted $B_i^1, \ldots, B_i^{|B_i|}$ and $B_i^{\langle a,b\rangle} = \{B_i^a, B_i^{a+1}, \ldots, B_i^b\}$, where $a \leq b$.

To specify a row in the a binary matrix M, we use the convention of only listing in the square brackets, the columns that contain `1` in this row. For example, $[1,8,5]$ represents a row with `1`'s in columns 1, 5, and 8, and `0`'s everywhere else. We will also use blocks of M to specify columns in the block, for example, if $B_1 = \{1,2,3,4,5\}$, then $[B_1, 7]$ would mean $[1,2,3,4,5,7]$, $[B_1 \setminus \{B_1^2\}, 6, 7]$ would mean $[1,3,4,5,6,7]$, and $[B_1^{\langle 2,4\rangle}, 6]$ would mean $[2,3,4,6]$.

2.2 Fixing the Order of Selected Columns in a Matrix

For every $k \geq 2, \delta \geq 1$, we have the following important property of matrices that have the (k,δ)-C1P. Note that the following construction does not depend on k as it uses only two ones per row.

Theorem 4. *For every $k \geq 2$ or $k = \infty$, $\delta \geq 1$ and $s \geq 2\delta + 3$, given binary matrix M on $n \geq s$ columns, $s + \delta + 1$ rows can be added to M to force s selected columns to appear together and in fixed order (or the reverse order) in any (k,δ)-C1 order of M.*

Proof. Let $k \geq 2$ (or $k = \infty$), $\delta \geq 1$, $s \geq 2\delta + 3$ and $n \geq s$. Without loss of generality, let $S = \{\overline{1}, \ldots, \overline{s}\}$ be the subset of s columns that we want to force to appear together and in this order (or the reverse order) in any (k,δ)-C1 order of M. We will show by induction on s that there are $s + \delta + 1$ rows of the type $[\overline{c}, \overline{d}]$, where $1 \leq c < d \leq s$ and $|c - d| \leq \delta + 1$, which force this order.

For the base case, let us assume that $s = 2\delta + 3$. We will show the base case by induction on δ. If $\delta = 1$, then $s = 2 \cdot 1 + 3 = 5$, and we add to M the following 7 rows: $[\overline{1}, \overline{2}]$, $[\overline{2}, \overline{3}]$, $[\overline{3}, \overline{4}]$, $[\overline{4}, \overline{5}]$, $[\overline{1}, \overline{3}]$, $[\overline{2}, \overline{4}]$, and $[\overline{3}, \overline{5}]$. It is easy to check that the claim holds and that the number of rows used is exactly $s + \delta + 1$. Now assume that the claim holds for $\delta = \delta_0$ and $s = s_0 = 2\delta_0 + 3$, where $\delta_0 \geq 1$. We will show that it holds also for $\delta = \delta_0 + 1$ and $s = 2\delta + 3 = 2\delta_0 + 5$. Using the induction hypothesis, there are $s_0 + \delta_0 + 1 = s - 2 + \delta - 1 + 1 = s + \delta - 2$ rows, which will force the correct order for columns $\overline{1}, \ldots, \overline{2\delta + 1}$. Note that all of these rows $[\overline{c}, \overline{d}]$ satisfy the condition $|c - d| \leq \delta + 1$, and hence, they can be added to M for parameters $\delta = \delta_0 + 1$ and $s = 2\delta_0 + 5$. In addition, we add to M three new rows: $[\overline{\delta + 1}, \overline{2\delta + 2}]$, $[\overline{\delta + 2}, \overline{2\delta + 3}]$ and $[\overline{2\delta + 2}, \overline{2\delta + 3}]$. The total number of rows added to M is now $s + \delta + 1$. Figure 2.1 shows the possible positions of columns $\overline{2\delta + 2}$ and $\overline{2\delta + 3}$ forced by rows $[\overline{\delta + 1}, \overline{2\delta + 2}]$ and $[\overline{\delta + 2}, \overline{2\delta + 3}]$ if we assume that rows $\overline{1}, \ldots, \overline{2\delta + 1}$ appear in the correct order. It is easy to see that the row $[\overline{2\delta + 2}, \overline{2\delta + 3}]$ is (k, δ)-consecutive only if columns $\overline{2\delta + 2}$ and $\overline{2\delta + 3}$ appear in the correct positions as well. This completes the induction on δ and we have that the claim holds for any $\delta \geq 1$ and $s = 2\delta + 3$, i.e., the base case for the induction on s.

Now, assuming that the claim holds for $s - 1$, where $s - 1 \geq 2\delta + 3$, we show that it holds also for s columns. By the induction hypothesis, there are $s + \delta$ rows which will force columns $\overline{1}, \ldots, \overline{s - 1}$ to appear in the correct order. We add one new row: $[\overline{s - \delta - 1}, \overline{s}]$. Since $s - \delta - 1 \geq \delta + 3$, there is only one position where column $\overline{s}$ can appear: next to $\overline{s - 1}$, i.e., all columns in S appear in correct order. The number of rows used is exactly $s + \delta + 1$. This completes the induction on s, and the claim follows. □

2.3 The Complexity of Deciding the (k, δ)-C1P

In this section we will show that for every $k \geq 2, \delta \geq 1, (k, \delta) \neq (2, 1)$, deciding the (k, δ)-C1P is NP-complete.

2.3.1 The Complexity of Deciding the (k, δ)-C1P for every $k, \delta \geq 2$

For every $k, \delta \geq 2$, we use Theorem 4 in a reduction from 3SAT to the problem of deciding the (k, δ)-C1P to show that this problem is NP-complete.

Theorem 5. *For every $k, \delta \geq 2$, deciding the (k, δ)-C1P is NP-complete.*

Proof. Consider $k, \delta \geq 2$. Let ϕ be a 3CNF formula over the n variables $\{v_1, \ldots, v_n\}$, with m clauses $\{c_1, \ldots, c_m\}$. We construct a matrix M_ϕ with $2n + d + 6m$ columns and $n + 7m + d + \delta + 1$ rows, where $d = \max\{2k-1, 2\delta+3\}$, such that M_ϕ has the (k, δ)-C1P if and only if ϕ is satisfiable.

Goldberg et al. [55] show that for every $k \geq 2$, given a 3CNF formula ϕ, they can construct a matrix M_ϕ that has the k-C1P if and only if ϕ is satisfiable. Our construction is based on theirs. In our construction, we associate the first $2n$ columns $\langle 1, 2n \rangle$ of M_ϕ with the variables $\{v_1, \ldots, v_n\}$. In particular, we associate variable v_i with the pair of columns $b_i = \{2i-1, 2i\}$, for $i \in \langle 1, n \rangle$. Variable v_i equal to *true* represents the statement about the order of the columns: "$2i-1$ is before $2i$" (v_i equal to *false* represents statement: "$2i-1$ is after $2i$"). Since a truth assignment to the formula ϕ represents a statement about a permutation of the columns of M_ϕ, we want to relate M_ϕ to the clauses $\{c_1, \ldots, c_m\}$ of ϕ in such a way that only the permutations of M_ϕ that are (k, δ)-consecutive correspond to truth assignments that satisfy ϕ and vice versa. This construction involves associating the last $6m$ columns $\langle 2n+d+1, 2n+d+6m \rangle$ with the clauses $\{c_1, \ldots, c_m\}$. In particular, we associate clause c_j with the block of five columns $B_j = \langle 2n+d+6j-4, 2n+d+6j \rangle$, while each block B_j is preceded by a column $a_j = \{2n+d+6j-5\}$. Finally, the set $\langle 2n+1, 2n+d \rangle$ of columns in the middle will be used to ensure that the construction works for parameters k and δ. The details are as follows.

The base of our construction is a subset of the columns of M_ϕ that we force to be together and in fixed order in any (k, δ)-C1 order of M_ϕ, and then we will build off of this base a construction similar to that of Goldberg et al. [55]. In particular, we impose this fixed order on this subset $\langle 2n+1, 2n+d \rangle$ of the columns in the middle of M_ϕ by adding $d + \delta + 1$ rows to M_ϕ according to Theorem 4. While these d columns must be together and in fixed order (or the reverse) in any (k, δ)-C1 order, we assume the former without loss of generality. We now build the remaining construction off of this block of d columns.

To force the blocks $b_1, \ldots, b_n$ to appear together and in this order, and before the set $\langle 2n+1, 2n+d \rangle$ of d columns in M_ϕ, we add the n rows $[b_i, b_{i+1}, \ldots, b_n, 2n+1, 2n+3, \ldots, 2n+2k-3, 2n+2k-1]$ to M_ϕ, for $i \in \langle 1, n \rangle$. Observe that, if block

b_n is not immediately to the left of the d columns, then there are more than $k-1$ gaps in the row $[b_n, 2n+1, 2n+3, \dots, 2n+2k-3, 2n+2k-1]$, while, for each $i \in \langle 1, n-1\rangle$, if block b_i is not immediately to the left of b_{i+1}, then there are more than $k-1$ gaps in the row $[b_i, b_{i+1}, \dots, b_n, 2n+1, 2n+3, \dots, 2n+2k-3, 2n+2k-1]$.

Next, to force the blocks $a_1, B_1, \dots, a_m, B_m$ to appear together and in this order, and *after* the set $\langle 2n+1, 2n+d\rangle$ of d columns in M_ϕ, we add the $2m$ rows $[2n+d-(2k-2), 2n+d-(2k-4), \dots, 2n+d-4, 2n+d-2, 2n+d, a_1, B_1, \dots, a_{j-1}, B_{j-1}, a_j]$ and $[2n+d-(2k-2), 2n+d-(2k-4), \dots, 2n+d-4, 2n+d-2, 2n+d, a_1, B_1, \dots, a_j, B_j]$ to M_ϕ, for $j \in \langle 1, m\rangle$.

Now the blocks of columns in any (k,δ)-C1 order of the matrix M_ϕ are ordered as follows: the blocks $b_1, \dots, b_n$ associated with the variables of ϕ, followed by the d columns $2n+1, \dots, 2n+d$, followed by the blocks $a_1, B_1, \dots, a_m, B_m$, where the blocks $B_1, \dots, B_m$ are associated with the clauses of ϕ. Since the restrictions placed on variable blocks $\{b_1, \dots, b_n\}$ and the clause blocks $\{B_1, \dots, B_m\}$ are the same as in Goldberg et al. [55], we simply have to add rows, similar to those in Goldberg et al. [55], to M_ϕ to associate each clause to its three variables to properly simulate 3SAT. The difference from our construction to that of Goldberg et al. [55], is what values the row takes within this segment $\langle 2n+1, 2n+d\rangle$ of d columns and the m columns $a_1, \dots, a_m$. We now present the details.

Suppose that clause c_j contains the literal v_α. We add the following (corresponding) row to M_ϕ: $[b_\alpha^2, b_{\alpha+1}, \dots, b_n, 2n+1, 2n+3, \dots, 2n+2k-7, 2n+2k-5, \langle 2n+2k-3, 2n+d\rangle, a_1, B_1, \dots, a_j, B_j^1]$. If v_α is *false*, this forces B_j^1 to be the first column of block B_j in any (k,δ)-C1 order of M_ϕ. Any other order of the columns of B_j would introduce a k-th gap in this row. If v_α appears negated in c_j, then we add the row $[b_\alpha^1, b_{\alpha+1}, \dots, b_n, 2n+1, 2n+3, \dots, 2n+2k-7, 2n+2k-5, \langle 2n+2k-3, 2n+d\rangle, a_1, B_1, \dots, a_j, B_j^1]$ instead. Suppose another literal in c_j is v_β. We add the row $[b_\beta^2, b_{\beta+1}, \dots, b_n, 2n+1, 2n+3, \dots, 2n+2k-7, 2n+2k-5, \langle 2n+2k-3, 2n+d\rangle, a_1, B_1, \dots, a_j, B_j^{\langle 1,4\rangle}]$. If v_β is *false*, this forces B_j^5 to be the last column of block B_j. Suppose the third literal of c_j is v_γ. We add the rows $[b_\gamma^2, b_{\gamma+1}, \dots, b_n, 2n+1, 2n+3, \dots, 2n+2k-7, 2n+2k-5, \langle 2n+2k-3, 2n+d\rangle, a_1, B_1, \dots, a_j, B_j^{\langle 1,2\rangle}]$ and $[b_\gamma^2, b_{\gamma+1}, \dots, b_n, 2n+1, 2n+3, \dots, 2n+2k-7, 2n+2k-5, \langle 2n+2k-3, 2n+d\rangle, a_1, B_1, \dots, a_j, B_j^{\langle 1,3\rangle}]$ to M_ϕ. If v_γ is *false*, this forces B_j^3 to be the middle col-

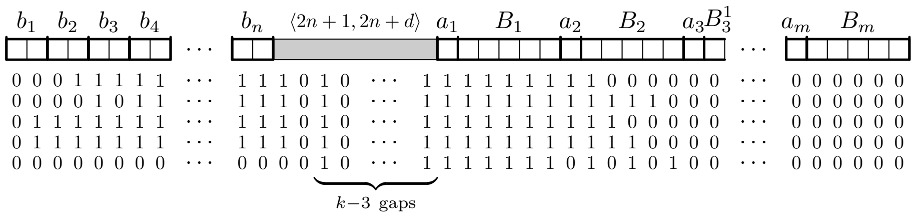


Figure 2.2: The structure of M_ϕ and the five rows encoding clause $c_2 = \{v_2 \vee \neg v_3 \vee v_1\}$.

umn of block B_j. Finally, we add the row $[2n+3, 2n+5, \ldots, 2n+2k-7, 2n+2k-5, \langle 2n+2k-3, 2n+d\rangle, a_1, B_1, \ldots, a_{j-1}, B_{j-1}, B_j^1, B_j^3, B_j^5]$ to M_ϕ. This last row is not (k,δ)-consecutive exactly when B_j^1, B_j^3, and B_j^5 are the first, middle and last columns of block B_j, as it contains k gaps then. This fifth row enforces the constraint that not all three literals of c_j can be *false*. Figure 2.2 illustrates the structure of matrix M_ϕ, along with these five rows that would be added to M_ϕ for clause $c_2 = \{v_2 \vee \neg v_3 \vee v_1\}$.

It remains to show that if any literal in c_j is *true*, then there is some order of the columns of block B_j such that these five rows are (k,δ)-consecutive. If v_α (resp., v_β) is *true*, we can order the columns $B_j^2, B_j^1, B_j^3, B_j^4, B_j^5$ (resp., $B_j^1, B_j^2, B_j^3, B_j^5, B_j^4$). If v_γ is *true*, the columns can be in any order that places B_j^1 (resp., B_j^5) in the first (resp., last) position, while placing B_j^2, B_j^3, B_j^4 in any of the four orders that avoids placing B_j^3 in the middle (as this fifth row would have k gaps in this case). Note that these orders work even when the corresponding variable is the only one that is *true*, and that in all of these orders, no row has a gap of size larger than two. Finally, we remark that if v_γ is the only variable that satisfies clause c_j, for example, then in all of the four (possible) orders of the columns where these five rows are (k,δ)-consecutive, there is a gap of size two in the fifth row. Hence this construction does not work for $\delta = 1$.

Since, for every $k, \delta \geq 2$, deciding the (k,δ)-C1P is clearly in NP, by the above reduction from 3SAT, it follows that for every $k, \delta \geq 2$, deciding the (k,δ)-C1P is NP-complete. $\square$

2.3.2 The Complexity of Deciding the $(k,1)$-C1P for every $k \geq 3$

We slightly modify the reduction from 3SAT in the proof of Theorem 5 to show that, for every $k \geq 3$, deciding the $(k,1)$-C1P is NP-complete.

Theorem 6. *For every $k \geq 3$, deciding the $(k,1)$-C1P is NP-complete.*

Proof. Consider $k \geq 3$. Let ϕ be a 3CNF formula over the n variables $\{v_1,\ldots,v_n\}$, with m clauses $\{c_1,\ldots,c_m\}$. We construct a matrix M_ϕ with $2n+d+4m$ columns and $n+4m+d+2$ rows, where $d = 2k-1$, such that M_ϕ has the $(k,1)$-C1P if and only if ϕ is satisfiable. We do this as follows.

We again use Theorem 4 to force the columns $\langle 2n+1, 2n+d\rangle$ to appear together and in fixed order in any $(k,1)$-C1 order of M_ϕ, and build a construction off of this block. We again associate columns $\langle 1,2n\rangle$ with the variables of ϕ, and associate each clause c_j with block B_j. However, B_j now has four columns rather than five, that is $B_j = \langle 2n+d+4j-3, 2n+d+4j\rangle$. Note also that we do not have the blocks a_j in this construction. We again add the appropriate rows to M_ϕ so that the columns of any $(k,1)$-C1 order of the matrix M_ϕ are ordered $b_1,\ldots,b_n$, followed by $2n+1,\ldots,2n+d$, followed by $B_1,\ldots,B_m$. The only major difference from Theorem 5 of this reduction is the manner in which we associate the clauses to their variables to property simulate 3SAT. The details are as follows.

We need to introduce only three more rows to associate the clauses to their variables to properly simulate 3SAT. Suppose that clause c_j contains literals v_α, v_β and v_γ. We add the row $[b_\alpha^2, b_{\alpha+1},\ldots,b_n, 2n+1, 2n+3,\ldots,2n+2k-9, 2n+2k-7, \langle 2n+2k-5, 2n+d\rangle, B_1,\ldots,B_{j-1}, B_j^{\langle 1,2\rangle}]$ to M_ϕ. If v_α is *false*, this forces B_j^1 and B_j^2 to be among the first three columns of block B_j in any $(k,1)$-C1 order of M_ϕ. Note that any other order of the columns of B_j would introduce either a gap of size 2, or a k-th gap in this row. Similarly, we add the rows $[b_\beta^2, b_{\beta+1},\ldots,b_n, 2n+1, 2n+3,\ldots,2n+2k-9, 2n+2k-7, \langle 2n+2k-5, 2n+d\rangle, B_1,\ldots,B_{j-1}, B_j^1, B_j^3]$ and $[b_\gamma^2, b_{\gamma+1},\ldots,b_n, 2n+1, 2n+3,\ldots,2n+2k-9, 2n+2k-7, \langle 2n+2k-5, 2n+d\rangle, B_1,\ldots,B_{j-1}, B_j^1, B_j^4]$ to M_ϕ. If v_β is *false*, this forces B_j^1 and B_j^3 to be among the first three columns of block B_j, and if v_γ is *false*, this forces B_j^1 and B_j^4 to be among the first three columns of block B_j. Finally, since $B_j^1, B_j^2, B_j^3, B_j^4$ cannot simultaneously be among the first three columns of block B_j, we have that not all three literals of c_j can be *false* in any $(k,1)$-C1 order of M_ϕ.

It remains to show that if any literal in c_j is *true*, then there is some order of the columns of block B_j such that these four rows are $(k,1)$-consecutive. If v_α (resp., v_β, and v_γ) is *true*, we can order the columns $B_j^3, B_j^1, B_j^4, B_j^2$ (resp., $B_j^2, B_j^1, B_j^4, B_j^3$, and $B_j^2, B_j^1, B_j^3, B_j^4$). Note that these orders work even when the corresponding variable is the only one that is *true*.

Since, for every $k \geq 3$, deciding the $(k,1)$-C1P is clearly in NP, by the above reduction from 3SAT, it follows that for every $k \geq 3$, deciding the $(k,1)$-C1P is NP-complete. $\square$

In summary, by Theorem 5 and Theorem 6, it follows that for every $k \geq 2, \delta \geq 1, (k,\delta) \neq (2,1)$, deciding the (k,δ)-C1P is NP-complete. The only open question that remains is the case of the complexity of deciding the (2,1)-C1P. In the next section we give a result for a special case of the (2,1)-C1P.

2.4 The (2,1)-C1P

All of the constructions in this chapter used to show NP-completeness of deciding the (k,δ)-C1P for $k \geq 2, \delta \geq 1, (k,\delta) \neq (2,1)$ divided the columns of a binary matrix M into an ordered sequence of blocks $B_1,\ldots,B_p$ by designing rows which force the columns of each block to appear together and the blocks to appear in the order $B_1,\ldots,B_p$ (or in the reversed order), i.e., for any $t < u$, column $d \in B_t$ and $e \in B_u$, d appears before e in any (k,δ)-C1 order of M. We will call any permutation of the columns of M that meets this condition a $\{B_1,\ldots,B_p\}$-*block-structured order*. Given any 3CNF formula ϕ, we then represented each variable and each clause with a block from $\{B_1,\ldots,B_p\}$, where the permutations of the columns within this block correspond to the configurations (a) *true* and *false*, if it is a variable block, or (b) which of its literals is set to *true* and *false*, if it is a clause block. The above restriction of columns into blocks then provided enough structure so that for each clause c, we could add some rows to M that introduce dependencies only between the permutations of columns of the block for c and the 3 blocks corresponding to each of c's literals, such that c is satisfied in ϕ if and only if these rows have the (k,δ)-C1P, and no other dependencies, i.e., that ϕ is satisfiable if and only if M has a (k,δ)-C1 $\{B_1,\ldots,B_p\}$-block-structured order.

Here we provide a polynomial-time $O(m^2 n(\ell+1)! + n\ell! 2^{3\ell})$ and space

$O(m^2 n\ell! + 2^\ell)$ algorithm, given binary $n \times m$ matrix M where its columns are divided into an ordered sequence of blocks $B_1, \ldots, B_p$ and each block contains at most some fixed constant number ℓ of columns ($|B_t| \leq \ell$ for all $t \in \langle 1, p \rangle$), which either

(a) decides if it has a $\{B_1, \ldots, B_p\}$-block-structured (2,1)-C1 order, or

(b) finds a proof that deciding the (2,1)-C1P is NP-complete.

Note that we can force any (2,1)-C1 order to be a $\{B_1, \ldots, B_p\}$-block-structured order by adding rows to the matrix similarly as it was done in Theorems 5 and 6.

One observation is that this algorithm is FPT in parameter ℓ. Another motivation for this result is that if the (2,1)-C1P is NP-complete even in this $\{B_1, \ldots, B_p\}$-block-structured case, then this algorithm provides an automated tool which could be used to prove this: with some instance of the problem, it would find the proof that deciding the (2,1)-C1P is NP-complete. We now give the algorithm in the next subsection.

2.4.1 The Algorithm

Let M be a binary matrix on m rows and n columns and $\mathscr{B} = \{B_1, \ldots, B_p\}$ be sets of columns of M where $|B_t| \leq \ell$ for all $t \in \langle 1, p \rangle$. The basic idea of the algorithm is as follows. First, it does some preprocessing on M to check if it has some necessary properties for it to have a $\mathscr{B}$-block-structured order that is also a (2,1)-C1 order. If this succeeds, it then checks another condition of matrix M. If this condition holds, it generates a set of 2-clauses of polynomial-size that is satisfiable if and only if M has such an order. If this condition does not hold, it is able to find in polynomial-time, proof that deciding the (2,1)-C1P is NP-complete.

Given M and $\mathscr{B} = \{B_1, \ldots, B_p\}$, for $t \in \langle 1, p \rangle$, let $\mathscr{U}_t$ denote the set of permutations of B_t that are (2,1)-C1P with respect to M, for some order of the columns outside of B_t. We will explain later exactly how to compute $\mathscr{U}_t$, but for now, we observe the following property:

Property 7. *The set of $\{B_1, \ldots, B_t\}$-block-structured (2,1)-C1 orders of M is a subset of $\mathscr{U} = \mathscr{U}_1 \times \cdots \times \mathscr{U}_p$.*

Let r_i, for $i \in \langle 1, m \rangle$ be a row of M. For a set B of columns of M, we use $\sigma(B, i)$ to denote the subset of columns of B that contain a `1` in row r_i. Let s^i (resp., e^i) $\in \langle 1, p \rangle$ be the index of the first (resp., last) block of M such that $\sigma(B_{s^i}, i)$ (resp., $\sigma(B_{e^i}, i)) \neq \emptyset$, and $\mathscr{B}_i = B_{s^i+1}, \ldots, B_{e^i-1}$, the sequence of blocks between B_{s^i} and B_{e^i} (we can assume that M does not contain any row on only `0`'s). Note that $\mathscr{B}_i$ may be empty, however, i.e., the case when $s^i = e^i$, or $e^i = s^i + 1$. Since Property 7 holds, it follows that

(i) if $\mathscr{B}_i$ in row r_i contains two or more `0`'s, then r_i, and hence, M is not (2,1)-C1P; and

(ii) if $\mathscr{B}_i$ in row r_i contains exactly one `0`, then, in B_{s^i} (resp., B_{e^i}) in r_i, there cannot be a single `0` to the right (resp., left) of any `1` in any (2,1)-C1 order of row r_i, and hence, of M. However, this effectively splits the block B_{s^i} (resp.,B_{e^i}) into the two blocks: $B'_{s^i} = \sigma(B_{s^i}, i)$ (resp., $B'_{e^i} = \sigma(B_{e^i}, i)$); and $B''_{s^i} = B_{s^i} \setminus \sigma(B_{s^i}, i)$ (resp., $B''_{e^i} = B_{e^i} \setminus \sigma(B_{e^i}, i)$). We can then replace the set $\mathscr{B} = \{B_1, \ldots, B_{s^i}, \ldots, B_{e^i}, \ldots, B_p\}$ of blocks with the new set $\mathscr{B}' = \{B_1, \ldots, B''_{s^i}, B'_{s^i}, \ldots, B'_{e^i}, B''_{e^i}, \ldots, B_p\}$ of blocks, and the set of $\mathscr{B}'$-block-structured $(2,1)$-C1 orders of matrix M with row r_i removed (as this row is always $(2,1)$-C1P in any $\mathscr{B}'$-block-structured order) will be the same as the set of $\mathscr{B}$-block-structured $(2,1)$-C1 orders of M.

Since both cases (i) and (ii) for row r_i can be determined in time $O(n)$ and space $O(1)$, and hence, in overall time $O(mn)$ and space $O(1)$ for M, we can assume that, in M, these cases do not apply, i.e., that $\sigma(\mathscr{B}_i, i) = \mathscr{B}_i$.

We now explain how to compute $\mathscr{U}_t$ for each $t \in \langle 1, p \rangle$. Since we ruled out cases (i) and (ii) in the previous paragraph, it follows that for each $t \in \langle 1, p \rangle$ and row r_i for $i \in \langle 1, m \rangle$, we have the following set of disjoint cases:

(1) $\sigma(B_t, i) = \emptyset$, i.e., $t \notin \langle s^i, e^i \rangle$;

(2) $\sigma(B_t, i) = B_t$, i.e., $t \in \langle s^i, e^i \rangle$; or

(3) neither (1) nor (2), then $t = s^i$ or e^i, and B_t contains some `0`'s and some `1`'s in row r_i, and

(a) $s^i < e^i$, or

(b) $s^i = e^i$, i.e., B_t is the only block of M where $\sigma(B_t, i) \neq \emptyset$

For $t \in \langle 1, p \rangle$, we will denote S_t^i as the set of permutations of each B_t that are $(2,1)$-C1P with respect to row r_i, for some order of the columns outside of B_t. If either case (1) or (2) holds, then any permutation of the columns of B_t is in S_t^i. In case (3a), when $t = s^i$ (resp., e^i), then, in row r_i, any permutation which does not place more than one `0` to the right (resp., left) of any `1` in B_t is in S_t^i. In case (3b), in row r_i, any $(2,1)$-C1 order of B_t is in S_t^i. Since $|B_t| \leq \ell$, determining if a permutation is in S_t^i takes time $O(\ell)$, and since there are at most $\ell!$ such permutations, computing S_t^i takes time $O((\ell+1)!)$ and space $O(\ell!)$. Then, set

$$\mathscr{U}_t = \bigcap_{i \in \langle 1, m \rangle} S_t^i, \tag{2.1}$$

and computing $\mathscr{U}_t$ for a given $t \in \langle 1, p \rangle$ takes time $O(m\ell!)$ and space $O(\ell!)$. Since p is $O(n)$, computing $\mathscr{U}_t$ for all $t \in \langle 1, p \rangle$ takes time $O(mn\ell!)$ and space $O(n\ell!)$ overall. Note that if $\mathscr{U}_t = \emptyset$ for some t, then $\mathscr{U} = \emptyset$, and hence, by Property 7, M does not have the $(2,1)$-C1P. We remark that if only one block B_1 is associated with M, then $\mathscr{U}_1$ is simply the set of $(2,1)$-C1 orders of M. This completes the details of the preprocessing phase on M to check if it has some necessary properties for it to have a $\{B_1, \ldots, B_p\}$-block-structured $(2,1)$-C1 order.

Up to this point, we have ruled out the trivial cases (i) and (ii) when M does not have such an order and we have computed $\mathscr{U}_t$ for all $t \in \langle 1, p \rangle$, and we can assume that $\mathscr{U}_t \neq \emptyset$. The set of $\mathscr{B}$-block-structured $(2,1)$-C1 orders of M is a subset of $\mathscr{U}$ (Property 7), however it may be the case that it is not equivalent to $\mathscr{U}$, as a choice of one permutation in some $\mathscr{U}_i$ and another in some $\mathscr{U}_j$ might lead to an order which is not $(2,1)$-consecutive. In particular, for any row r_i for $i \in \langle 1, m \rangle$ where $s^i < e^i$, B_{s^i} and B_{e^i} (or neither) can be permuted such that exactly one `0` is to the right (resp., left) of any `1` in row r_i, but both blocks cannot be in this state if r_i is to be $(2,1)$-consecutive. We will express this dependency on the permutations of B_{s^i} and B_{e^i} with a disjunction on two Boolean variables, defined below. For $t \in \langle 1, p \rangle$, let P_t^i be the set of permutations $\pi_t \in \mathscr{U}_t$ that do not place any `0` to the right (resp., left) of any `1` in B_t in row r_i, in the case (3a) when $t = s^i$ (resp., e^i). So that P_t^i is defined for every block/row pair, we let $P_t^i = \mathscr{U}_t$ in cases (1), (2) and (3b) for

$t \in \langle 1,p \rangle$, $i \in \langle 1,m \rangle$. Note that, for a given $t \in \langle 1,p \rangle$ and $i \in \langle 1,m \rangle$ (like with S^i_t) that P^i_t can also be constructed in time $O((\ell+1)!)$ and space $O(\ell!)$, for overall time $O(mn(\ell+1)!)$ and space $O(mn\ell!)$ to compute P^i_t for all $t \in \langle 1,p \rangle$ and $i \in \langle 1,m \rangle$. Let Boolean variable $X_{t,i}$ represent $\pi_t \in P^i_t$, for $t \in \langle 1,p \rangle$, $i \in \langle 1,m \rangle$. It follows that r_i is $(2,1)$-consecutive if and only if it satisfies

$$X_{s^i,i} \vee X_{e^i,i}. \tag{2.2}$$

Note that Equation 2.2 is defined for all rows r_i for $i \in \langle 1,m \rangle$ (in the case that $s^i = e^i$, Equation 2.2 is a tautology by the fact that $\mathscr{U}_t \neq \emptyset$ for all $t \in \langle 1,p \rangle$).

In the previous paragraph, we saw that, for any given row r_i, there is a one-to-one correspondence between satisfying truth assignments to Equation 2.2 and $(2,1)$-C1 orders of r_i. However, the same correspondence between $(2,1)$-C1 orders of M and satisfying truth assignments of

$$\bigwedge_{i \in \langle 1,m \rangle} X_{s^i,i} \vee X_{e^i,i} \tag{2.3}$$

does not hold in general. This is due to the fact that when a set $A \subseteq \langle 1,m \rangle$ of two or more rows are involved, one of $\{s^i, e^i\}$ for each r_i, $i \in A$ can coincide on the single block B_t, and $\bigcap_{i \in A} P^i_t = \emptyset$. This means that a truth assignment τ to the pairs of variables $X_{t,i}$ corresponding to the rows of A may not be *valid*, where we say that a truth assignment τ is valid when there is a $\mathscr{B}$-block-structured order $\pi = \pi_1, \ldots, \pi_p$ of the columns of M such that $\tau(X_{t,i}) = \mathit{true}$ if and only if $\pi_t \in P^i_t$ for all $t \in \langle 1,p \rangle$, $i \in \langle 1,m \rangle$. So, in addition to τ satisfying Equation 2.3, we must also ensure that τ is valid. This can be done simply by ensuring for any such set of rows A where $\bigcap_{i \in A} P^i_t = \emptyset$ for some $t \in \langle 1,p \rangle$, that not all $X_{t,i}$, $i \in A$ are set to true, which can be encoded by

$$\bigwedge_{t \in \langle 1,p \rangle} \bigwedge_{A \subseteq \langle 1,m \rangle \bigcap_{i \in A} P^i_t = \emptyset} \bigvee_{i \in A} \neg X_{t,i}. \tag{2.4}$$

While the (2,1)-C1 orders of M correspond to the satisfying assignments of Equation 2.4, this SAT formulation can have clauses of size as large as m, since A can be as large as m.

We now give the following condition of M, which can be checked in polynomial-time $O(mn\ell + n\ell!2^{3\ell})$. If the condition holds, then Equation 2.4 can be replaced by an equivalent set of 2-clauses, and if the condition does not hold, then an NP-completeness proof can be constructed.

Condition 8. *For every $t \in \langle 1,p\rangle$, and $A \subseteq \langle 1,m\rangle$, $\bigcap_{i\in A} P_t^i = \emptyset$ implies that there exists $i,j \in A$ such that $P_t^i \cap P_t^j = \emptyset$.*

It follows that if this condition holds, then for every $t \in \langle 1,p\rangle$, it is sufficient to forbid $X_{t,i}$ and $X_{t,j}$ from both being *true* in τ for every pair $i,j \in \langle 1,m\rangle$ such that $P_t^i \cap P_t^j = \emptyset$. We can hence replace Equation 2.4 with

$$\bigwedge_{t\in\langle 1,p\rangle} \bigwedge_{i,j\in\langle 1,m\rangle P_t^i \cap P_t^j = \emptyset} \neg X_{t,i} \vee \neg X_{t,j}. \tag{2.5}$$

Clearly this is a 2SAT formulation of polynomial-size $O(m^2n)$. Note that Equation 2.5 can be constructed in polynomial-time $O(m^2n\ell!)$.

We now show how to perform the polynomial-time check to see if this condition holds for the particular instance M, and how to construct an NP-completeness proof if Condition 8 does not hold. To check this condition, we have to check for every $t \in \langle 1,p\rangle$ if there is an $A \subseteq \langle 1,m\rangle$ with $|A| > 2$ such that $\bigcap_{i\in A} P_t^i = \emptyset$. For a given $t \in \langle 1,p\rangle$, $|B_t| \leq \ell$, and since M is a binary matrix, there are only 2^ℓ unique rows in B_t. It takes time $O(m\ell + \ell 2^\ell)$ to find this set of unique rows and space $O(2^\ell)$ to store (or index) this set. From this set, there are 2^{2^ℓ} choices for A. For each choice of A, we have to compute $\bigcap_{i\in A\subseteq\langle 1,m\rangle} P_t^i$ which takes time $O(\ell!2^\ell)$. For each of these intersections, we have to compute $P_t^i \cap P_t^j$ for each $i,j \in A$. Since computing $P_t^i \cap P_t^j$ takes time $O(\ell!)$, and there are $2^{2\ell}$ pairs i,j, this step takes time $O(\ell!2^{2\ell})$. Hence, the time of this check for a given $t \in \langle 1,p\rangle$ is $O(m\ell + \ell 2^\ell + 2^\ell \cdot (\ell!2^\ell + \ell!2^{2\ell}))$ which simplifies to $O(m\ell + \ell!2^{3\ell})$. Again, since p is $O(n)$, this check takes overall time $O(mn\ell + n\ell!2^{3\ell})$. Since P_t^i for each $t \in \langle 1,p\rangle$ and $i \in \langle 1,m\rangle$ has already been computed previously, the only additional space used is $O(2^\ell)$ to store the set of unique rows for the current B_t, and $O(1)$ for the pair i,j, and hence this check uses space $O(2^\ell)$.

Now, suppose that we find a set $A \subseteq \langle 1,m\rangle$ with $|A| > 2$ such that $\bigcap_{i\in A} P_t^i = \emptyset$. For simplicity, let A be the set of rows $\{r_1, r_2, r_3\}$. If this is the case, it follows that for some $t \in \langle 1,p\rangle$, $P_t^1 \cap P_t^2 \cap P_t^3 = \emptyset$, while for any pair $\{i,j\} \subset \{1,2,3\}$ where

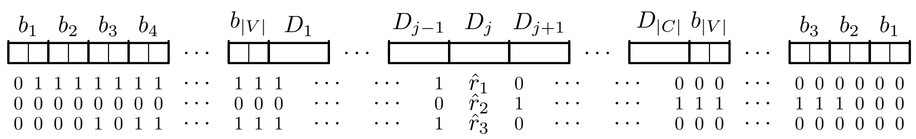

b_1 b_2 b_3 b_4 $b_{|V|}$ D_1 D_{j-1} D_j D_{j+1} $D_{|C|}$ $b_{|V|}$ b_3 b_2 b_1
0 1 1 1 1 1 1 1 ⋯ 1 1 1 ⋯ ⋯ 1 $\hat{r}_1$ 0 ⋯ ⋯ 0 0 0 ⋯ 0 0 0 0 0 0
0 0 0 0 0 0 0 0 ⋯ 0 0 0 ⋯ ⋯ 0 $\hat{r}_2$ 1 ⋯ ⋯ 1 1 1 ⋯ 1 1 1 0 0 0
0 0 0 0 1 0 1 1 ⋯ 1 1 1 ⋯ ⋯ 1 $\hat{r}_3$ 0 ⋯ ⋯ 0 0 0 ⋯ 0 0 0 0 0 0

Figure 2.3: The structure of the construction for a 3CNF formula ϕ on the set V of variables and C of clauses, along with the 3 rows encoding the clause $c_j = \{v_1 \vee v_2 \vee \neg v_3\}$. The blocks $b_1, \ldots, b_{|V|}$ correspond to the variables of ϕ in exactly the same way as in the construction of Subsection 2.3.1. The blocks $D_1, \ldots, D_{|C|}$ correspond to the clauses. Here, for $i \in \{1,2,3\}$, $\hat{r}_i$ is row r_i restricted to the columns of B_t, and P_t^1, P_t^3 (resp., P_t^2) are sets of permutations that do not place any `0` to the left (resp., right) of any `1` in B_t in rows r_1, r_3 (resp., r_2). It follows that all truth assignments to the literals of c_j are (2,1)-C1 orders except for the case when all 3 literals are false (c_j is not satisfied), since $P_t^1 \cap P_t^2 \cap P_t^3 = \emptyset$. Note that for each $i \in \{1, \ldots, |V|\}$, rows can be added to force the copy of variable block b_i on the left and right of the clause blocks to encode the same truth value.

$i \neq j$, $P_t^i \cap P_t^j \neq \emptyset$. This property allows us to use A to build a 3-clause gadget, for a reduction from 3SAT, similar to that of Subsection 2.3.1, to the problem of deciding if M has a $\mathscr{B}$-block-structured (2,1)-C1 order. Figure 2.3 illustrates the structure of this construction along with the 3 rows that would be added for the clause $c_j = \{v_1 \vee v_2 \vee \neg v_3\}$. Note that if $|A| > 3$, then a $|A|$-clause gadget can be built in a similar fashion. Since $|A| \leq 2^\ell$, this construction is of size polynomial in $\|M\|$, and hence deciding the (2,1)-C1P would be NP-complete if Condition 8 does not hold for some M.

Finally, we summarize the time and space complexity of this algorithm. The preprocessing phase, i.e., checking cases (i) and (ii) for each row of M takes time $O(mn)$ and space $O(1)$. For each $t \in \langle 1, p \rangle$ and $i \in \langle 1, m \rangle$, computing S_t^i (and P_t^i) takes time $O((\ell+1)!)$ and space $O(\ell!)$, for overall time $O(mn(\ell+1)!)$ and space $O(mn\ell!)$ for this step. Computing $\mathscr{U}_t$ for all $t \in \langle 1, p \rangle$ takes time $O(mn\ell!)$ and space $O(n\ell!)$ overall. Performing the check for Condition 8 takes time $O(mn\ell + n\ell!2^{3\ell})$ and space $O(2^\ell)$. If the condition holds, then it computes Equation 2.5, which takes time $O(m^2 n\ell!)$, generating a 2SAT formulation of size $O(m^2 n)$. In summary, it follows that this algorithm runs in time $O(m^2 n(\ell+1)! + n\ell!2^{3\ell})$ and space $O(m^2 n\ell! + 2^\ell)$.

Theorem 9. *Given binary matrix M on n columns and m rows and a collection $\mathscr{B} = \{B_1, \ldots, B_p\}$ of sets of columns of M where $|B_t| \leq \ell$ for all $t \in \langle 1, p \rangle$ for some fixed constant number ℓ, there is an algorithm that runs in polynomial-time $O(m^2 n(\ell+1)! + n\ell! 2^{3\ell})$ and space $O(m^2 n\ell! + 2^\ell)$ which either*

(a) decides if there is $\mathscr{B}$-block-structured order of M that is also a (2,1)-C1 order, or

(b) finds a proof that deciding the (2,1)-C1P is NP-complete.

While this algorithm checks Condition 8 for the particular instance M, we conjecture that Condition 8 holds for all binary matrices. If this is the case, as a corollary of Theorem 9, we could omit the check of this condition for a faster algorithm.

Corollary 10. *If Condition 8 holds for all binary matrices, then given binary matrix M on n columns and m rows and a collection $\mathscr{B} = \{B_1, \ldots, B_p\}$ of sets of columns of M where $|B_t| \leq \ell$ for all $t \in \langle 1, p \rangle$ for some fixed constant number ℓ, there is an algorithm that runs in polynomial-time $O(m^2 n(\ell+1)!)$ and space $O(m^2 n\ell!)$ which decides if there is $\mathscr{B}$-block-structured order of M that is also a (2,1)-C1 order.*

2.5 The Complexity of Deciding the (∞, δ)-C1P

Here we show that for every $\delta \geq 1$, deciding the (∞, δ)-C1P is NP-complete. The first step is to reduce 3SAT(3), the version of the 3SAT Problem where no variable appears more than twice positively and more than once negatively to an auxiliary version of the 3SAT Problem. We then reduce this auxiliary version to the problem of deciding the (∞, δ)-C1P for the result.

2.5.1 The 3SAT(L:2,R:2) Problem

First we reduce from 3SAT(3), the version of the 3SAT Problem with 2-clauses and 3-clauses, and where no variable appears more than twice positively and more than once negatively [120, p. 183, Prop. 9.3], to an auxiliary version of the 3SAT

Problem, namely 3SAT(L:2,R:2): the version of the 3SAT Problem with 2-clauses and 3-clauses, where each clause is assigned the label L or R (for left or right) such that for each label, no variable appears more than once positively and more than once negatively in the corresponding set of clauses.[1]

Lemma 11. *The 3SAT(L:2,R:2) Problem is NP-complete.*

Proof. We are given an instance to the 3SAT(3) Problem: a set V of variables and C of 2 and 3-clauses, such that for each $v \in V$, v appears no more than twice in C and $\neg v$ appears no more than once in C. For each $v \in V$ with two positive occurrences, we replace one of the occurrences of v with the new variable v'. We then label all the clauses of this new instance with L. Note that in this set of clauses labelled with L, no variable appears more than once positively and once negatively. Now, for each appearance of v', we add the two new clauses $c_v^1 = v' \vee \neg v$ and $c_v^2 = v \vee \neg v'$, and label them both with R. These two clauses enforce the constraint that $v = v'$ in any satisfying assignment to this new instance of the 3SAT Problem, thus this new instance is satisfiable if and only if the original 3SAT Problem instance is satisfiable. This new instance of the 3SAT Problem has 2- and 3-clauses, and for each of the labels L and R, no variable appears more than once positively and once negatively. Thus we have transformed in polynomial time the instance of the 3SAT(3) Problem to an instance of the 3SAT(L:2,R:2) Problem that is satisfiable if and only if the original 3SAT(3) instance is satisfiable. Since the 3SAT(L:2,R:2) Problem is clearly in NP, it follows that the 3SAT(L:2,R:2) Problem is NP-complete. □

2.5.2 The Complexity of Deciding the $(\infty,1)$-C1P

We now show that the problem of deciding the $(\infty,1)$-C1P is NP-Complete by giving a reduction from 3SAT(L:2,R:2). We will later generalize this reduction to show that for every $\delta \geq 1$, deciding the (∞,δ)-C1P is NP-Complete.

Theorem 12. *Deciding the $(\infty,1)$-C1P is NP-complete.*

[1]We remark that the exact formulation of 3SAT(3) in Papadimitriou [120] allows also variables with one positive and two negated occurrences, however these can easily be converted to the other type of variables by replacing them with their negations in all clauses. Clearly, this does not affect the complexity of the problem.

Proof. We are given an instance ϕ of the 3SAT(L:2,R:2) Problem: a set V of variables and the sets C^L and C^R of 2- and 3-clauses, such that for each $v \in V$, v and $\neg v$ each appear no more than once in C^S, for $S \in \{L,R\}$. We use ϕ to build a matrix M_ϕ such that ϕ is satisfiable if and only if M_ϕ has the $(\infty,1)$-C1P.

The idea of the construction is that for each variable $v_i \in V = \{v_1,\ldots,v_n\}$, the matrix M_ϕ will have the block of columns b_i, called the *variable block*, to represent the value of this variable. Matrix M_ϕ will also contain the blocks of columns $b_{0,1},\ldots,b_{n,n+1}$ of *dummy blocks* that will interleave the variable blocks. We will add some rows to M_ϕ to force the individual columns of each of the variable and dummy blocks to appear together and in fixed order, or the reverse order. The direction of block b_i will represent the value of the variable v_i. We will then add some rows to M_ϕ to force only the order $b_{0,1},b_1,b_{1,2},\ldots,b_{n-1,n},b_n,b_{n,n+1}$ (or the reverse order) of these blocks, while the individual variable blocks may switch direction relative to this order. If variable block b_i is in the same order relative to this order of all of the blocks then its corresponding variable v_i has value *true*, otherwise it has value *false*. The matrix M_ϕ will also have an additional $2n$ free columns. To each clause $c \in C = \{C^L \cup C^R\}$ we associate a unique empty free column f_c. This is possible since for every $S \in \{L,R\}$, each variable appears no more than once positively and once negatively in C^S, and each $c \in C^S$ contains at least 2 variables, and hence $|C^S| \leq 2n/2 = n$. Thus $|C^L| + |C^R| \leq 2n$. We then add some rows to M_ϕ to force these $2n$ free columns to fall (in any order) between the $2n$ pairs of adjacent $b_{i-1,i},b_i$ and $b_i,b_{i,i+1}$ blocks, for $i \in \langle 1,n\rangle$, such that there is one free column for each hole.

For a clause $c \in C^L$ (resp., C^R) where c contains variables v_α, v_β (and v_γ for a 3-clause), we assign this clause to column f_c of the $2n$ free columns, and we add a row to M_ϕ that forces the column f_c to be to the left (resp., right) of either block b_α, b_β (or b_γ for a 3-clause). However, column f_c can only go to the left (resp., right) of the block of a variable when its corresponding literal is set to the value that satisfies clause c. Note by the construction that each variable can satisfy at most one left and one right clause, which is sufficient because each literal appears at most once in a right (resp., left) clause. These properties will imply that only when, for every $c \in C^L$ (resp., C^R), column f_c can be placed to the left (resp., right) of a b_i, for $i \in \langle 1,n\rangle$, for a v_i that is set to a value that satisfies c, i.e., ϕ is satisfied,

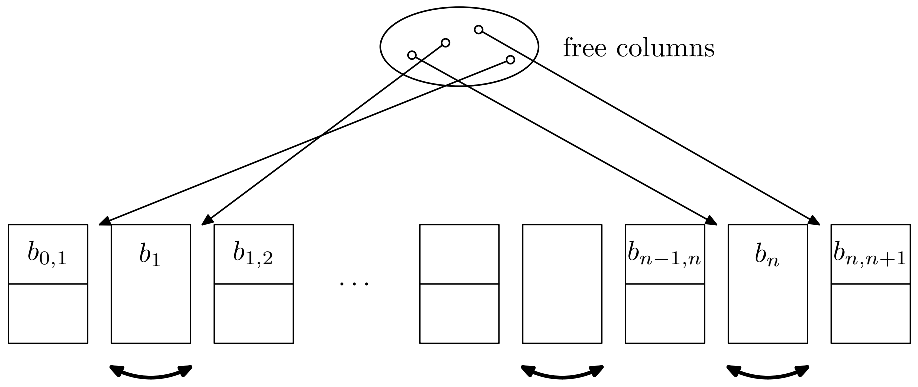


Figure 2.4: The structure of matrix M_ϕ.

is there a $(\infty,1)$-C1 order of M_ϕ, and vice versa. We now give the full details of the construction in what follows.

For each variable $v_i \in V = \{v_1,\dots,v_n\}$, we add the set of columns $b_i = \{b_i^1,\dots,b_i^5\}$ to M_ϕ. In addition, for every $i \in \langle 1,n\rangle$, we add the set of columns $b_{i-1,i} = \{b_{i-1,i}^1,\dots,b_{i-1,i}^5\}$ to M_ϕ. For each set of columns b_i for $i \in \langle 1,n\rangle$, and $b_{i-1,i}$ for $i \in \langle 1,n+1\rangle$, we add to M_ϕ the rows according to Theorem 4 to force the columns of each set to appear together and in fixed order (or the reverse) in any $(\infty,1)$-C1 order of M_ϕ, i.e., in any $(\infty,1)$-C1 order of M_ϕ, set b_i will appear either as the sequence $b_i^1,\dots,b_i^5$ or $b_i^5,\dots,b_i^1$ of consecutive columns, and similarly for the columns in sets $b_{i-1,i}$. We will refer to the b_i as *variable blocks* and the $b_{i-1,i}$ as *dummy blocks*. Note that Theorem 4 requires that a set of columns must have size $2\delta+3$ before such an order can be enforced on it, this is why each block is of size five. In addition, we add $2n$ free columns to M_ϕ.

Now, for each pair of blocks $b_{i-1,i}, b_i$ and $b_i, b_{i,i+1}$ for $i \in \langle 1,n\rangle$, we add rows $[b_{i-1,i} \setminus \{b_{i-1,i}^1\} \cup b_i]$ and $[b_i \cup b_{i,i+1} \setminus \{b_{i,i+1}^5\}]$ to force these pairs to be together with at most one free column in between them. This enforces that the blocks appear in the order $b_{0,1}, b_1, b_{1,2},\dots,b_{n-1,n}, b_n, b_{n,n+1}$ (or the reverse) in any $(\infty,1)$-C1 order of M_ϕ. The first (resp., last) column of the dummy blocks is omitted to fix their direction (relative to the order of the blocks) under the assumption that there is a free column between each pair of neighboring blocks, which we will now enforce with the following row. We add to M_ϕ the row $[B \cup F]$, where $B = b_{0,1}^{\langle 2,4\rangle} \cup b_1^{\langle 2,4\rangle} \cup$

$\cdots \cup b_n^{\langle 2,4 \rangle} \cup b_{n,n+1}^{\langle 2,4 \rangle}\}$, and F is the set of $2n$ free columns. It now follows that between each $b_{i-1,i}, b_i$ and $b_i, b_{i,i+1}$ pair for $i \in \langle 1,n \rangle$, there must lie at least one column from F, in any $(\infty,1)$-C1 order of M_ϕ. Since we have exactly $2n$ pairs, between each pair there must be exactly one. Figure 2.4 depicts all $(\infty,1)$-C1 orders of the current matrix M_ϕ. Note that the columns in each variable block can be oriented either in the same direction as the order of all of the blocks, or in the reverse direction. If variable block b_i is oriented in the same direction as the order of all of the blocks, this corresponds to the setting of the variable v_i to *true*, while the reverse direction corresponds to v_i being *false*. Now it remains to add rows to M_ϕ to force the free column associated with each clause to fall next to only the blocks of variables that are set to a value that satisfies the clause.

Let $c \in C^L$ (resp., C^R) contain the variables x_α, x_β (and x_γ for a 3-clause), and let $f_c \in F$ be the free column associated with clause c. We add the row $[B \cup F \setminus \{f_c\} \cup S_c]$ to M_ϕ, where S_c is defined as follows. If $c \in C^L$, then for each $j \in \{\alpha, \beta\}$ ($j \in \{\alpha, \beta, \gamma\}$ for a 3-clause), if v_j appears positively (resp., negatively) in c, set S_c contains the columns $\{b_{j-1,j}^5, b_j^1\}$ (resp., $\{b_{j-1,j}^5, b_j^5\}$). Otherwise, if $c \in C^R$, then for each j, if v_j appears positively (resp., negatively) in c, set S_c contains the columns $\{b_j^5, b_{j,j+1}^1\}$ (resp., $\{b_j^1, b_{j,j+1}^1\}$). Adding these extra ones around the variable blocks b_j for each j forces f_c to fall only to the immediate left (resp., right) of these b_j in any $(\infty,1)$-C1 order of M_ϕ. Furthermore, f_c can only fall to the immediate left (resp., right) of a b_j if it is oriented in a direction such that corresponding variable v_j is set to a value that sets its literal to *true*, i.e., if v_j satisfies c. Hence, the satisfying assignments of any individual clause c correspond to the $(\infty,1)$-C1 orders of the submatrix of M_ϕ consisting of the row added for clause c, and all of the rows previously added to M_ϕ for the blocks b_i for $i \in \langle 1,n \rangle$, and $b_{i-1,i}$ for $i \in \langle 1,n+1 \rangle$.

After adding the row for all clauses $c \in C^L \cup C^R$, the set of remaining $(\infty,1)$-C1 orders of M_ϕ (if there exist any) correspond to the cases where for every clause $c \in C^L$ (resp., C^R), its corresponding column f_c is placed to the immediate left (resp., right) of a block of a variable that is set to a value (*true* or *false*) that satisfies c, that is, to satisfying assignments of ϕ. Conversely, if ϕ has a satisfying assignment, then we can assign each $c \in C^L$ (resp., C^R) to a unique $v \in V$ that satisfies c, in the sense that either v or $\neg v$ satisfies c, i.e., each $v \in V$ will satisfy at most one clause from C^L and at most one clause from C^R. We can make this claim because v and $\neg v$ each appear no more than once in C^L (resp., C^R), and at most one of v and $\neg v$ satisfies a given clause c. Thus we can assign each column f_c of M_ϕ to a unique slot to the immediate left (resp., right) of block b_i for $i \in \langle 1, n \rangle$, for the corresponding v_i that satisfies the clause c. Thus M_ϕ has a $(\infty, 1)$-C1 order. Hence, ϕ is satisfiable if and only if M_ϕ has the $(\infty, 1)$-C1P.

In summary, given a 3SAT(L:2,R:2) formula ϕ with n variables and $m \leq 2n$ clauses, we have constructed a matrix M_ϕ with $12n+5$ columns and $16n+m+8$ rows such that M_ϕ has the $(\infty, 1)$-C1P if and only if ϕ is satisfiable. Given that deciding the $(\infty, 1)$-C1P is clearly in NP, and Lemma 11, it follows that deciding the $(\infty, 1)$-C1P is NP-complete. $\square$

2.5.3 The Complexity of Deciding the (∞, δ)-C1P

We now generalize the construction given in Subsection 2.5.2 to show that for every $\delta \geq 1$, the problem of deciding the (∞, δ)-C1P is NP-complete by reduction from 3SAT(L:2,R:2).

Theorem 13. *For every $\delta \geq 1$, deciding the (∞, δ)-C1P is NP-complete.*

Proof. Consider $\delta \geq 1$. Here, given an instance ϕ of 3SAT(L:2,R:2), we build a matrix M_ϕ such that ϕ is satisfiable if and only if M_ϕ has the (∞, δ)-C1P. The idea of the construction is the same as that of the proof of Theorem 12: it will again have the blocks b_i for $i \in \langle 1, n \rangle$, and $b_{i-1,i}$ for $i \in \langle 1, n+1 \rangle$ as well as $2n$ free columns for the clauses, only the blocks will need more columns, and we will need to add more rows to M_ϕ in order for it to behave in the same way for arbitrary δ.

For each block b_i for $i \in \langle 1, n \rangle$, and $b_{i-1,i}$ for $i \in \langle 1, n+1 \rangle$ we again add to M_ϕ the rows according to Theorem 4 to force each individual block to be in fixed order (or the reverse) in any (∞, δ)-C1 order of M_ϕ. Thus, each block will contain $2\delta+3$ columns. In order to force each pair of blocks $b_{i-1,i}, b_i$ and $b_i, b_{i,i+1}$ for $i \in \langle 1, n \rangle$, to be together, with at most one free column in between them, thus enforcing a total order on the blocks, we add the rows $[b_{i-1,i}^{\langle \delta+1, \delta+4 \rangle} \cup b_i]$ and $[b_i \cup b_{i,i+1}^{\langle \delta, \delta+3 \rangle}]$. Note here, that the first (resp., last) δ columns of the dummy blocks are omitted to fix their direction (relative to the order of the blocks) under the assumption that there is a free column between each pair of neighboring blocks, which we enforce by adding to M_ϕ the row $[B \cup F]$, where $B = b_{0,1}^{\langle\delta+1,\delta+3\rangle} \cup b_1^{\langle\delta+1,\delta+3\rangle} \cup \cdots \cup b_n^{\langle\delta+1,\delta+3\rangle} \cup b_{n,n+1}^{\langle\delta+1,\delta+3\rangle}$, and F is a set of $2n$ free columns. Now M_ϕ again has the desired structure, as depicted in Figure 2.4. Now it remains to add rows to M_ϕ for the clauses.

Let $c \in C^L$ (resp., C^R) contain the variables x_α, x_β (and x_γ for a 3-clause), and let $f_c \in F$ be the free column associated with clause c. We add the row $[B \cup F \setminus \{f_c\} \cup S_c]$ to M_ϕ, where S_c is defined as follows. If $c \in C^L$, then for each $j \in \{\alpha, \beta\}$ ($j \in \{\alpha, \beta, \gamma\}$ for a 3-clause), if v_j appears positively (resp., negatively) in c, set S_c contains the columns $\{b_{j-1,j}^{2\delta+3}, b_j^1\}$ (resp., $\{b_{j-1,j}^{2\delta+3}, b_j^{2\delta+3}\}$). Otherwise, if $c \in C^R$, then for each j, if v_j appears positively (resp., negatively) in c, set S_c contains the columns $\{b_j^{2\delta+3}, b_{j,j+1}^1\}$ (resp., $\{b_j^1, b_{j,j+1}^1\}$). Now this matrix M_ϕ will have the same behavior as in the proof of Theorem 12, hence ϕ is satisfiable if and only if M_ϕ has the (∞, δ)-C1P.

In summary, for every $\delta \geq 1$, given a 3SAT(L:2,R:2) formula ϕ with n variables and $m \leq 2n$ clauses, we have constructed a matrix M_ϕ with $(4\delta+8)n + 2\delta + 3$ columns and $(6\delta+10)n + m + 3\delta + 4$ rows such that M_ϕ has the (∞, δ)-C1P if and only if ϕ is satisfiable. Given that for every $\delta \geq 1$, deciding the (∞, δ)-C1P is clearly in NP, and Lemma 11, it follows that for every $\delta \geq 1$, deciding the (∞, δ)-C1P is NP-complete. □

Chapter 3

The Gapped Consecutive-Ones Property for Matrices of Bounded Maximum Degree

In this chapter, we study the (k,δ)-C1P with a third parameter d, the bound on the maximum degree of M. In Section 3.1 we first provide an algorithm for the case of the (d,k,δ)-C1P when all three parameters are fixed constants. In Section 3.2, we show, in four subsections, that deciding the (d,k,∞)-C1P for every $d > k \geq 2$ is NP-complete. First, in Subsection 3.2.1, we give the definition of a type of hypergraph covering problem. In Subsection 3.2.2 we show that a special case of this hypergraph covering problem is NP-complete, and then in Subsection 3.2.3 we generalize this construction to show that the general case of this hypergraph covering problem is NP-complete. Finally, in Subsection 3.2.4 we show a direct correspondence of the general case of this hypergraph covering problem to deciding the (d,k,∞)-C1P for every $d > k \geq 2$ to give the result of this Section 3.2.

3.1 An Algorithm for Matrices of Bounded Maximum Degree

A binary matrix M has maximum degree d if every row contains at most d entries `1`. We show now that, when d and δ are constant (which implies that k is also constant, since $k \leq d$), then deciding the (k,δ)-C1P is tractable. We rely on a connection to graph bandwidth, and an algorithm of Saxe [135] for deciding graph bandwidth. We now give several definitions, theorems and eventually the algorithm from Saxe [135], and our extensions to these to give an algorithm for deciding the (d,k,δ)-C1P.

We first define a *layout* of a graph, or a mapping of its vertices to distinct positive integers, and then the *bandwidth* of a layout.

Definition 14 (Layout of a Graph)**.** Saxe [135] *Let $G = (V,E)$ be a graph with $|V| = n$. A* layout *of G is a one-to-one mapping $f : V \to \langle 1,n\rangle$.*

Definition 15 (Bandwidth of a Layout)**.** Saxe [135] *Given graph $G = (V,E)$ with $|V| = n$, and a layout f of G, the* bandwidth *of f is defined as the maximum distance between the images under f of any two vertices that are connected by an edge in G. That is,*

$$\text{bandwidth}(f) = \max\{f(u) - f(v) \mid \{u,v\} \in E\}.$$

The bandwidth of a graph is then the smallest bandwidth for any of its layouts.

Definition 16 (Bandwidth of a Graph)**.** Saxe [135] *Given graph $G(V,E)$ with $|V| = n$,*

$$\text{bandwidth}(G) = \min\{\text{bandwidth}(f) \mid f \text{ is a layout of } G\}.$$

We now show the connection of graph bandwidth to the (d,k,δ)-C1P. Let M be an $m \times n$ binary matrix and $G_M = (V_M, E_M)$ be the undirected graph defined as follows: $V_M = \langle 1,n\rangle$ (each vertex of G_M represents a column of M), and there is an edge $\{i,j\} \in E_M$ if and only if there is a row of M with entries `1` in columns i and j. The following property then follows immediately from this definition:

Property 17. *If M has maximum degree d and M has the (k,δ)-C1P, then* bandwidth(G_M) *is at most $d + (k-1)\delta - 1$.*

We hence denote a *layout* of a binary matrix M to be a layout f of its G_M, while the *bandwidth* of such a layout is the bandwidth of f (where the domain of this layout of M is its columns $\langle 1,n\rangle$ corresponding to the vertices of G_M that form the domain of f). We then denote that the *bandwidth*, bandwidth(M), of a binary matrix M

is bandwidth(G_M). It is an algorithm for deciding for some given graph G, if bandwidth$(G) \leq b$ for some fixed constant b, that is the main result of Saxe [135]. In the following, we give the details of this algorithm, and how it can be extended to give an algorithm for deciding the (d,k,δ)-C1P. This relies, of course, on the above Property 17. We first need to give some of the preliminary assumptions, definitions and theorems (and their extensions for our purposes) of Saxe [135].

First, we note that if graph $G = (V,E)$ with $|V| = n$ is not connected, then G has a layout of bandwidth $\leq b$ if and only if each of its components has such a layout. Also, it is clearly impossible for G to have such a layout if G has any vertex of degree greater than $2b$. We therefore assume that G is (i) connected and (ii) has no vertex of degree greater than $2b$. Since b is a fixed constant, we can determine (ii) in linear time $O(n)$, and, given that (ii) holds, we can determine (i) in linear time as well [135]. Similarly, we assume that any matrix M given as input to deciding the (d,k,δ)-C1P emits a graph G_M that has properties (i) and (ii).

We now introduce the key notion from Saxe [135] of a *partial layout*, some related definitions with respect to deciding if a graph has bandwidth $\leq b$ where b is a fixed constant, and our extensions of some of these definitions so that we can later extend the algorithm of Saxe [135] to obtain an algorithm for deciding if a binary matrix has the (d,k,δ)-C1P.

Definition 18 (Partial Layout of a Graph)**.** Saxe [135] *Let $G = (V,E)$ be a graph with $|V| = n$. A* partial layout *of G is a one-to-one mapping $f : U \rightarrow \langle 1, p\rangle$, where $U \subseteq V$ and $|U| = p$, i.e., $0 \leq p \leq n$.*

Definition 19 (Feasible Partial Layout)**.** Saxe [135] *We say that a partial layout f of a graph G is* feasible *if it can be extended to a (total) layout g, such that* bandwidth$(g) \leq b$*.*

Definition 20 (Bandwidth of a Partial Layout)**.** Saxe [135] *The* bandwidth *of a partial layout f of a graph G is the maximum distance between the images of any two edge-connected vertices of G which are in the domain of f.*

Definition 21 (Edge Dangling from a Partial Layout)**.** Saxe [135] *Given partial layout f of a graph $G = (V,E)$, if $\{u,v\} \in E$ and u is in the domain of f and v is not, then edge $\{u,v\}$ is said to be* dangling *from f.*

Here we denote a *partial layout* of a binary matrix M to be a partial layout of its G_M. Note that this emits a submatrix M' of M on the columns U. The remainder of these definitions carry over directly to matrices M, i.e., in terms of G_M, with the exception of feasibility, which is a bit more complicated.

Definition 22 (Feasible Partial Layout of a Binary Matrix)**.** *We say that a partial layout f of a binary matrix M is* feasible *if it is a feasible partial layout of G_M, and if it can be extended to a (total) layout g, such that* $\text{bandwidth}(g) \leq b$*, and the order of the columns of M given by $g^{-1}(1), \ldots, g^{-1}(n)$ is a (k, δ)-C1 order.*

We now introduce the notions from Saxe [135] of a *plausible partial layout*, and the *active region* of a partial layout.

Definition 23 (Plausible Partial Layout of a Graph)**.** Saxe [135] *Given partial layout f of a graph $G = (V, E)$, where f is of size p, it is clear that f cannot be feasible unless*

(1) $\text{bandwidth}(f) \leq b$*, and*

(2) whenever u and v are vertices of G such that $f(u) < p - b$ and $\{u, v\} \in E$, then v is also in the domain of f.

If f satisfies both of these conditions, then f is said to be a plausible partial layout.

In order to extend this above definition so that it holds for also binary matrices, we have to add to it the following third and fourth properties

(3) submatrix M' given by f has the (k, δ)-C1P, and

(4) for each row r of M, if the degree of r in M' is less than its degree in M, then

  (a) r in M' has the $(k - 1, \delta)$-C1P, and

  (b) the rightmost `1` in r of M' is followed by at most δ `0`'s.

Note that $G = G_M$ in (2) of the above property. Finally, we give the following definition of *active region* which carries over directly to the case of binary matrices.

Definition 24 (Active Region of a Partial Layout)**.** Saxe [135] *Given partial layout f of a graph G, where f is of size p, the sequence $(f^{-1}(\max(p-b+1,1)),\ldots,f^{-1}(p))$ taken together with the set of dangling edges of f is called the* active region *of f.*

We now present the theorem of Saxe [135] on which Saxe's principal algorithm depends.

Theorem 25. Saxe [135] *Let f and g be two plausible partial layouts of G having identical active regions. Then,*

(1) f and g have identical domains, and

(2) f is feasible if and only if g is feasible.

Proof. Since G is connected, the domains of f and g must each consist precisely of those vertices which are path-connected to vertices in the active region by paths not including any dangling edges. Thus, (1) holds. To see that (2) holds, we need only note that any assignment of the remaining vertices which extends either f or g to a total layout of bandwidth $\leq b$ must also extend the other to such a layout. □

Note that since we defined active region and what it means for a partial layout of a binary matrix to be feasible and plausible, that Theorem 25 carries over to the case of a binary matrix M also, where $G = G_M$ and the assignment that extends either f and g to a total layout of bandwidth $\leq b$ also has the (k,δ)-C1P in the proof of this theorem.

Finally, we present the notions from Saxe [135] of a *successor* and *predecessor* of a plausible partial layout.

Definition 26 (Successor of a Plausible Partial Layout)**.** Saxe [135] *Let f be a plausible partial layout of G. Then a* successor *of f is a plausible partial layout g which extends f by precisely one element. In this case, the active region of g is also said to the be the successor of the active region of f.*

Definition 27 (Predecessor of a Plausible Partial Layout)**.** Saxe [135] *If plausible partial layout g is the successor of plausible partial layout f, then (the active region of) f is a* predecessor *of (the active region of) g.*

Again, because we have defined all of these notions in the case of binary matrices, these notions of successor and predecessor also carry over directly to the case of binary matrices.

As in Saxe [135], Theorem 25 allows us to say that two plausible partial layouts of a binary matrix M are *equivalent* if they have identical active regions. We can now easily extend the algorithm of Saxe to obtain a breadth-first search over the space of all induced equivalence classes of plausible partial layouts, i.e., the active regions. Here, again, since each active region consists of at most b vertices and each vertex has no more than $2b$ edges, each of which may or may not be dangling, the number of equivalence classes is bounded above by

$$\sum_{0 \leq i \leq b} \binom{n}{i} (i!)(2^{2b})^i = O(n^b). \tag{3.1}$$

We are now ready to present our extension of the algorithm of Saxe [135] for deciding if a binary matrix M has the (d,k,δ)-C1P. Here, we need only extend Saxe's algorithm with a data structure that stores a submatrix M' corresponding to each active region, and some procedures associated with this submatrix to test for the (k,δ)-C1P of this active region. Note that since we assume that M has bounded degree d, by Property 17 we can test here for bandwidth $\leq b$, where $b = d + (k-1)\delta - 1$. Note also that, by the definition of G_M, it follows that b is greater than the distance between the leftmost `1` and rightmost `1` in any row of any (k,δ)-C1 order of M. Hence, it is sufficient to test for the (k,δ)-C1P of only the submatrix M' corresponding to each active region (of size b), and not all of M.

The algorithm uses the following two data structures:

(1) A (fifo) queue Q whose elements are active regions.

(2) An array A which contains one element for each possible active region. Each element $A[r]$ of A consists of a Boolean flag $A[r]$`.examined`, telling whether the active region r has already been considered in the search, and a list $A[r]$`.unplaced` of vertices which is intended to list all vertices *not* in the domain of each plausible partial layout with active region r. Here, we extend each element $A[r]$ so that it also contains a $n \times b$ (sub-) matrix $A[r].M'$ which stores the submatrix of M that corresponds (in this order) to the columns

$f^{-1}(\max(p-b+1,1),\ldots,f^{-1}(p))$ of active region r.

At the start of the algorithm, Q is initialized to contain the single element representing the active region (henceforth denoted Φ) of the empty partial layout $\emptyset$. The flag $A[\Phi]$`.examined` is set to *true* and $A[\Phi]$`.unplaced` is initialized to list all the elements of V. The remaining $A[r]$`.examined` are initially *false*, and the remaining $A[r]$`.unplaced` are uninitialized. Each $A[r].M'$ is also set to the matrix on zero columns (the empty matrix). The algorithm now proceeds as follows:

Algorithm 1 Algorithm of Saxe [135] for testing the bandwidth of a graph.

```
1. Extract an active region $r$ from the head of $Q$.
2. From $A[r]$.unplaced, determine the successors of $r$. To determine if $s$, the
   active region obtained by extending $r$ with some $c \in A[r]$.unplaced is a suc-
   cessor, we first check (as in done in Saxe [135]) to see if $s$ is a the active region
   of a plausible partial layout of $G_M$. In addition, we compute $A[s].M'$ by adding
   column $c$ to the end of $A[r].M'$. This new active region $s$ can only be a successor
   if $A[s].M'$ satisfies properties (3) and (4) that extend Definition 23.
3. for each successor $s$ of $r$ such that $A[s]$.examined is false, perform the fol-
   lowing steps:
   a. Set $A[s]$.examined to true.
   b. Compute $A[s]$.unplaced by deleting the last vertex of $s$ from
      $A[r]$.unplaced.
   c. If $A[s]$.unplaced is the empty set, then halt, asserting that
      bandwidth$(G) \leq b$.
   d. Insert $s$ at the end of $Q$.
4. If $Q$ is empty, then halt, asserting that bandwidth$(G) > b$. Otherwise, go to Step
   1.
```

We now analyze the time and space complexity of the algorithm. Since there are $O(n^b)$ active regions r, and with each r we associate the $O(n)$ elements $A[r]$`.unplaced`, the $n \times b$ matrix $A[r].M'$, which is of size $O(mn)$, and some constant size flags, the space required by the algorithm is $O(mn^{b+2})$. For the running time, we first note that (as in Saxe's algorithm) that Steps 1 through 4 will be executed $O(n^b)$ times. Again, each individual execution of Steps 1 and 4 take constant time, so their contribution to the total running time is $O(n^b)$. In each execution of Step 2, $A[r]$`.unplaced` can have $O(n)$ elements, and the test that $A[s].M'$ for each potential successor s in $A[r]$`.unplaced` satisfies properties (3) and (4) that extend Definition 23 takes time $O(mn)$, since $A[s].M'$ is of size $O(mn)$. Hence Step 2 contributes $O(mn^{b+2})$ to the total execution time. In each execution of Step 3, again Steps 3.a through 3.d maybe executed as many as n times, and that Step 3.b takes time $O(n)$, Step 3 takes time $O(n^{b+2})$ (which is already less than that of Step 2, so we leave out the analysis of Saxe [135] for bringing down this upper bound).

We hence have our version of the following theorem from Saxe [135] for deciding the bandwidth of a binary matrix.

Theorem 28. *Let b be any positive integer. Then, given some binary matrix M, there is an algorithm which decides if bandwidth$(M) \leq b$ using time and space $O(mn^{b+2})$.*

Proof. To test the bandwidth of M, we first perform a time $O(n)$ depth-first search which either

(1) determines that G_M has some vertex of degree greater than $2b$, or

(2) partitions G_M into connected components, none of which have any vertex of degree greater than $2b$.

In case (1), we know immediately that bandwidth$(M) > k$. In case (2), we apply Algorithm 1 to the submatrices of M that correspond to the connected components of G_M. □

By the above Theorem 28 and Property 17, we have the following theorem which gives us the result.

Theorem 29. *Let M be an $m \times n$ binary matrix such that every row has at most d entries* 1*. Deciding if M has the (k,δ)-C1P can be done in time and space $O(mn^{d+(k-1)\delta+1})$.*

3.2 The (d,k,∞)-C1P

Here, we show that deciding the (d,k,∞)-C1P for every $d > k \geq 2$ is NP-complete. This proof is broken down into the following subsections. In Subsection 3.2.1 we define first a hypergraph covering problem that will be used later to show NP-completeness of this case. In Subsection 3.2.2 we then show that a special case of this covering problem for 3-uniform hypergraphs is NP-complete. In Subsection 3.2.3 we use the NP-completeness construction of Subsection 3.2.2 to show that this covering problem defined in Subsection 3.2.1 is NP-complete in general. Finally, in Subsection 3.2.4, we give a correspondence of this covering problem of Subsection 3.2.1 to the problem of deciding the (d,k,∞)-C1P for the result of this section.

3.2.1 A Hypergraph Covering Problem

We first define the following hypergraph covering problem. In the sections that follow, we will show that this problem is NP-complete, and that it corresponds exactly to the problem of deciding the (d,k,∞)-C1P for the hardness result of this chapter. Note that a hypergraph $H = (V,E)$ is d-*uniform* when all its hyperedges are d-*edges*, that is, hyperedges that contain exactly d vertices.

Definition 30 (p-Covering of a d-Uniform Hypergraph)**.** *Given a d-uniform hypergraph $H = (V,E)$ and an integer p, let $K_{|V|}$ be a complete graph on V and let $\mathscr{P}_p$ be the set of all subsets of $E(K_{|V|})$ with exactly p edges. A p-*covering *of H is a graph $G = (V,E')$ such that there exists a map $\mathbf{c} : E \to \mathscr{P}_p$ such that*

(a) for every $h \in E$, and for every $e \in \mathbf{c}(h)$, $e \subseteq h$; and

(b) $E' = \bigcup_{h \in E} \mathbf{c}(h)$.

*Here, we say that set $\mathbf{c}(h)$ p-*covers *the hyperedge h and that G p-*covers *H.*

Informally, a p-covering of a d-uniform hypergraph is a graph constructed by picking p edges from each hyperedge.

Problem 31 (d-Uniform Hypergraph p-Covering by Paths (d-UH-p-CP))**.** *Given a d-uniform hypergraph $H = (V,E)$ and an integer $p < d$, is there a p-covering of H which consists only of disjoint paths?*

Variations of this problem were defined in previous works [59–61]. The first variation allowed the hypergraph to have only 2, 3 and 4-edges, where 2- and 3-edges were covered by picking one edge, while 4-edges were covered by two parallel edges, and required that the covering contains only disjoint edges and vertices. This variation was shown to be polynomial-time solvable which provided an algorithm for a special version of haplotyping problem via galled-tree networks [59]. The second variation allowed only 3-uniform hypergraphs, and required all connected components of the covering to be paths of length at most 3. This variation was shown to be NP-complete [61]. A slightly more complex version of this was then used to show that in general the haplotyping problem via galled-tree networks is NP-complete [60].

In the next section, we show that a special case of this problem, namely the 3-UH-1-CP Problem, is NP-complete, which is then generalized in Section 3.2.3 to show NP-completeness of the d-UH-p-CP Problem for every $d-2 \geq p \geq 1$.

3.2.2 The 3-Uniform Hypergraph 1-Covering by Paths Problem

We now show that the 3-Uniform Hypergraph 1-Covering by Paths (3-UH-1-CP) Problem is NP-complete.

Theorem 32. *The* 3-*UH*-1-*CP Problem is NP-complete.*

Proof. Clearly, the problem is in NP. We will show it is also NP-hard by reduction from 3SAT(3), a restricted version of 3SAT, proved NP-complete by Papadimitriou [120], in which every variable has exactly two positive and one negative occurrence in the clauses.[1] We will call a p-covering of a hypergraph *valid* if it consists only of disjoint paths. Note that a valid p-covering does not contain vertices of degree 3 or more and does not contain cycles. Given 3SAT(3) formula ϕ with variables $X = \{x_1, \ldots, x_n\}$ and clauses $C = \{c_1, \ldots, c_m\}$, we now construct a 3-uniform hypergraph H_ϕ on at most $12n + 15m$ hyperedges which contains, among other vertices, a vertex for each literal of ϕ (there are $3n$ such vertices) that has a valid 1-covering if and only if ϕ is satisfiable.

[1]We remark that the exact formulation of 3SAT(3) in Papadimitriou [120] allows also variables with one positive and two negated occurrences, but these can easily be converted to the other type of variables by replacing them with their negations in all clauses. Clearly, this does not affect the complexity of the problem.

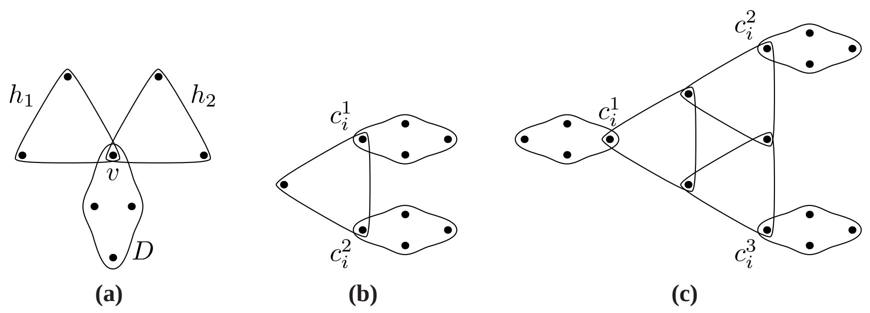


Figure 3.1: (a) A simple dependency on 1-coverings of two touching hyperedges enforced by a copy of D (depicted as a diamond). (b) The 2-clause and (c) 3-clause gadgets for clause c_i.

First we give an important building block that is used throughout this construction: the complete 3-uniform hypergraph D on 4 vertices. In any valid 1-covering of D, there is no isolated vertex. Indeed, assume for contradiction that v is the isolated vertex in a valid covering G of D. Let u_1, u_2, u_3 be the remaining three vertices. Then there is a pair u_i, u_j such that $\{u_i, u_j\}$ is not an edge in G. However, no edge is 1-covering hyperedge $\{v, u_i, u_j\}$, a contradiction. We will use several copies of D in the construction to introduce a dependency on 1-coverings of touching hyperedges and depict them as diamonds in the figures. For instance, consider the hypergraph in Figure 3.1a. Since in any valid 1-covering G of this hypergraph, v is a member of an edge in D, at most one of the hyperedges h_1 and h_2 can "pick" an edge involving v, otherwise vertex v would have degree 3 or more.

Now to the main construction. Consider the instance ϕ of 3SAT(3) with variables $X = \{x_1, \dots, x_n\}$ and clauses $C = \{c_1, \dots, c_m\}$. In the construction, any valid covering selects a set of literals (more precisely, the vertices corresponding to these literals), i.e., positive and negative occurrences of variables. If this selection satisfies the following two properties:

(1) every clause selects at least one literal, and

(2) for every $x \in X$, at most one of x and $\neg x$ is selected,

then this selection can be used to build a satisfying truth assignment for ϕ as follows: for every $i \in \{1, \dots, n\}$ if x_i (resp., $\neg x_i$) is in the selection, set the value of

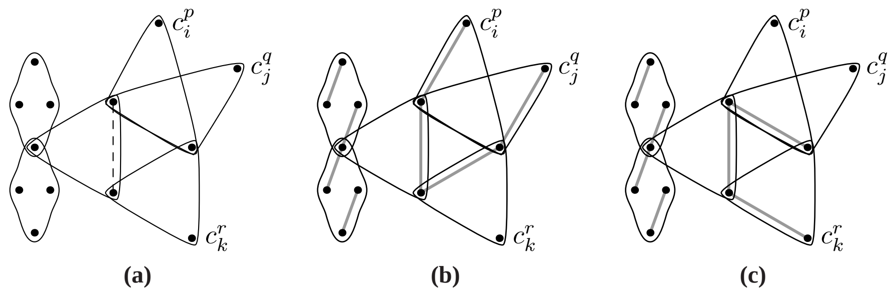


Figure 3.2: (a) The variable gadget for variable with positive occurrences c_i^p and c_j^q and negated occurrence c_k^r in the clauses. The dashed edge is always picked in any valid 1-covering. (b) Grey edges are picked when this variable is set to *false* in a satisfying assignment of ϕ. (c) Grey edges are picked when the variable is set to *true*.

x_i to *true* (resp., *false*). If neither x_i and or $\neg x_i$ is in the selection, pick the value at random. We design a hypergraph H_ϕ composed of clause gadgets which will guarantee the first condition and variable gadgets which will ensure the second condition.

Figures 3.1b and 3.1c depict the 2-clause and 3-clause gadgets, respectively. Given a valid 1-covering G of the clause gadget for clause c_i with literals c_i^1, c_i^2 (and c_i^3 for a 3-clause), we say that a literal vertex c_i^j is *selected* in G, if c_i^j is contained in two edges of the covering G. Note that in both clause gadgets at least one of the literal vertices is selected in any valid covering. This is obvious for the 2-clause gadget. For the 3-clause gadget, if none of the literal vertices is selected in a valid 1-covering of this gadget, then in the three hyperedges in Figure 3.1c, no picked edge involves c_i^1, c_i^2 or c_i^3. But this creates a cycle, a contradiction. Now, each literal vertex c_i^j will also appear in exactly one variable gadget described in the next paragraph. If a literal vertex c_i^j is selected in a valid covering then it cannot be contained in any edge that covers the hyperedges of the variable gadget, otherwise c_i^j has degree 3 or more in this covering. The variable gadget for each $x \in X$ will use this property to ensure that literal vertices x and $\neg x$ are not selected at the same time.

Figure 3.2a depicts the variable gadget for variable $x \in X$ with the two positive occurrences c_i^p and c_j^q, and one negated occurrence c_k^r of this variable x in the clauses. Note that if both a positive and the negated literal vertices of x are selected by a clause gadget in a valid 1-covering of H_ϕ, then it forces a cycle in the variable gadget of x, a contradiction. It follows that if H_ϕ has a valid 1-covering then ϕ is satisfiable.

Conversely, if ϕ has a satisfying assignment τ, let us pick one literal for each clause which makes it satisfied in τ and build the 1-covering of H_ϕ as follows. In each clause gadget, (i) in each hyperedge of this clause gadget that contains a literal vertex, pick an edge containing the literal vertex if this literal was selected for this corresponding clause, and (ii) for each diamond, choose any of the 3 valid 1-coverings of this diamond that consist of 2 parallel edges. In the variable gadgets, pick the edges as depicted in Figure 3.2b if the variable has value *false* in τ and otherwise, pick the edges as depicted in Figure 3.2c. By selecting edges in this fashion, every hyperedge of H_ϕ is 1-covered by an edge, and each literal vertex is adjacent to at most two edges in the 1-covering, one of them lying in the diamond. Hence, there is no vertex of degree 3 and no cycles in this 1-covering, i.e., this 1-covering is valid.

Since the number of hyperedges used in the construction is at most $12n+15m$, i.e., linear in the size of ϕ, this construction can be built in polynomial-time, and hence, the 3-UH-1-CP Problem is NP-hard. □

In the following section, we generalize this construction to show that for every $d-2 \geq p \geq 1$, the d-UH-p-CP Problem is NP-complete.

3.2.3 The d-Uniform Hypergraph p-Covering by Paths Problem

We now show how the construction of Section 3.2.2 can be generalized to show that for every $d-2 \geq p \geq 1$, the d-Uniform Hypergraph p-Covering by Paths (d-UH-p-CP) Problem is NP-complete. The main building block in this new construction is the following d-uniform hypergraph that generalizes the hypergraph D (the diamond) from the previous construction of Section 3.2.2.

Lemma 33. *For any $d-2 \geq p \geq 1$, there exists a d-uniform hypergraph $D_{d,p} = (V,E)$ with a distinguished vertex $v \in V$ that has the following properties:*

1. $|V| = 2d-p-1$ *and* $|E| = \binom{2(d-p)-1}{d-p}$;

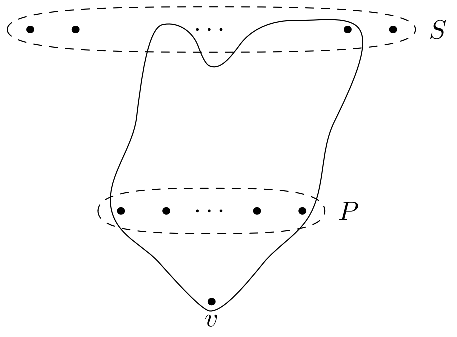


Figure 3.3: Hypergraph $D_{d,p}$: only one of the $\binom{|S|}{d-p}$ hyperedges is shown.

2. *in any valid p-covering of $D_{d,p}$, v is not isolated; and*

3. *hypergraph $D_{d,p}$ has a valid p-covering in which v has degree 1.*

Proof. Let $D_{d,p} = (V,E)$ be the d-uniform hypergraph on the vertex set $V = S \cup P \cup \{v\}$ where $|S| = 2(d-p)-1$, $|P| = p-1$, and v is the single distinguished vertex. For every subset $S' \subseteq S$ of size $d-p$, we add a hyperedge on the d vertices $S' \cup P \cup \{v\}$ to E, i.e., $|E| = \binom{2(d-p)-1}{d-p}$. Hypergraph $D_{d,p}$ is depicted in Figure 3.3. We now show that this hypergraph satisfies conditions 2 and 3 of the lemma. Here, again, we call a graph p-covering of a hypergraph *valid* if it consists only of disjoint paths.

Assume, for contradiction, that v is isolated in a valid p-covering G of $D_{d,p}$. Since G is some collection of paths on the vertex set $S \cup P$, *virtual* edges can always be added to G to extend this collection to a single path G' on this set. In what follows, we will find a hyperedge in $D_{d,p}$ and show that it contains less than p edges in G', and hence, less than p edges in G, and thus, is not covered by G.

Path G' defines a total order on its vertex set $S \cup P$ (there are two such total orders, but we can choose either one, without loss of generality). Let $t : S \cup P \to \mathbb{N}$ be this total order. If we follow the vertices of path G' according to t, it starts at some vertex in one of S or P, alternates between the two sets, and then terminates in one of these sets. Hence, the subgraph G'_S (resp., G'_P) of G' induced on vertex set S (resp., P) is some collection of paths on S (resp., P), say $S_1, \ldots, S_r$ (resp., $P_1, \ldots, P_\ell$), where for any $i < j$, vertex $u \in S_i$ (resp., P_i) and $u' \in S_j$ (resp., P_j),

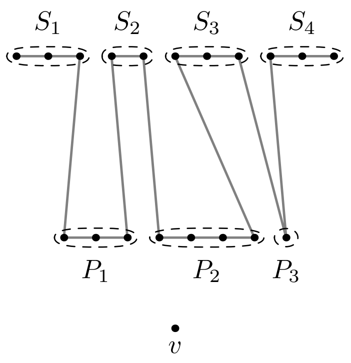


Figure 3.4: The path G' through vertex set $S \cup P$ that alternates between subpaths completely in S and completely in P. Some of the shown edges may be virtual.

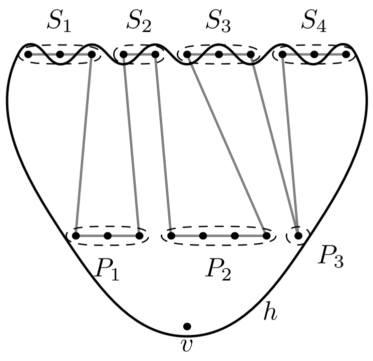


Figure 3.5: Hyperedge h of $D_{d,p}$ which contains less than p edges from G' depicted in Figure 3.4.

$t(u) < t(u')$, cf. Figure 3.4.

Let us order the elements of S according to total order t and let S' be the odd numbered elements of S according to this order. Since $|S| = 2(d-p)-1$, $|S'| = d-p$. Now, consider the d-edge $h = S' \cup P \cup \{v\}$ of $D_{d,p}$. Hyperedge h is indeed an edge in $D_{d,p}$ since it contains $P \cup \{v\}$ and a subset, namely S', of size $d-p$ of S. Hyperedge h for the example of Figure 3.4 is depicted in Figure 3.5. We will show that this hyperedge contains less than p edges from G'.

Let us count the number of edges of G' that are contained in h. Each path P_i, $i = 1, \ldots, \ell$ is completely contained in h, and thus contributes to h the $|P_i| - 1$ edges that connect the vertices of this path. On the other hand, since S' is the set

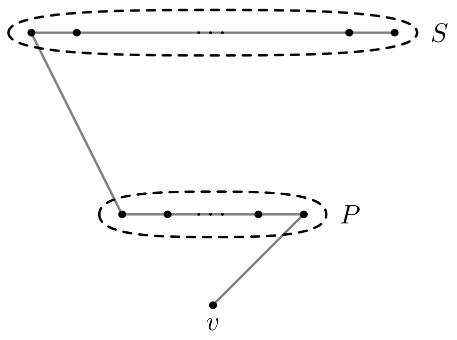


Figure 3.6: A valid p-covering of $D_{d,p}$ in which vertex v has degree 1.

of odd numbered elements of S according to total order t, none of the edges in S_j, $j = 1, \ldots, r$ is contained in h. Finally, we need to consider edges of the path G' crossing between the sets S and P. We will show that for each $i = 1, \ldots, \ell$, there is at most one crossing edge starting at a vertex of P_i and ending in S that is contained in h. There are at most two edges starting at a vertex of P_i and ending in some vertex of S. If the number of these edges is less than two, the claim holds. Assume there are two such edges. They must start at the endpoints of P_i and end in the consecutive elements of S (according to t). Hence, at most one of them is ending in the odd numbered element of S, i.e., contained in h. It follows that the number of crossing edges contained in h is at most ℓ. Hence, h contains at most $\ell + \sum_{i=1}^{\ell}(|P_i| - 1) = |P| = p - 1$ edges of G', and hence, at most $p - 1$ edges of G, thus it is not p-covered by G, a contradiction. We can conclude that in any valid p-covering of $D_{d,p}$, vertex v has degree at least one.

Finally, we show that $D_{d,p}$ has a valid p-covering in which vertex v has degree 1. Consider the path G that starts at v and then visits all vertices in P and then all vertices in S, cf. Figure 3.6. Consider any hyperedge $h = S' \cup P \cup \{v\}$ where S' is some subset of S of size $d - p$. The hyperedge h contains $P \cup \{v\}$, and thus the subpath of G induced by these vertices. This subpath has $p - 1$ edges. Consider the subgraph of G induced by S'. If this subgraph contains at least one edge, we pick this edge for h, and hence, h is p-covered by G. Otherwise, S' must consist only of odd numbered elements of the subpath of G induced by S, and thus it contains the first vertex of this subpath. Hence, h contains the edge of G connecting sets S and P, and we pick this edge for h, i.e., it is p-covered by G. $\square$

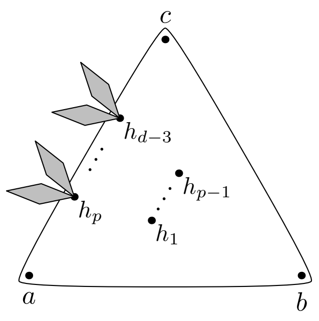


Figure 3.7: Vertices and hyperedges added to $\bar{H}$ to simulate the 3-edge $h = \{a,b,c\}$. The grayed diamonds depict copies of $D_{d,p}$.

In the following theorem we will use many copies of $D_{d,p}$ to simulate the behavior of a 3-edge in the 1-covering problem with a d-edge in the p-covering problem.

Theorem 34. *For every $d-2 \geq p \geq 1$, the d-UH-p-CP Problem is NP-complete.*

Proof. Clearly, this problem is in NP. We will show that it is also NP-hard by reduction from the 3-UH-1-CP Problem that was shown to be NP-complete in Section 3.2.2.

Given a 3-uniform hypergraph $H = (V,E)$, we will construct a d-uniform hypergraph $\bar{H}$ that has a valid p-covering if and only if H has a valid 1-covering. For each 3-edge $h = \{a,b,c\} \in E$ we add the corresponding d-edge $\bar{h} = \{a,b,c,h_1,\ldots,h_{d-3}\}$ to $\bar{H}$. To simulate in $\bar{H}$, the behavior of h, we then add $2(d-p-2)$ copies of $D_{d,p}$ to $\bar{H}$, where the distinguished vertex v of each copy is identified with one of the vertices $h_p,\ldots,h_{d-3}$ such that each of them is used exactly twice. Figure 3.7 illustrates all vertices and hyperedges added to $\bar{H}$ for this 3-edge h in H. We note that all vertices other than a,b,c added to $\bar{H}$ for h are disjoint from all other vertices.

Now, assume that there is a valid p-covering $\bar{G}$ of $\bar{H}$. We will construct a 1-covering G of H as follows. For each $h \in E$, consider the subgraph $\bar{G}_{\bar{h}}$ of $\bar{G}$ induced by the vertices in $\bar{h}$. It must have at least p edges. By Lemma 33, vertices $h_p,\ldots,h_{d-3}$ are incident to some edges of $\bar{G}$ in two different diamonds, and since there is no vertex of degree 3 in $\bar{G}$, they are isolated vertices in $\bar{G}_h$. Hence, we have p edges in the $p+2$ element set $\{a,b,c,h_1,\ldots,h_{p-1}\}$ which cannot create a cycle. It hence follows that these vertices must form at most two components. Therefore, at least one pair of the vertices a,b,c must lie in the same component. If there is only one such pair, we add it to G as an edge. If all three vertices a,b,c are connected, we add to G a pair which remains connected after removing the third vertex. As a consequence of this choice, each edge $\{u,v\}$ in G covering a hyperedge h in H corresponds to a path in $\overline{G}$ connecting u and v. In addition, all internal vertices of these paths are not in V, and since hyperedges in $\overline{H}$ share only vertices in V, they are pairwise internally vertex disjoint.

The graph G constructed above is obviously a 1-covering of H. Let us check that it is also valid. First, if there is a vertex $u \in V$ with degree 3 or more, then there are three internally disjoint paths starting at u in $\overline{G}$, i.e., u would have degree at least 3 in $\overline{G}$, a contradiction. Second, if there is a cycle $u_1,u_2,\ldots,u_k,u_1$ in G, then for each edge $\{u_i,u_{i+1}\}$ in G, we have a path connecting u_i and u_{i+1} in $\overline{G}$. Since these paths are internally vertex disjoint, they create a cycle in $\overline{G}$, a contradiction.

Conversely, assume there is a 1-covering G of H. We construct a p-covering $\overline{G}$ of $\overline{H}$ as follows. Cover each copy of $D_{d,p}$ such that the distinguished vertex has degree 1 (this is possible by Lemma 33). For each hyperedge $h = \{a,b,c\} \in E$, without loss of generality, let $\{a,b\}$ be the edge that covers h in G. Then cover hyperedge $\overline{h}$ by a path starting at a, visiting all vertices $h_1,\ldots,h_{p-1}$ and ending at b, while the vertex c is an isolated vertex. This is a p-covering $\overline{H}$ and it is easy to verify that it is also valid.

Finally, let us check that the construction is polynomial. The number of vertices of $\overline{H}$ is $|V|+|E|[d-3+2(d-p-2)(2d-p-2)]$ and the number of edges is $|E|\left[1+2(d-p-2)\binom{2(d-p)-1}{d-p}\right]$. Since d and p are assumed to be constants, the reduction is polynomial. □

3.2.4 The Complexity of Deciding the (d,k,∞)-C1P

We now show that for every $d > k \geq 2$, deciding the (d,k,∞)-C1P is NP-complete, by showing the correspondence of this problem to the d-UH-$(d-k)$-CP Problem. A d-uniform hypergraph $H = (V,E)$ can be represented as a binary matrix B_H with $|V|$ columns and $|E|$ rows, where for each hyperedge $h \in E$, we add a row with `1`'s in the columns corresponding to vertices in h and `0`'s everywhere else. Obviously, the degree of every row of B_H is d and there is a one-to-one correspondence between d-uniform hypergraphs and such matrices.

Lemma 35. *A d-uniform hypergraph $H = (V,E)$ can be $(d-k)$-covered by disjoint paths if and only if matrix B_H has the (d,k,∞)-C1P.*

Proof. Assume first that H has a valid covering G. Since G consists of disjoint paths, there is a Hamiltonian path P on V containing all edges of G. This path defines an order on the vertices in V. Consider the order of the columns of matrix B_H based on this order (V is the set of columns B_H). We will show that this order is (d,k,∞)-consecutive. Since each row of B_H contains exactly d `1`'s, it is enough to show that $d-k$ pairs of these d columns are adjacent in this order. The d columns containing `1`'s in each row form a hyperedge in H. Since G is a valid $(d-k)$-covering, there are edges between $d-k$ pairs of these d columns in G. Since P contains all edges of G, it contains also these $d-k$ edges and hence, each of the corresponding $d-k$ pairs of columns are adjacent in the order. It follows that the order of B_H is (d,k,∞)-consecutive.

Conversely, assume that matrix B_H is (d,k,∞)-consecutive. Let $\pi = v_{i_1},\dots,v_{i_n}$ be the order of the columns in a (d,k,∞)-consecutive order of B_H. Now, for any hyperedge $h = \{v_{j_1}, v_{j_2}, \dots, v_{j_d}\}$ of H, there is a row in B_H with `1`'s in these d columns, hence, $d-k$ pairs of the columns in h must be adjacent in the order π. Consider the following covering G of H: for every hyperedge pick the edge between each pair of adjacent columns/vertices. Note that every edge in G is $\{v_{i_j}, v_{i_{j+1}}\}$ for some j. Hence, G has no vertex of degree 3 or higher, nor any cycle, thus G is a collection of disjoint paths, i.e., a valid $(d-k)$ covering of H. □

By Theorem 34 and Lemma 35 it follows that for every $d > k \geq 2$, deciding the (d,k,∞)-C1P is NP-complete.

Theorem 36. *For every $d > k \geq 2$, deciding the (d,k,∞)-C1P is NP-complete.*

We remark that the (d,k,∞)-C1P is the k-C1P [55] for matrices of bounded degree d. Goldberg et al. [55] posed the open question of the complexity of deciding the 2-C1P for matrices with a limit ℓ on the number of ones per row, *and per column*. This is motivated by a typical setting in physical mapping, where a clone will only contain a small number of probes, and there is only limited coverage of the entire sequence by the clones (cf. Chapter 1 for details on physical mapping). Since our construction of Theorem 32 in Subsection 3.2.2 which implies that deciding the 2-C1P for matrices of bounded degree 3 is NP-complete uses also only 7 ones per column, we have the following corollary which closes this open question of Goldberg et al. [55].

Corollary 37. *Deciding the 2-C1P with a limit 3 on the number of ones per row and 7 on the number of ones per column is NP-complete.*

Chapter 4

The Consecutive-Ones Property with Multiplicity

In this chapter we show in Section 4.1 that deciding the *m*C1P is NP-complete for matrices with degree at most 3 and $\mathbf{m}(s) \leq 2$ for each $s \in S$, where S is the set of columns of M. We then present in Section 4.2 the two restricted variants of the *m*C1P given in Wittler and Stoye [151], namely the Consecutive-Ones Property with Multiplicity for Framed Rows (*m*C1P(fr)) and the Consecutive-Ones Property with Multiplicity for Nested Rows (*m*C1P(ne)). In Subsection 4.2.1 (resp., Subsection 4.2.2) we detail the *m*C1P(fr) (resp., *m*C1P(ne)) variant, its biological motivation, and show that deciding the *m*C1P(fr) (resp., *m*C1P(ne)) is NP-complete for matrices with degree at most 6 (resp., 3) and $\mathbf{m}(s) \leq 2$ for each $s \in S$. Then, in Section 4.3 we give a tractability result for a case of the *m*C1P, motivated by handling ancestral telomeres in the reconstruction of AGO.

4.1 The Consecutive-Ones Property with Multiplicity (*m*C1P)

Here, we show that deciding the *m*C1P is NP-complete for matrices with degree at most 3 and $\mathbf{m}(s) \leq 2$ for each $s \in S$, where S is the set of columns of M.

Theorem 38. *Given a degree 3 matrix M on set S of columns, deciding the mC1P for M is NP-complete for a multiplicity vector $\mathbf{m}$ where $\mathbf{m}(s) \leq 2$ for each $s \in S$.*

Before giving the proof, we would like to emphasize that this is the strongest possible result. If the maximum multiplicity would be one, this is just an instance of the classical C1P. If the degree of M is restricted to 2, then this corresponds to the model of adjacencies, and can hence by solved using the method based on Eulerian graphs given in Wittler and Stoye [151].

Proof. One can easily formulate an algorithm that verifies a given solution, i.e., a C1 order with multiplicity in polynomial time, which shows that the problem belongs to the complexity class NP. We will show NP-hardness of deciding the mC1P by reduction from 3SAT(3), which has been proven to be NP-complete by Papadimitriou [120]. 3SAT(3) is a restricted version of 3SAT in which every variable has exactly two positive and one negative occurrence in the clauses.[1]

Here, we again reduce from a type of hypergraph covering problem as we did in Chapter 3 to show NP-hardness of deciding the (d,k,δ)-C1P. Given a 3SAT(3) formula ϕ with variables $X = \{x_1,\ldots,x_n\}$ and clauses $C = \{c_1,\ldots,c_m\}$, we construct a matrix M_ϕ consisting of at most $5n+2m$ columns of multiplicity at most two and at most $5n+3m$ rows of degree three or less for which a C1 order σ with multiplicity exists if and only if ϕ is satisfiable.

For this instance ϕ of 3SAT(3), we say that a clause *selects* one of its literals in a truth assignment of ϕ if this literal has value *true* in this assignment. Obviously, a truth assignment of ϕ is a satisfying truth assignment if and only if every clause selects at least one literal and for every $x \in X$, at most one of x and $\neg x$ is selected. We design an instance M_ϕ composed of clause gadgets which will guarantee the first condition and variable gadgets which will ensure the second condition.

For each 2-clause c_i with literals c_i^1 and c_i^2, we add to M_ϕ the two columns c_i^1 and c_i^2, each of multiplicity 2, and the two columns c_i^* and c_i^{**}, each of multiplicity 1, and the rows $S_i^1 = [c_i^1, c_i^2, c_i^*]$ and $S_i^2 = [c_i^*, c_i^{**}]$. This is referred to as the *2-clause gadget*. For each 3-clause c_i with literals c_i^1, c_i^2 and c_i^3, we add to M_ϕ the three columns c_i^1, c_i^2 and c_i^3, each with multiplicity 2, and the row $S_i = [c_i^1, c_i^2, c_i^3]$. This is referred to as the *3-clause gadget*.

[1]We remark that the exact formulation of 3SAT(3) in Papadimitriou [120] allows also variables with one negated and two positive occurrences, but these can easily be converted to the other type of variables by replacing them with their negations in all clauses. Clearly, this does not affect the complexity of the problem.

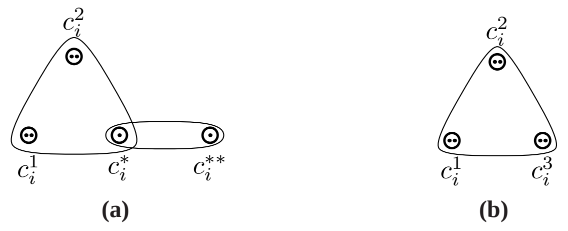


Figure 4.1: Graphical representations of the (a) 2-clause gadget and (b) 3-clause gadget for clause c_i. The multiplicity of the columns (resp., vertices) is indicated by the number of dots. Rows are depicted by ellipses surrounding two vertices or triangles surrounding three vertices, respectively.

Figure 4.1 shows graphical representations of these gadgets, which also highlights that M_ϕ can be viewed as a hypergraph with a vertex for each column and a hyperedge for each row. A C1 order with multiplicity of M_ϕ is then a collection of walks on this hypergraph that *covers* each hyperedge (for each hyperedge e there is a connected subwalk containing all vertices in e) such that no vertex v is visited more than $\mathbf{m}(v)$ times.

We say that in string σ, a clause gadget *selects* a literal column c_i^j, if, in σ, c_i^j is enclosed on both the left and right side by at least one column of this gadget. Note that in both clause gadgets, at least one of the literal columns is selected in any valid string σ. For the 2-clause gadget, string σ has to contain one of the substrings $c_i^1 c_i^2 c_i^* c_i^{**}$ or $c_i^2 c_i^1 c_i^* c_i^{**}$, or one of their reversals. For the 3-clause gadget, string σ has to contain one of the substrings $c_i c_i^2 c_i^3$ or $c_i^2 c_i^1 c_i^3$, or $c_i^1 c_i^3 c_i^2$ or one of their reversals. Clearly a literal column is always selected in each of these gadgets for any string σ that is a C1 order with multiplicity of M_ϕ.

Now, all $3n$ literal columns c_i^j from the set of clause gadgets for C will appear in the variable gadgets, where the variable gadget *selects* this column c_i^j, if c_i^j is again enclosed on both the left and the right side by at least one column of the gadget in σ. So if a literal column c_i^j is selected by a clause gadget, then it cannot be selected by this variable gadget, since σ is a string and thus c_i^j can be framed by at most two other columns of σ. The variable gadget for each $x \in X$ will use this property to ensure that literal vertices x and $\neg x$ are not selected at the same time.

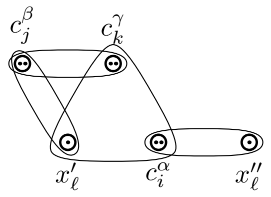


Figure 4.2: Graphical representation of the variable gadget for variable x_ℓ with positive occurrences c_i^α and c_j^β and negated occurrence c_k^γ in the clauses.

For each variable x_ℓ with the two positive occurrences c_i^α and c_j^β and the negated occurrence c_k^γ, we already added to M_ϕ the columns c_i^α, c_j^β and c_k^γ, each of multiplicity two in the corresponding clause gadgets for the clauses containing x_ℓ and $\neg x_\ell$. We further add to M_ϕ the two columns x'_ℓ and x''_ℓ, each of multiplicity one, and the four rows $P_\ell^1 = [c_i^\alpha, c_k^\gamma, x'_\ell]$, $P_\ell^2 = [x'_\ell, c_j^\beta]$, $P_\ell^3 = [c_j^\beta, c_k^\gamma]$, $P_\ell^4 = [c_i^\alpha, x''_\ell]$. This is referred to as the *variable gadget* for x_ℓ, depicted in Figure 4.2.

Now, consider a C1 order σ with multiplicity for M_ϕ where the literal c_k^γ is selected by some clause gadget. Since one copy of c_k^γ is used up by this clause gadget, σ must contain the substring $c_j^\beta c_k^\gamma c_i^\alpha x'_\ell$ or its reversal because it is the only way to ensure consistency in σ for rows P_ℓ^1 and P_ℓ^3 in M_ϕ with the one remaining copy of c_k^γ. If literal c_j^β is also selected by some clause gadget, then there is no way that σ can be consistent with P_ℓ^2 in σ. While if literal c_i^α is also selected by some clause gadget, then there is no way that σ can be consistent with P_ℓ^4, which contradicts the fact that σ is a C1 order with multiplicity for M_ϕ. It follows that if M_ϕ has a C1 order σ with multiplicity, then ϕ is satisfiable.

We now show that the converse holds, namely if ϕ has a satisfying truth assignment τ, then M_ϕ has a C1 order σ with multiplicity. Given τ, we construct σ as follows. For any variable x_ℓ with the two positive occurrences c_i^α and c_j^β and the negative occurrence c_k^γ, in τ, either of the two cases must hold:

1. c_i^α and c_j^β are *false* and c_k^γ is *true*: In this case, we create the substring $c_k^\mu c_k^\gamma c_k^\nu$, satisfying S_k, or $c_k^\mu c_k^\gamma c_k^* c_k^{**}$ satisfying S_k^1 and S_k^2, depending on whether c_k^γ is part of a 3- or a 2-clause. Further, we create the substrings $c_j^\beta c_k^\gamma c_i^\alpha x'_\ell c_j^\beta$ and $c_i^\alpha x''_\ell$, fulfilling all $P_\ell^{1,2,3,4}$.

2. c_i^α and c_j^β are *true* and c_k^γ is *false*: In this case, we create the substrings $c_i^\mu c_i^\alpha c_i^\nu$, satisfying S_i (resp., $c_i^\mu c_i^\alpha c_i^*, c_i^{**}$ satisfying S_i^1 and S_i^2) and $c_j^\mu c_j^\beta c_j^\nu$, satisfying S_j (resp., $c_j^\mu, c_j^\beta, c_j^* c_j^{**}$ satisfying S_j^1 and S_j^2). Further, we create the substring $x_\ell'' c_i^\alpha c_k^\gamma x_\ell' c_j^\beta c_k^\gamma$, fulfilling all $P_\ell^{1,2,3,4}$.

The requirements for σ imposed by all given rows of M_ϕ fulfilled. It remains to be shown that the multiplicity constraint is met as well. None of the columns c_k^γ, (resp., c_k^*, c_k^{**}), x_ℓ' and x_ℓ'' used in the first case are affected by the second case for any other variable and none of the columns c_i^α, (resp., c_i^*, c_i^{**}), c_j^β, (resp., c_j^*, c_j^{**}), x_ℓ' and x_ℓ'' are affected by the first case for any other variable. For all of these columns, the multiplicity constraint is met. The column c_i^α is used twice in case one. The same column will occur as c_i^μ or c_i^ν in the second case for some other variable. But since in both of the corresponding substrings, the column is the first or last element, they can be merged into one substring using only two copies of c_i^α. Here we might have to reverse the substring $c_j^\alpha x_\ell''$ to $x_\ell'' c_j^\alpha$, still fulfilling P_ℓ^4. The same argument holds for c_j^β.

Analogously, the column c_k^γ is already used twice in case two. The same column will occur as c_k^μ or c_k^ν in the first case for some other variable. But since in both of the corresponding substrings, the column is the first or last element, they can be merged to one substring using only two copies of c_k^γ. Here we might have to reverse one of the substrings, still fulfilling the restrictions by the rows of M_ϕ.

Since each column only occurs in one row S_i (resp., S_i^1) of M_ϕ, each substring induced by rows $P_\ell^{1,2,3,4}$ has to be merged on one side, i.e., no cycles are created in the set of walks covering the corresponding hypergraph. Eventually, any concatenation of the constructed substrings yields a string σ that is a C1 order with multiplicity of M_ϕ. Thus if ϕ has a satisfying assignment τ, then M_ϕ has a C1 order σ with multiplicity.

Since the number of columns used in the construction is at most $5n+2m$, the number of rows is at most $4n+2m$, and each row is of degree at most 3, i.e., the construction is linear in the size of ϕ, it can be built in linear time, and hence, deciding the mC1P is NP-hard for matrices with degree at most 3, and no column has multiplicity greater than two. □

4.2 Two Variants of the *m*C1P

Besides its classical definition, there are different generalizations of the *m*C1P discussed in the literature, such as r-windows [40, 49], max-gap clusters [68, 69, 122], and approximate gene clusters [19, 125]. Since deciding the *m*C1P is NP-complete, any generalization is NP-hard as well.

In contrast to generalizations, there are also restricted variants of the *m*C1P that are relevant to settings in the reconstruction of AGOs. In the following, we will discuss such models, in particular the Consecutive-Ones Property with Multiplicity for Framed Rows (*m*C1P(fr)) and Consecutive-Ones Property with Multiplicity for Nested Rows (*m*C1P(ne)).

4.2.1 The *m*C1P(fr) Variant

The C1P of binary matrices where each row is framed by two columns, or the model of common intervals framed by two genes (whose orientations have to be conserved also), was first introduced as *conserved intervals* on permutations in Bergeron and Stoye [14]. In the reconstruction of AGOs, framed rows on permutations was the first model to formally state the problem of finding putative AGOs [15]. Here, we define the *m*C1P(fr), which models framed rows on *sequences*, to account for duplicate markers, for example.

Definition 39 (Framed Row of a Matrix)**.** *Let M be a binary matrix on the set of columns $S = \{1, \ldots, n\}$. A* framed row *(for $r \subseteq S$) of M is denoted $[a\ r\ b]$, where its two* extremities *(or framing columns) $a, b \in S$. We sometimes refer to the columns of r as the* inner columns *of this framed row. A framed row $[a\ r\ b]$ is* contained *in a sequence σ on S if, somewhere in σ, a and b appear with the set of characters of the substring between a and b taken only from r.*

Definition 40 (Consecutive-Ones Property with Multiplicity for Framed Rows (*m*C1P(fr)))**.** *A binary matrix M on the set of columns $S = \{1, \ldots, n\}$ with framed rows has the mC1P(fr) if there is a sequence σ that contains each framed row of M.*

The obvious relationship of the *m*C1P and the *m*C1P(fr) allows us to infer an important correlation of these properties: any instance of the *m*C1P can be reduced to an instance of the mC1P(fr). Based on this, we can deduce the following statement.

Theorem 41. *Given a degree 6 matrix M on set of columns S, deciding the mC1P(fr) is NP-complete for a multiplicity vector $\mathbf{m}$ where $\mathbf{m}(s) \leq 2$ for each $s \in S$.*

Proof. Again, one can easily formulate an algorithm that verifies a given solution for correctness in polynomial time, which shows that the problem belongs to the complexity class NP. NP-hardness is shown by reducing the case for the mC1P used in the proof of Theorem 38 to the mC1P(fr).

The basic idea is to replace each row $B = \{e_1, \ldots, e_m\}$ by a framed row $\mathbb{B} = [\tilde{B}\{e_1, \ldots, e_m, \ldots\}\bar{B}]$ containing, besides others, the columns of B as inner columns. Then, if this new instance allows for a valid sequence σ, there is a sequence σ' for the original instance of the mC1P by simply removing all newly introduced columns from σ such that only the columns contained in the rows B are left in σ'. Because the inner columns of all framed rows have to occur contiguously in σ, the columns of the original rows occur contiguously in σ'.

Since the rows of the matrix used in the proof of Theorem 38 overlap, the framing columns have to be included into the set of inner columns of overlapping framed rows. However, no row is included in another. This allows us to use the following technique which ensures that, if there is a valid sequence for the original matrix, there is a valid sequence for the constructed set of framed rows. Together with the argument in the previous paragraph, this will yield equivalence of the two instances of the C1P with multiplicity.

For each row $B = \{e_1, \ldots, e_m\}$ overlapping with rows $B_1, \ldots, B_k$, we create a framed row $\mathbb{B} = [\tilde{B}\{e_1, \ldots, e_m, \tilde{B}_1, \ldots, \tilde{B}_k, \bar{B}_1, \ldots, \bar{B}_k\}\bar{B}]$ containing the framing columns of $\mathbb{B}_1, \ldots, \mathbb{B}_k$, the framed rows constructed for $B_1, \ldots, B_k$. Note that this means that the framing columns $\tilde{B}$ and $\bar{B}$ also appear as inner columns to $\mathbb{B}_1, \ldots, \mathbb{B}_k$. All rows used in the proof of Theorem 38 have the property that, for a valid sequence (a C1 order with multiplicity), an occurrence of a given row can overlap with the occurrence of only one other row on each side. Assume, the occurrence of some B overlaps with B_l in $e_1, \ldots, e_l$ on one side and with B_r in $e_r, \ldots, e_m$ on the other side. Then, we can extend the substring that fulfills B to

$\tilde{B}, e_1, \ldots, e_l, \overline{B_l}, \ldots, \tilde{B}_r, e_r, \ldots, e_m, \overline{B}$. Between $\overline{B_l}$ and $\tilde{B}_r$ we include all remaining inner columns of $\mathbb{B}$ in an arbitrary order. The resulting substring fulfills $\mathbb{B}$ and also allows a realization of the framed rows created for B_l and B_r. This extension can be performed for all rows such that, finally, all framed rows are contained in the extended, overall string. What remains to be shown is that such a construction is possible using at most six inner columns in each framed common row, as well as that a maximum multiplicity of two is sufficient.

To minimize the number of inner columns, we do not always add both framing columns to all overlapping rows. The structure of the rows used in the gadgets of the proof of Theorem 38 restricts the possible overlaps of their occurrences in a valid sequence. As can be seen in the proof of Theorem 38, if there is a valid sequence, we can construct one using the following orders (or their reversals) of row occurrences within the gadgets:

$$P_\ell^3, P_\ell^1, P_\ell^2 \text{ or } P_\ell^4, P_\ell^1, P_\ell^2, P_\ell^3, \text{ and } S_i^1, S_i^2\,.$$

Within the gadget, P_ℓ^3 can only be followed by P_ℓ^1. We thus add $\overline{P_\ell^3}$ (but not $\tilde{P}_\ell^3$) to the inner columns of $\mathbb{P}_\ell^1$, and $\tilde{P}_\ell^1$ (but not $\overline{P_\ell^1}$) to those of $\mathbb{P}_\ell^3$. Analogously, we add $\overline{P_\ell^1}$ to $\mathbb{P}_\ell^2$ and $\tilde{P}_\ell^2$ to $\mathbb{P}_\ell^1$, $\overline{P_\ell^4}$ to $\mathbb{P}_\ell^1$ and $\tilde{P}_\ell^1$ to $\mathbb{P}_\ell^4$, $\overline{P_\ell^2}$ to $\mathbb{P}_\ell^3$ and $\tilde{P}_\ell^3$ to $\mathbb{P}_\ell^2$, and $\overline{S_i^1}$ to $\mathbb{S}_i^2$ and $\tilde{S}_i^2$ to $\mathbb{S}_i^1$.

A 2-clause gadget overlaps the gadgets of two variables, say x_j and x_k. As can be seen in the proof of Theorem 38, if there is a valid sequence, we can construct one with one of the following orders (or their reversals) of row occurrences:

$$P_j^p, S_i^1, S_i^2, \text{ or } P_k^q, S_i^1, S_i^2$$

where $p, q \in \{2, 3, 4\}$, depending on where it overlaps the variable gadgets. Thus, we add $\tilde{S}_i^1$ to the inner columns of $\mathbb{P}_j^p$ and $\mathbb{P}_k^q$.

A 3-clause gadget overlaps the gadgets of three variables, say x_j, x_k and x_ℓ in the column c_i^1, c_i^2 and c_i^3, respectively. As can be seen in the proof of Theorem 38, if there is a valid sequence, we can construct one with one of the following orders (or their reversals) of row occurrences:

$$P_j^p, S_i, P_k^q \text{ or } P_j^p, S_i, P_\ell^r \text{ or } P_k^q, S_i, P_\ell^r,$$

where $p, q, r \in \{2,3,4\}$, depending on where the variable gadgets are overlapped by S_i. We add $\tilde{S}_i$ to the inner columns of $\mathbb{P}_j^p$, $\overline{S_i}$ to the inner columns of $\mathbb{P}_k^q$, and $\tilde{S}_i$ and $\overline{S_i}$ to the inner columns of $\mathbb{P}_\ell^r$. This way, in any of the three cases, there is at least one copy of each framing column of $\mathbb{S}_i$ available on both sides.

In summary, we reduce a given set of rows as used in the proof of Theorem 38 to a set of framed rows with at most six inner columns as follows. For the rows $P_\ell^{1,2,3,4}$ used in the variable gadget for x_ℓ, we create the framed rows

$$\begin{aligned}
&\mathbb{P}_\ell^1 = [\tilde{P}_\ell^1 \, \{c_i^\alpha, c_k^\gamma, x'_\ell, \tilde{P}_\ell^2, \overline{P_\ell^3}, \overline{P_\ell^4}\} \, \overline{P_\ell^1}],\\
&\mathbb{P}_\ell^2 = [\tilde{P}_\ell^2 \, \{x'_\ell, c_j^\beta, \overline{P_\ell^1}, \tilde{P}_\ell^3\} \cup I_j^\beta \, \overline{P_\ell^2}],\\
&\mathbb{P}_\ell^3 = [\tilde{P}_\ell^3 \, \{c_j^\beta, c_k^\gamma, \tilde{P}_\ell^1, \overline{P_\ell^2}\} \cup I_k^\gamma \, \overline{P_\ell^3}] \text{ and}\\
&\mathbb{P}_\ell^4 = [\tilde{P}_\ell^4 \, \{c_i^\alpha, x''_\ell, \tilde{P}_\ell^1\} \cup I_i^\alpha \, \overline{P_\ell^4}], \text{ where}\\
&I_t^\mu = \begin{cases} \{\tilde{S}_t\} \text{ (or } \{\tilde{S}_t^1\} \text{ if } c_t^\alpha \text{ appears in a 2-clause)} & \text{if } \mu = \alpha \\ \{\overline{S_t}\} \text{ (or } \{\tilde{S}_t^1\}) & \text{if } \mu = \beta \\ \{\tilde{S}_t, \overline{S_t}\} \text{ (or } \{\tilde{S}_t^1\}) & \text{if } \mu = \gamma. \end{cases}
\end{aligned}$$

For the rows $S_i^{1,2}$ used in the 2-clause gadget for c_i, we create the framed rows

$$\begin{aligned}
&\mathbb{S}_i^1 = [\tilde{S}_i^1 \, \{c_i^1, c_i^2, c_i^*, \tilde{S}_i^2, \tilde{P}(c_i^1), \tilde{P}(c_i^2)\} \, \overline{S_i^1}] \text{ and}\\
&\mathbb{S}_i^2 = [\tilde{S}_i^2 \, \{c_i^*, c_i^{**}, \overline{S_i^1}\} \, \overline{S_i^2}],
\end{aligned}$$

where we define $\overline{P}(c_i^j)$ to be the right framing column of the (unique) $\mathbb{P}_\ell^m$ that contains c_i^1 and c_i^2 as inner columns.

For the row S_i used as in the 3-clause gadget for c_i, we create the framed row

$$\mathbb{S}_i = [\tilde{F}_i \, \{c_i^1, c_i^2, c_i^3, \tilde{P}(c_i^1), \tilde{P}(c_i^2), \tilde{P}(c_i^3)\} \, \overline{P_i}],$$

where we define $\tilde{P}(c_i^j)$ to be the left framing column of the (unique) $\mathbb{P}_\ell^m$ that contains c_i^1, and c_i^2 or c_i^3 as inner columns.

It remains to be shown that a maximum multiplicity of two for all newly added columns suffices. This is true, because each new column is included in the inner columns of at most two framed rows. In fact, we can assign a multiplicity of one to some of these columns. We define: $m(\bar{P}^{1,2,4}) = m(\tilde{S}_i^2) = m(\bar{S}_i^2) = 1$ and $m(\tilde{P}^{1,2,3,4}) = m(\bar{P}^3) = m(\tilde{S}_i) = m(\bar{S}_i) = 2$.

Since the number of columns used in the construction is at most $6n+8m$, the number of framed rows is at most $4n+2m$, and each framed rows contains at most six inner columns, i.e., the construction is linear in the size of ϕ, it can be built in linear time, and hence, deciding the mC1P(fr) is NP-hard for matrices with degree at most 6, and no column has multiplicity greater than two. □

Please note that, again, for a maximum multiplicity of one, polynomial solutions exist. Framed rows with no inner columns are equivalent to adjacencies, for which there is an efficient solution [151]. However, there is a gap left for framed rows with one to five inner columns. For these, the complexity is still open.

4.2.2 The *m*C1P(ne) Variant

Hoberman and Durand [69] discussed nestedness as a desired property of gene clusters (ancestral syntenies in our case) and proposed a first algorithm to identify respective clusters. Recently, Blin et al. [17] formally defined and studied *nested common intervals*, and gave efficient algorithms to detect them in genomes modeled both as permutations and as sequences. Here, we define a notion of the mC1P for nested rows.

Definition 42 (Nested Row of a Matrix)**.** *Let M be a binary matrix on the set of columns $S = \{1,\dots,n\}$. The structure of a nested row of M is defined recursively. A* nested row *in M is either*

(i) a row $\{a,b\}$ of degree 2, or

(ii) a tuple (c,a) of a nested row c and a column a.

A nested row (c,a) (resp., $\{a,b\}$) is contained *in a sequence σ on S if a is adjacent to a substring σ' of σ such that the character set of c is σ', and c is contained in σ' (resp., a and b are adjacent in σ). Here, the character set CS of a nested row is defined recursively as (i) $CS(\{a,b\}) = \{a,b\}$, and (ii) $CS((c,a)) = CS(c) \cup \{a\}$*

Example 43. *Consider the sequence* $\sigma = 5421236$*. The nested row* $((\{2,3\},1),4)$ *is contained in* σ *as illustrated below, where the occurrences of the (nested) sub-rows are indicated by lines:*

$$(5,\underline{4},2,\underline{\underline{1},\underline{2,3}},6)\,.$$

In contrast, $((\{1,3\},2),4)$ *is not contained in* σ *since, although* 4 *is adjacent to a substring with character set* $\{1,2,3\}$ *in* σ*, none of the occurrences of* 2 *is adjacent to a substring with character set* $\{1,3\}$.

Note that row $(\{2,3\},3)$ *is not contained in s, because* 3 *is not adjacent to a substring with character set* $\{2,3\}$*, whereas row* $(\{1,2\},2)$ *is contained in* σ*:*

$$(5,4,\underline{2,1},\underline{2},3,6)\,.$$

Definition 44 (Consecutive-Ones Property with Multiplicity for Nested Rows (mC1P(ne)))**.** *A binary matrix M on the set of columns* $S = \{1,\dots,n\}$ *with nested rows has the mC1P(ne) if there is a sequence* σ *on S that contains each nested row of M.*

We show now that even the strict assumption of nestedness is not strong enough to allow an efficient verification of this variant. In fact, similar to deciding the mC1P, there is no gap left for fixed-parameter tractability in the considered parameters.

Theorem 45. *Given a degree 3 matrix M on the set of columns S, deciding the mC1P(ne) for M is NP-complete for a multiplicity vector* $\mathbf{m}$ *where* $\mathbf{m}(s) \leq 2$ *for each* $s \in S$*.*

Proof. NP-hardness is proven by reduction from 3SAT(3) using a construction very similar to that of Theorem 38. Given 3SAT(3) formula ϕ, we will again design an instance M_ϕ of the matrix on *nested* rows comprising of clause gadgets and a variable gadget, and then argue why they simulate (or rather that deciding the mC1P(ne) for this instance simulates) exactly this instance ϕ.

For each 2-clause c_i with literals c_i^1 and c_i^2, we add to M_ϕ the two columns c_i^1

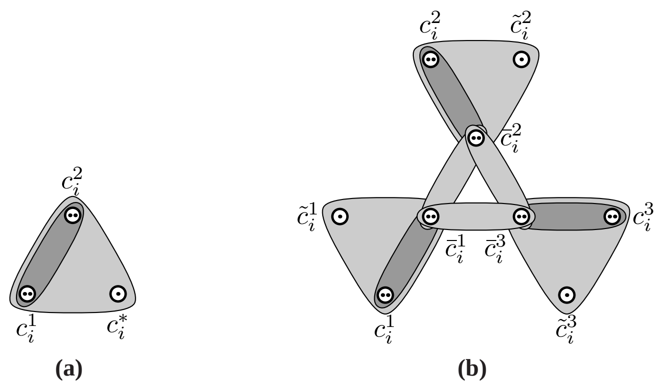


Figure 4.3: Graphical representations of the (a) 2-clause gadget and (b) 3-clause gadget for clause c_i in the mC1P(ne) case.

and c_i^2, each of multiplicity two, and the column c_i^* of multiplicity one, and the *nested* row $S_i^1 = (\{c_i^1, c_i^2\}, c_i^*)$. The 2-clause gadget is depicted in Figure 4.3a.

For each 3-clause c_i with literals c_i^1, c_i^2 and c_i^3, we add to M_ϕ the three columns c_i^1, c_i^2 and c_i^3, each with multiplicity two, the three columns $\tilde{c}_i^1, \tilde{c}_i^2$ and $\tilde{c}_i^3$, each with multiplicity one, the three columns $\bar{c}_i^1, \bar{c}_i^2$ and $\bar{c}_i^3$, each with multiplicity two and the six nested rows $S_i^1 = (\{c_i^1, \bar{c}_i^1\}, \tilde{c}_i^1), S_i^2 = (\{c_i^2, \bar{c}_i^2\}, \tilde{c}_i^2), S_i^3 = (\{c_i^3, \bar{c}_i^3\}, \tilde{c}_i^3), S_i^4 = \{\bar{c}_i^1, \bar{c}_i^2\}, S_i^5 = \{\bar{c}_i^2, \bar{c}_i^3\}, S_i^6 = \{\bar{c}_i^3, \bar{c}_i^1\}$. The 3-clause gadget is depicted in Figure 4.3b.

Note again that in both clause gadgets, at least one of the literal columns is selected in any valid string σ. For the 2-clause gadget, string σ has to contain one of the substrings c_i^1, c_i^2, c_i^* or c_i^2, c_i^1, c_i^*, or one of their reversals, thus a literal columns is always selected in this case. In the 3-clause gadget, if no literal column is selected in string σ, i.e., σ contains substrings $\tilde{c}_i^q, \bar{c}_i^q, c_i^q$ (or their reversals) for $q \in \{1,2,3\}$, there is only one remaining copy of $\bar{c}_i^q$ for $q \in \{1,2,3\}$ and hence there is no way that σ can be consistent with all of $S_i^{\{4,5,6\}}$ simultaneously, which is a contradiction. Therefore at least one literal column is selected in this case as well.

For each variable x_ℓ with the two positive occurrences c_i^α and c_j^β and the negated occurrence c_k^γ, we already added to M_ϕ the columns c_i^α, c_j^β and c_k^γ, each of multiplicity two in the corresponding clause gadgets for the clauses containing

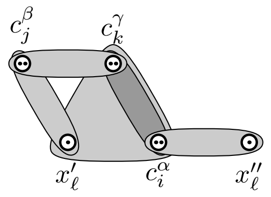


Figure 4.4: Graphical representation of the variable gadget for variable x_ℓ with positive occurrences c_i^α and c_j^β and negated occurrence c_k^γ in the clauses in the mC1P(ne) case.

x_ℓ and $\neg x_\ell$. We further add to M_ϕ the two columns x'_ℓ and x''_ℓ, each of multiplicity one, and the four nested rows $P_\ell^1 = (\{c_i^\alpha, c_k^\gamma\}, x'_\ell)$, $P_\ell^2 = \{x'_\ell, c_j^\beta\}$, $P_\ell^3 = \{c_j^\beta, c_k^\gamma\}$, $P_\ell^4 = \{c_i^\alpha, x''_\ell\}$. The variable gadget is depicted in Figure 4.4.

Now, consider a valid string σ where the literal c_k^γ is selected by some clause gadget. Since one copy of c_k^γ is used up by this clause gadget, σ must contain the substring $c_j^\beta, c_k^\gamma, c_i^\alpha, x'_\ell$ or its reversal because it is the only way to ensure consistency for nested rows P_ℓ^1 and P_ℓ^3 with the one remaining copy of c_k^γ. If literal c_j^β is also selected by some clause gadget, then there is no way that σ can be consistent with P_ℓ^2. While if literal c_i^α is also selected by some clause gadget, then there is no way that σ can be consistent with P_ℓ^4, which is a contradiction to the fact that σ is valid. It follows that if M_ϕ has a valid string σ, then ϕ is satisfiable.

We now show that the converse holds, namely if ϕ has a satisfying truth assignment τ, then M_ϕ has a valid string σ. Given τ, we construct σ as follows. For any variable x_ℓ with the two positive occurrences c_i^α and c_j^β and the negative occurrence c_k^γ, in τ, either of the two cases must hold:

1. c_i^α and c_j^β are *false* and c_k^γ is *true*: In this case, we create the substring $\tilde{c}_k^\gamma, c_k^\gamma, \overline{\text{ç}}_k^\gamma, \overline{\text{ç}}_k^\mu, c_k^\mu, \tilde{c}_k^\mu, \overline{\text{ç}}_k^\mu, \overline{\text{ç}}_k^\nu, c_k^\nu, \tilde{c}_k^\nu, \overline{\text{ç}}_k^\nu, \overline{\text{ç}}_k^\gamma$ satisfying $S_k^{\{1,\ldots,6\}}$ or $c_k^\mu, c_k^\gamma c_k^*$ satisfying S_k^1, depending on whether c_k^γ is part of a 3- or 2-clause. Further, we again create substrings $c_j^\beta, c_k^\gamma, c_i^\alpha, x'_\ell, c_j^\beta$ and c_i^α, x''_ℓ, fulfilling all $P_\ell^{\{1,2,3,4\}}$.

2. c_i^α and c_j^β are *true* and c_k^γ is *false*: In this case, we create the substrings $\tilde{c}_k^\alpha, c_k^\alpha, \overline{\text{ç}}_k^\alpha, \overline{\text{ç}}_k^\mu, c_k^\mu, \tilde{c}_k^\mu, \overline{\text{ç}}_k^\mu, \overline{\text{ç}}_k^\nu, c_k^\nu, \tilde{c}_k^\nu, \overline{\text{ç}}_k^\nu, \overline{\text{ç}}_k^\alpha$ satisfying $S_k^{\{1,\ldots,6\}}$ (resp., $c_i^\mu, c_i^\alpha, c_i^*$ satisfying S_i^1) and $\tilde{c}_k^\beta, c_k^\alpha, \overline{\text{ç}}_k^\beta, \overline{\text{ç}}_k^\mu, c_k^\mu, \tilde{c}_k^\mu, \overline{\text{ç}}_k^\mu, \overline{\text{ç}}_k^\nu, c_k^\nu, \tilde{c}_k^\nu, \overline{\text{ç}}_k^\nu, \overline{\text{ç}}_k^\beta$ satisfying $S_k^{\{1,\ldots,6\}}$ (resp., $c_j^\mu, c_j^\beta, c_j^*$ satisfying S_j^1). Further, we again create the substring

$x'', c_i^\alpha, c_k^\gamma x', c_j^\beta, c_k^\gamma$, fulfilling all $P_\ell^{1,2,3,4}$.

The requirements imposed by all given nested rows are fulfilled. Since none of the columns used in the first (resp., second) case are affected by the second (resp., first) case for any other variable (this time, c_i^α, c_j^β and c_k^γ appear only once in either of the two cases), the multiplicity constraint is met as well. Eventually, again, any concatenation of the constructed substrings yields a string σ that is valid w.r.t. M_ϕ. Thus, if ϕ has a satisfying assignment τ, then M_ϕ has a valid string σ.

Since the number of columns used in this construction is at most $5n+6m$, the number of nested rows is at most $5n+9m$, and each nested row is of size at most three, i.e., the construction is linear in the size of ϕ, it can be built in linear time, and hence, deciding the mC1P(ne) is NP-hard for matrices with degree at most 3, and no column has multiplicity greater than two. $\square$

Indeed, deciding the mC1P is a hard problem, since even two restricted versions of it are hard. In the next section, however, we present a class of deciding the mC1P that is tractable, motivated by handling telomeres in the reconstruction of AGOs.

4.3 A Tractability Result for the Consecutive-Ones Property with Multiplicity

In this section, we present a tractability result for a family of matrices where every row of M has (i) at most one entry `1` in columns with multiplicity greater than one, or (ii) exactly two entries `1` in columns with multiplicity greater than one and no other entries. Our proofs rely on the two classical concepts of PQ-trees and Eulerian graphs. We first give the following technical preliminaries.

4.3.1 Preliminaries

Let M be a binary matrix, with rows $R = \{r_1, \dots, r_m\}$, columns $S = \{s_1, \dots, s_n\}$ and ℓ entries `1`. We represent a row r of M as a subset of S, defined as the set of s_i such that $M[r, s_i] = 1$. A column s with multiplicity $\mathbf{m}(s) > 1$ is called a *multicolumn* and a row r containing a multicolumn (i.e., $M[r, s] = 1$ for some column s with $\mathbf{m}(s) > 1$) is called a *multirow*. A multirow that does not contain any other multirow is called *minimal*. We say a binary matrix M with multiplicity vector $\mathbf{m} : S \to \mathbb{N}$

M	1	2	3	4	5	a	b
r_1	1	1	0	0	0	1	1
$\hat{r}_1$	1	1	0	0	0	0	0
r_2	1	1	1	0	0	0	0
r_3	0	0	1	1	1	0	1
$\hat{r}_3$	0	0	1	1	1	0	0
r_4	0	0	0	1	1	0	1
$\hat{r}_4$	0	0	0	1	1	0	0
r_5	1	0	0	1	1	0	0

(a)

$\hat{M}$	1	2	3	4	5
r_1	1	1	0	0	0
r_2	1	1	1	0	0
r_3	0	0	1	1	1
r_4	0	0	0	1	1
r_5	1	0	0	1	1

(b)

Figure 4.5: (a) Binary matrix M, with matched multirows. Let $\mathbf{m}(1) = \cdots = \mathbf{m}(5) = 1$ and $\mathbf{m}(a) = \mathbf{m}(b) = 2$: a and b are multicolumns and r_1, r_3 and r_4 are multirows. Row r_3 is not minimal, because it contains r_4. (b) The corresponding matrix $\hat{M}$. Since in $\hat{M}$, by definition $\hat{r}_i = r_i$ for all multirows r_i, the matched multirows are discarded.

has *matched multirows* if, for every multirow $r \subseteq S$ that contains at least two entries 1 in non-multicolumns, there exists a row $\hat{r}$ which is a copy of r where all entries in multicolumns have been discarded (i.e., switched from 1 to 0). We denote by $\hat{M}$ the binary matrix obtained from M by discarding all multicolumns. In this work, we assume that all matrices we deal with have matched multirows unless otherwise stated. Figure 4.5 illustrates the above definitions.

We now have the important lemma about the mC1P of matrices with matched mutlirows, which leads to this tractability result.

Lemma 46. *Every C1 order with multiplicity of M with multiplicity vector* $\mathbf{m}$ *contains a C1 order of $\hat{M}$ as a subsequence. As a consequence, if a binary matrix M has the mC1P, then $\hat{M}$ has the C1P.*

This lemma suggests that, to decide if M has the mC1P for a given multiplicity vector $\mathbf{m}$, we can first check if $\hat{M}$ has the C1P, and then extend a C1 order of $\hat{M}$ into an C1 order with multiplicity of M by adding copies of multicolumns. Note that the matrix $\hat{M}$ in Figure 4.5 does not have C1P, and hence, M does not have mC1P. However, if we omit column r_5, then 12345 is a C1 order of $\hat{M}$, which can be extended to the following C1 order with multiplicity of M: $ab12345b$. To account for the fact that there can be an exponential number of C1 orders of $\hat{M}$, we use PQ-trees, a linear size structure that can describe all C1 orders of $\hat{M}$, defined below. For a more complete treatment of PQ-trees, we refer the reader to Booth and Lueker [21] or Meidanis et al. [106].

The *frontier* $F(T)$ of a PQ-tree T of a matrix M on columns S is the sequence of S obtained by reading the labels of its leaves from left to right. The *frontier* of an internal (P- or Q-) node N in T is the frontier of the subtree rooted at N. Let $\{F(N)\}$ be the set of elements appearing in the sequence $F(N)$. Two PQ-trees are equivalent if one can be obtained from the other by applying a sequence of the following transformation rules: (RP) arbitrarily permute the children of a P-node; (RQ) reverse the order of the children of a Q-node.

Theorem 47. Booth and Lueker [21] *If a binary matrix M has the C1P, there exists a unique equivalence class PQ_M of PQ-trees with the property that there is a one-to-one correspondence between the C1 orders of M and the frontiers of the PQ-trees of PQ_M, and a PQ-tree belonging to PQ_M can be constructed in linear time.*

Each PQ-tree in the equivalence class PQ_M satisfies the following properties (that are implicitly given in Booth and Lueker [21] and McConnell [102]) which we will use in this section.

Property 48. *Let M be a binary matrix that has the C1P with rows R and T a PQ-tree in the equivalence class PQ_M. Then*

1. *for every row $r \in R$, there is a node N in T such that either $\{F(N)\} = r$, if N is a P-node, or r is consecutive in $F(N)$, if N is a Q-node;*

2. *for every node N different from the root of T, there is a row $r \in R$ such that $\{F(N)\} \subseteq r$; and*

3. *for every Q-node N, and every two consecutive children N_1 and N_2 of N, there is a row $r \in R$ such that $\{F(N_1)\} \cup \{F(N_2)\} \subseteq r$.*

Finally, we recall briefly the technique used to prove that matrices with two entries 1 per row (usually called matrices of *degree* 2) form a class of tractable instances for deciding the *m*C1P as we will use it to prove our main result. Such matrices can be naturally represented as a collection of adjacency constraints $\mathscr{A} = \{\{a_i, b_i\}\}_{i=1}^{m}$ on the set S, where $a_i \neq b_i$ and the collection is a set (no duplicate elements). Collection $\mathscr{A}$ is *consistent* with respect to $\mathbf{m}$ if there is a sequence σ on S such that each adjacency is consecutive in σ. We will refer to this sequence as a *consistency sequence* of $\mathscr{A}$ and $\mathbf{m}$. Note that an C1 order with multiplicity of M is a consistency sequence of the corresponding collection $\mathscr{A}$ and $\mathbf{m}$, and vice versa, and hence, M has the mC1P for $\mathbf{m}$ if and only if $\mathscr{A}$ is consistent with respect to $\mathbf{m}$. Given a collection of adjacencies $\mathscr{A}$, we define the graph $G_{\mathscr{A}}$ with vertex set S and edges given by adjacencies.

Theorem 49. Wittler and Stoye [151] *A collection of adjacencies $\mathscr{A}$ is consistent with respect to a multiplicity vector $\mathbf{m}$ if and only if for all $s_i \in S$, $\text{degree}_{G_{\mathscr{A}}}(s_i) \leq 2\mathbf{m}(s_i)$ and for each connected component $B \subseteq S$ of $G_{\mathscr{A}}$, for at least one $s_i \in B$, $\text{degree}_{G_{\mathscr{A}}}(s_i) < 2\mathbf{m}(s_i)$.*

The above theorem relies on the fact that the graph $G_{\mathscr{A}}$ satisfying the above conditions can be extended to a multigraph on $S \cup \{s_0\}$ that has an Eulerian cycle. It can be easily seen that the proof presented in Wittler and Stoye [151] applies to generalized adjacencies, where we allow $a_i = b_i$ and the collection to be a multiset, and we require that each adjacency in $\mathscr{A}$ appears in σ in a unique position. Note that $G_{\mathscr{A}}$ is now a multigraph with self-loops. We have the following corollary.

Corollary 50. *A collection of generalized adjacencies $\mathscr{A}$ is consistent with respect to a multiplicity vector $\mathbf{m}$ if and only if for all $s_i \in S$, $\text{degree}_{G_{\mathscr{A}}}(s_i) \leq 2\mathbf{m}(s_i)$ and for each connected component $B \subseteq S$ of $G_{\mathscr{A}}$, for at least one $s_i \in B$, $\text{degree}_{G_{\mathscr{A}}}(s_i) < 2\mathbf{m}(s_i)$.*

4.3.2 A Tractable Case of Deciding the *m*C1P

Our main result is that deciding the mC1P is tractable for a large family of matrices with constraints on the maximum number of entries `1` in multicolumns a row can have. The motivation for studying this particular family of matrices arises from incorporating information on telomeres in ancestral gene order reconstruction (cf. Chapter 1)

Theorem 51. *Let M be a binary matrix and $\mathbf{m}$ a multiplicity vector such that*

(1) M has matched multirows, and

(2) each row contains either (i) at most one entry `1` *in multicolumns, or (ii) two entries* `1` *in multicolumns and no other entries. Deciding if M has the mC1P for* **m** *can be done in polynomial time and space.*

We split the proof into two parts. First, we consider the case (2i) where M with multiplicity vector **m** contains a single multicolumn, and we show that deciding if M has the *m*C1P for **m** can be done efficiently using PQ-trees. Then we show how to handle the general case using Corollary 50 which relies on Eulerian cycles. Finally, in Section 4.3.3, we give an algorithm for building a PQ-tree which describes all sequences that satisfy the consecutivity requirement (condition (i) of Property 3 defined in Chapter 1).

The Case of a Single Multicolumn

We assume that the multiplicity vector **m** defines only one multicolumn denoted by c'. According to Lemma 46, M satisfies the *m*C1P only if $\hat{M}$ has the C1P, which can be checked in linear time (Theorem 47). Assume that $\hat{M}$ has the C1P and let T be a PQ-tree from the equivalence class $PQ_{\hat{M}}$. We then aim at finding a PQ-tree from $PQ_{\hat{M}}$ (by applying operations (RP) and (RQ) on T) whose frontier can be extended to a valid C1 order with multiplicity by inserting copies of c'. We say that inserting a copy of c' into $F(T)$ *breaks* a row r of $\hat{M}$ if r is not consecutive in the resulting sequence. An example is given in Figure 4.6.

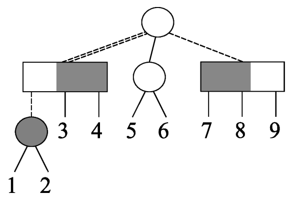
Recall that rows are subsets of S. As M has matched multirows, all rows in $\hat{M}$ are also rows in M. Since the consecutivity of the `1`'s in each row of $\hat{M}$ in the frontier $F(T)$ has to be maintained when inserting copies of c', no c' can be inserted into a position where it breaks any row of $\hat{M}$. Lemma 52 below is a consequence of this observation.

Lemma 52. *Let M be a binary matrix with matched multirows, and* **m** *be a multiplicity vector defining exactly one multicolumn c'. Assume that M has the mC1P, and let T be a PQ-tree from $PQ_{\hat{M}}$ and T' an extension of T whose frontier $F(T')$ is an mC1 order of M.*

1. *If the root of T is a P-node, then, for each child node N of the root, c' can only appear as the first or last element of the frontier $F(N)$ in T'.*

	1	2	3	4	5	6	7	8	9	c'
r_1	1	1	0	0	0	0	0	0	0	1
$\hat{r}_1$	1	1	0	0	0	0	0	0	0	0
r_2	1	1	1	0	0	0	0	0	0	0
r_3	0	0	1	1	0	0	0	0	0	1
$\hat{r}_3$	0	0	1	1	0	0	0	0	0	0
r_4	0	0	0	0	0	0	1	1	0	1
$\hat{r}_4$	0	0	0	0	0	0	1	1	0	0
r_5	0	0	0	0	0	0	0	1	1	0
r_6	0	0	0	0	1	1	0	0	0	0

(a)



(b)

Figure 4.6: (a) Binary matrix M, with matched multirows. Let $\mathbf{m}(c') = 2$. (b) PQ-tree belonging to the equivalence class $PQ_{\hat{M}}$. P-nodes are represented by circular nodes and Q-nodes by rectangular nodes. An example of a valid C1 order with multiplicity is $c'1234c'78956$ which is obtained by taking the equivalent PQ-tree with frontier 123478956 and inserting two copies of c' into the corresponding positions. Notice that inserting c' between 2 and 3 would break row r_2.
Illustration of Algorithm 2. LCA$(\hat{r}_1)$ and the respective segments of LCA$(\hat{r}_{3,4})$ are highlighted in gray and the respective paths are depicted by dashed lines. The upper left edge is contained in two paths. Here, $K_1 = 1$ and $K_2 = 1$, thus $K = 2 \leq \mathbf{m}(c') = 2$.

2. *If the root of T is a Q-node, the copies of c' in T' can only appear as the first and/or last element of the frontier $F(T')$.*

Proof. It follows by Property 48.2 that for every child N of the root of T, any pair of consecutive leaves in $F(N)$ belongs to a row of $\hat{M}$, and hence, inserting c' between these leaves breaks this row.

In addition, if the root of T is a Q-node, then by Property 48.3, for any two consecutive children N_1 and N_2 of the root, there is a row of $\hat{M}$ that contains elements of $F(N_1)$ and of $F(N_2)$. This prevents the insertion of c' into root between N_1 and N_2 as this would break such a row. Hence c' can appear only at the extremities of $F(T')$. □

Lemma 52 rules out many positions in $F(T)$ where to insert copies of c': indeed, copies of c' can only be inserted at extremities of the subsequences of $F(T)$ formed by children of the root (and only at the extremities of $F(T)$, if the root is a

Q-node). On the other hand, each multirow specifies a position where a copy of c' must be inserted. These two constraints give rise to a polynomial algorithm which we describe in the following.

Algorithm 2 starts with a PQ-tree for $\hat{M}$ and works in two stages. First (Step 3), based on Lemma 52, it checks if there is a way to permute nodes in the subtrees rooted at each child of the root such that for each multirow $r = \hat{r} \cup \{c'\}$, rows in $\hat{r}$ appear as a prefix or a suffix of the frontier of some child. To satisfy the consecutivity requirement for each multirow r it is enough to add copies of c' to $F(T)$ before or after the frontier of the child of the root containing $\hat{r}$. To satisfy the multiplicity constraint imposed by $\mathbf{m}$, we need to permute the children of the root and possibly reverse the order of the frontier of some children. The basic idea is that we can save one copy of c' if a child requiring a copy of c' on the right is followed by a child requiring a copy of c' on the left. Whether enough copies of c' can be saved to satisfy the multiplicity constraint is checked in Steps 4–5.

Let $r = \hat{r} \cup \{c'\}$ be a multirow. By Property 48.1, there is in T either a P-node that contains exactly the columns in $\hat{r}$ in its subtree, or a Q-node with a segment of two or more consecutive children which together contain exactly the columns in $\hat{r}$ in their subtrees. This node is the least common ancestor in T of the columns in $\hat{r}$, and hence, will be denoted by LCA$(\hat{r})$.

Now to argue that Algorithm 2 is correct. If condition 3.c.i applies, r would require the insertion of a copy of c' within $F(U)$ in any PQ-tree of $PQ_{\hat{M}}$, which contradicts Lemma 52. The paths indicate positions where copies of c' have to be added to the frontier so that the consecutivity requirement is satisfied. Following Lemma 52, we have to verify whether we can transform T such that all paths lie on the outside of the subtree of a child of the root of T. If conditions 3.c.ii–3.c.iv apply, there are two or more competing multirows, and we cannot transform T such that all of the corresponding paths lie on the outside of the subtree of a child of the root of T. Paths that are sub-paths of one another are excluded by not considering any multirow $r = \hat{r} \cup \{c'\}$ which contains another multirow $r' = \hat{r}' \cup \{c'\}$ (line 3). These rows do not need to be considered at this stage, because in any order with c' adjacent to the elements in $\hat{r}'$, since $\hat{r}' \subseteq \hat{r}$, c' is also adjacent to the elements in $\hat{r}$. If the root of T is a P-node, we have to consider the children of the root node separately: We could insert a copy of c' on both sides of a frontier of a child of the

Algorithm 2 Deciding the mC1P for a matrix M with matched multirows and a multiplicity vector $\mathbf{m}$ defining a single multicolumn c'.

```
1. Check if $\hat{M}$ has the $m$C1P.
2. If not, return false, else let $T$ be a PQ-tree from $PQ_{\hat{M}}$.
3. For each minimal multirow $r = \hat{r} \cup \{c'\}$ in $M$ do
   a. Locate $N := \mathrm{LCA}(\hat{\mathrm{r}})$.
   b. Let $P_r$ be the path from $N$ to the root of $T$.
   c. For each edge $e = \{U, V\}$ in $P_r$, where $U$ is the parent of $V$ do
      i. If $U$ is a Q-node and $V$ is neither its first nor its last child, return false;
      ii. If the root of $T$ is a Q-node and $e$ also belongs to the path $P_{r'}$ defined by
          another minimal multirow $r'$, return false;
      iii. If $U$ is not the root of $T$ and $e$ also belongs to the path defined by another
           minimal multirow, return false;
      iv. If $U$ is the root of $T$ and $e$ also belongs to the paths defined by at least
          two other minimal multirows, return false.
4. If the root of $T$ is a Q-node, return *true*.
5. If the root of $T$ is a P-node:
   a. Let $K_1$ and $K_2$ be the number of children of the root of $T$ belonging to exactly
      one or two paths defined by minimal multirows, respectively.
   b. $K := \left\lceil \frac{K_1}{2} \right\rceil + K_2 + \begin{cases} 1 & \text{if } K_1 = 0 \text{ and } K_2 > 0, \\ 0 & \text{otherwise.} \end{cases}$
   c. Return $K \leq \mathbf{m}(c')$
```

root, i.e., at most two paths can join above such a child node. In levels below the root, only one path can be moved to the border of the subtree, i.e., no two edges can join.

If conditions 3.c.i–iv do not apply for a multirow r, there is a way to transform T (with rules (RP) and (RQ)) in the nodes on the path P_r (excluding the root) so that the frontier of $N = \mathrm{LCA}(\hat{\mathrm{r}})$ appears as a prefix or suffix of the frontier of N', where N' is a child of the root lying on the path P_r. Next, we will show that all these transformations can be performed simultaneously without any conflict. Obviously, the conflicts could only occur if the paths P_r share vertices other than root. Condition 3.c.iv guarantees that there are never three or more minimal multirows in the same subtree rooted at a child N' of the root. Condition 3.c.iii guarantees that if there are two minimal multirows in the same subtree rooted at a child N' of the root, their paths must meet only in N', and hence, one can appear as a prefix and one as a suffix of the frontier of N'. However, if the root is a Q-node, by Lemma 52, column c' can be attached only on one side of the frontier of N', and hence, only one minimal multirow can appear in the subtree rooted at N', which is checked in condition 3.c.ii.

Hence, if Step 3 succeeds for all rows, there is a PQ-tree in $PQ_{\hat{M}}$ from which we can obtain a sequence of the columns fulfilling the consecutivity requirement of M by inserting copies of c' into its frontier at positions indicated by the paths of multirows. Steps 4–5 check if the multiplicity constraint imposed by $\mathbf{m}$ can be satisfied. Note, that if the root of T is a Q-node (Step 4), then the multiplicity constraint is satisfied since $\mathbf{m}(c') \geq 2$.

In Step 5, we count the number of copies of c' required to satisfy all multirows. The position where to insert these copies are given by the paths. Since the root of T is a P-node, we can rearrange the children of the root such that one copy of c' would coincide with two paths (from neighboring children). For instance, we can greedily pair nodes with one path each, using $\lceil K_1/2 \rceil$ copies and then include nodes with two paths (one path on each side) in-between, requiring one further copy each, K_2 in total. If $K_1 = 0$ and $K_2 > 0$, chaining the two-path nodes results in $K_2 + 1$ copies of c'. It is easy to see that this joining process is optimal.

If the number of required copies of c' does not exceed the given maximum multiplicity $\mathbf{m}(c')$, the given matrix M with multiplicity vector $\mathbf{m}$ has the mC1P. Finally, to complete the proof of the correctness of the algorithm, we only need to notice that the result of Algorithm 2 does not depend on the choice of the PQ-tree T of $PQ_{\hat{M}}$, as the LCAs and paths are invariant under the transformation rules (RQ) and (RP).

The analysis of the time and space complexity of Algorithm 2 is as follows. First, Steps 1 and 2 can be completed in $O(m+n+\ell)$ time and space using the algorithm described in McConnell [102]; note that T can then be encoded in $O(n)$

space. Next, Step 3 is composed of at most m iterations, each of them requiring time $O(n)$, the maximum length of a path from N to the root of T, as each path is obviously processed in time linear in its length. This gives an $O(mn)$ time complexity for Step 3. For similar reasons, Step 4 can be achieved in time $O(mn)$, which gives an overall worst-case time complexity of $O(mn)$. This completes the proof of the case of a single multicolumn in Theorem 51.

Completing the Proof of Theorem 51

Proof of Theorem 51. Given matrix M with multiplicity vector $\mathbf{m}$ and having matched multirows, let S' be its set of multicolumns. A multirow containing multicolumn $c' \in S'$, will be called a *c'-multirow*. Algorithm 3 works in the same two stages as Algorithm 2. However, the second stage is more complex. It requires building the collection of generalized adjacencies $\mathscr{A}$ on set $S' \cup \{s_0\}$ by replacing each child of the root of the PQ-tree T for $\hat{M}$ by an adjacency and then applying Corollary 50.

Algorithm 3 Deciding the mC1P for a matrix M with matched multirows and a multiplicity vector $\mathbf{m}$.

```
1. Run the first 4 steps of Algorithm 2, where $c'$ is any element of $S'$.
2. Construct a multiset of generalized adjacencies $\mathscr{A}$ on set $S' \cup \{s_0\}$ as follows.
   For every child $N$ of the root of $T$ do
     a. If $N$ belongs to exactly one path defined by multirows, say by a $c'$-
        multirow, add adjacency $\{c', s_0\}$ to $\mathscr{A}$;
     b. If $N$ belongs to two paths defined by multirows, say by a $c'$-multirow and
        a $d'$-multirow ($c'$ and $d'$ may be equal), add adjacency $\{c', d'\}$ to $\mathscr{A}$.
3. Report if $\mathscr{A}$ is consistent with respect to $\mathbf{m}$ (use Corollary 50).
```

Correctness of Step 1 follows from the correctness of the first stage of Algorithm 2. If Step 1 succeeds, we can assume that the root of T is a P-node (the case when the root is a Q-node is handled in Step 1), and hence, it is enough to satisfy the multiplicity constraint by permuting the children of the root and possibly reversing the order of the frontiers of some children. Let π be this order of children of the root. In Step 2, the algorithm constructs the multiset of generalized adjacencies $\mathscr{A}$ whose consistency sequence (produced in Step 3) describes the way to do this as follows. Children that belong to zero paths defined by multirows will not introduce any adjacency constraints and can be placed at the end of π in any order and orientation. For any other child of the root, we have a unique position in the consistency sequence, hence we can order and orient these children based on these positions. Next, we insert copies of multicolumns as follows. For each subsequence $c_1c_2c_3$ of the consistency sequence, where adjacency $\{c_1, c_2\}$ corresponds to child N_1 and $\{c_2, c_3\}$ to N_2, if $c_2 \neq c_0$, we insert a copy of c_2 between the frontiers of N_1 and N_2 in $F(T)$. Hence, the number of copies of a multicolumn $c' \in S'$ is equal to the number of its occurrences in the consistency sequence. Therefore, the frontier $F(T)$ with all required copies of multicolumns inserted satisfies the multiplicity constraint given by $\mathbf{m}$. It is easy to see that if there is an mC1 order of M, then we can extract from it an order of the children of the root which gives this consistency sequence.

The analysis of the time complexity is as follows. The first stage of the algorithm is a subroutine of Algorithm 2, and hence, has a time and space complexity of order $O(mn)$. Since the number of children of the root of T that belong to at least one path defined by multirows is at most m, the number of adjacencies in $\mathscr{A}$ is at most m, and hence, building $\mathscr{A}$ takes time $O(m)$. Finally, checking the degree conditions (applying Corollary 50) takes time $O(n)$. Hence, the total time and space complexity of the algorithm is $O(mn)$.

Finally, Algorithm 3 can also be easily extended to the case when the matrix also contains rows of degree 2 containing two multicolumns, as follows. First, we run Steps 1 and 2 where we ignore multirows containing two multicolumns. Then, we add to $\mathscr{A}$ also an adjacency for every such multirow. Finally, we run Step 3 of the algorithm on this new collection $\mathscr{A}$. It is easy to see that the time complexity of this new algorithm is still $O(mn)$. Hence, the theorem holds. $\square$

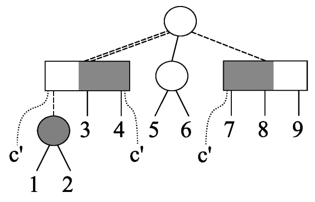


Figure 4.7: Augmented PQ-tree T' for the matrix given in Figure 4.6. (In fact, to get an augmented PQ-tree from the original PQ-tree shown in Figure 4.6, no modifications are necessary other than attaching leaf nodes labeled c' at appropriate locations.) Only the trees in the equivalence class of T' where the left side of the right Q-node is placed adjacent to the left Q-node have shortened frontiers that meet the multiplicity constraint ($\mathbf{m}(c') = 2$), for example, $c'\,1\,2\,3\,4\,c'\,7\,8\,9\,5\,6$.

4.3.3 Building a PQ-tree which Describes All Sequences that Satisfy the Consecutivity Requirement

Here, we describe how a given PQ-tree $T \in PQ_{\hat{M}}$ can be augmented to a PQ-tree T' which represents the set of all sequences σ, up to "pumping" occurrences of multicolumns, that satisfy the consecutivity requirement (condition (i) of Property 3 in Chapter 1) in that the frontier of any tree in the equivalence class of T' is such a sequence σ. However, not all frontiers meet the multiplicity constraint (condition (ii) of Property 3). For some trees in the equivalence class of T', the respective frontier contains pairs of adjacent occurrences of a multicolumn c', each of which can be replaced by one occurrence of c' without breaking any row of M (violating the consecutivity requirement). This reduces the number of used copies of the multicolumns. Only such shortened frontiers which meet the multiplicity constraint are valid *m*C1 orders, and, in fact, the set of such shortened frontiers is exactly the set of valid *m*C1 orders of M. Figure 4.7 shows an example.

To construct an augmented PQ-tree T', we process the original tree T in a bottom-up fashion along the paths P_r defined in Algorithm 2, starting with the LCAs. We replace an LCA by a new Q-node which has a copy of its corresponding multicolumn c' as its first child and further children, depending on whether the LCA itself and its parent are P or Q-nodes. These intuitive transformation rules are detailed in Figure 4.8. Then, any parent node of a newly obtained Q-node is refined

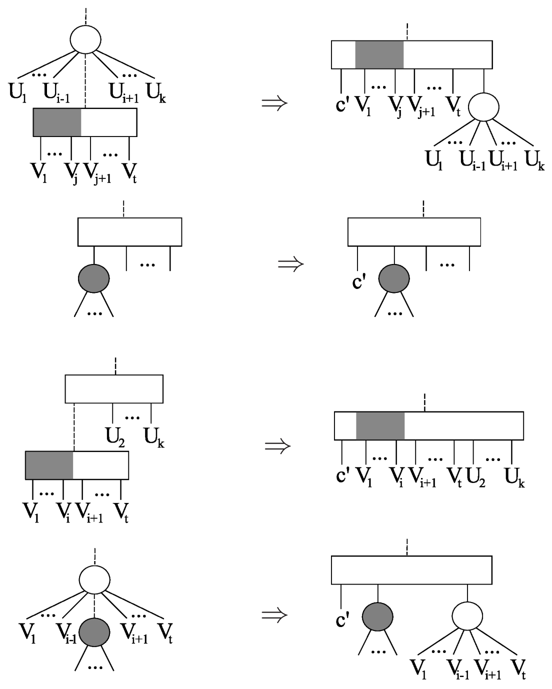


Figure 4.8: Transformation rules for the LCAs to construct an augmented PQ-tree. An LCA and its parent node are replaced by the nodes shown on the right. The LCA (or the segment of an LCA, respectively) are highlighted in gray.

to a new Q-node, moving up the copy of c', as shown in Figure 4.9. This process is iterated until we reach the root node. Since a node that is a child of the root can be contained in two paths, separate (but similar) rules are required, illustrated in Figure 4.10.

Further specific rules which apply if an LCA is a child of the root of T or if the root node is a Q-node are straightforward. In some cases, after generating the tree as described above, simplifications can be carried out, such as replacing a P-node with a single child by a direct edge or substituting a Q-node with two children by a P-node. Analogously to Algorithm 2, that only checks if a matrix has the mC1P, the above construction of an augmented PQ-tree T' can be carried out in $O(mn)$ time.

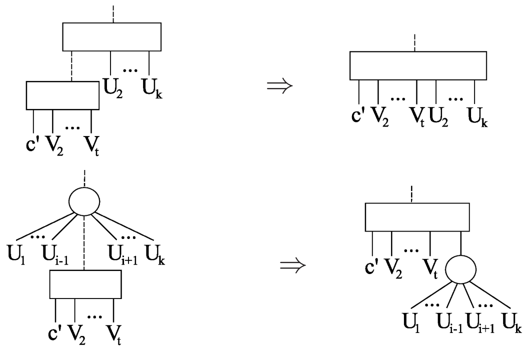
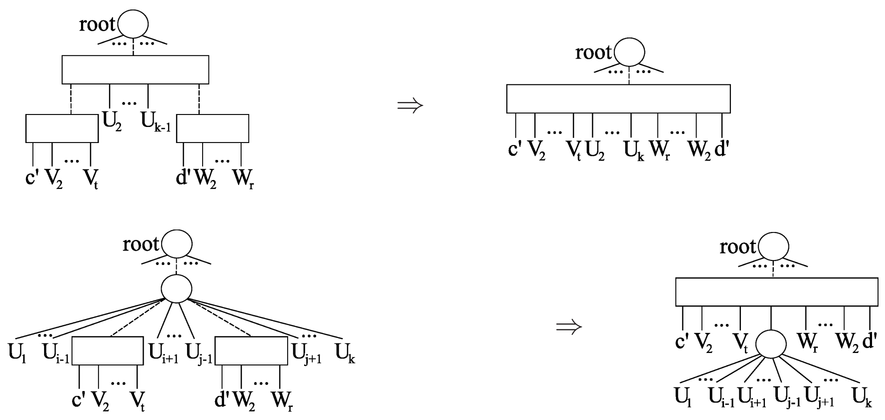


Figure 4.9: Transformation rules for bottom-up iteration to construct an augmented PQ-tree. A newly created Q-node and its parent node are replaced by the nodes shown on the right.



root
U2 … Uk-1
c' V2 … Vt
d' W2 … Wr
⇒
root
c' V2 … Vt U2 … Uk Wr … W2 d'
root
U1 … Ui-1 Ui+1 … Uj-1 Uj+1 … Uk
c' V2 … Vt
d' W2 … Wr
⇒
root
c' V2 … Vt Wr … W2 d'
U1 … Ui-1 Ui+1 … Uj-1 Uj+1 … Uk

Figure 4.10: Special transformation rules for bottom-up iteration to construct an augmented PQ-tree. A newly created Q-node two levels below the root node and its parent node are replaced by the nodes shown on the right.

Chapter 5

The Generalized Cladistic Character Compatibility Problem

The authors of Benham et al. [11, 12] give a polynomial-time algorithm for the case of the GCCC Problem where for each character, the set of states of each species forms a directed path in its character tree. It thus follows that if the character trees are non-branching, then the Incomplete Cladistic Character Compatibility Problem can be solved in polynomial time. The complexity of this case when each character has at most two states was further improved in Pe'er et al. [123]. In Benham et al. [11, 12], it was shown that the GCCC Problem is NP-complete using a construction involving character trees that are branching. However, the authors argued that in this setting the situation when a trait becomes hidden and then reappears does not happen, hence in Benham et al. [12] they posed an open case of the GCCC Problem where each character tree has one branch $0 \to 1 \to 2$ and the collection of sets of states for each species is $\{\{0\},\{1\},\{2\},\{0,2\}\}$. We call this the Benhan-Kannan-Warnow (BKW) Case. They then showed in Benham et al. [11] that if a "wildcard" set $\{0,1,2\}$ is added to the collection, the problem is NP-complete.

Here, we study the complexity of cases of the GCCC Problem for non-branching character trees with 3 states and set of states chosen from the set $\{\{0\},\{1\},\{2\},\{0,2\},\{0,1,2\}\}$ when the phylogeny tree that is a solution to this problem is restricted to be (a) any single-branch tree, (b) path or (c) tree, cf. Table 5.1. In Gramm et al. [57], the authors state that searching for path phylogenies is strongly motivated by the characteristics of human genotype data: 70% of real instances that admit a tree phylogeny also admit a path phylogeny.

This chapter is structured as follows, with the results summarized in Table 5.1. In Section 5.1 we formally define the Generalized Cladistic Character Compatibility Problem. In Section 5.2 we study several types of ordering problems, some being polynomial, while others are NP-complete; some of them is then used to determine the complexity of several cases in Table 5.1. Section 5.3 contains the tractability results of this chapter. Subsection 5.3.1 gives a polynomial-time algorithm based on that of Benham et al., Benham et al. [11, 12] for the case of the GCCC Problem for (a) where for each character, the set of states of each species forms a directed path in its character tree, giving entries (3a) and (7a) of Table 5.1. In Subsection 5.3.2, we first show that (5a–b) of Table 5.1 are equivalent to deciding the C1P. We then show that the BKW Case is polynomial-time solvable by giving an algorithm based on PQ-trees [21, 106] associated with the C1P, giving also the entries (6a), (8a) and (9a) of Table 5.1. In Subsection 5.3.3 we show that case (8b) is polynomial by showing that any instance of this case can be reduced to solving an instance of polynomial case (8a). Section 5.4 contains the intractability results of this chapter. Here we show that cases (10a–b) are NP-complete by reduction from the Path Triple Consistency (PTC) Problem of Section 5.2, and then how NP-completeness of an instance of the GCCC Problem for (a) can be transformed into certain instances of the same problem for (b) and (c). Finally, we show that cases (3b), (6b), (7b) and (9b) are NP-complete by reduction from the Left Element Fixed Path Triple Consistency (LEF-PTC) Problem of Section 5.2. Note that this last result includes the fact that case (9b), the BKW Case of the GCCC Problem for (b) is NP-complete.

5.1 The Generalized Cladistic Character Compatibility (GCCC) Problem

Let S be a set of species. A *generalized (cladistic) character* [11, 12] on S is a pair $\hat{\alpha} = (\alpha, T_\alpha)$, such that:

(a) α is a function $\alpha : S \to 2^{Q_\alpha}$, where Q_α denotes the set of states of $\hat{\alpha}$.

	$\mathscr{Q}\backslash$soln	(a) branch	(b) path	(c) tree
(1)	$\mathscr{Q} \subseteq \{\{0\},\{1\},\{2\}\}$	P [2]	P [2]	P [2]
(2)	$\{\{0,1,2\}\} \subseteq \mathscr{Q} \subseteq \{\{0\},\{1\},\{2\},\{0,1,2\}\}$; $\lvert\mathscr{Q}\rvert \leq 2$	trivial	trivial	trivial
(3)	$\{\{0,1,2\}\} \subseteq \mathscr{Q} \subseteq \{\{0\},\{1\},\{2\},\{0,1,2\}\}$; $\lvert\mathscr{Q}\rvert \geq 3$	P (Th. 62)	NP-c (Th. 72)	P [11, 12]
(4)	$\mathscr{Q} \subseteq \{\{0\},\{0,2\},\{0,1,2\}\}$ or $\mathscr{Q} \subseteq \{\{2\},\{0,2\},\{0,1,2\}\}$	trivial	trivial	trivial
(5)	$\{\{1\},\{0,2\}\}$	P (Lem. 63)	P (Lem. 63)	?
(6)	$\{\{0\},\{1\},\{0,2\}\}$	P (Th. 66)	NP-c (Th. 72)	?
(7)	$\{\{0\},\{2\},\{0,2\}\}(\cup\{\{0,1,2\}\})$	P (Th. 62)	NP-c (Th. 72)	P [11, 12]
(8)	$\{\{1\},\{2\},\{0,2\}\}$	P (Th. 66)	P (Cor. 69)	?
(9)	$\{\{0\},\{1\},\{2\},\{0,2\}\}$ *	P (Th. 66)	NP-c (Th. 72)	?
(10)	$\{\{1\},\{0,2\},\{0,1,2\}\} \subseteq \mathscr{Q}$	NP-c (Th. 70)	NP-c (Th. 70)	NP-c [11]

Table 5.1: Complexity of all cases of the GCCC Problem for the character tree $0 \to 1 \to 2$ and set of states chosen from the set $\mathscr{Q} \subseteq \{\{0\},\{1\},\{2\},\{0,2\},\{0,1,2\}\}$. The BKW Case is marked with *.

(b) $T_\alpha = (V(T_\alpha), E)$ is a rooted character tree with nodes bijectively labelled by the elements of Q_α.

The GCCC Problem is to find a perfect phylogeny [20] of a set of species with generalized characters:

Problem 53 (Generalized Cladistic Character Compatibility (GCCC) Problem)**.** *Given a set S of species and a set C of generalized characters on S, is there a rooted tree $T = (V_T, E_T)$ and a "state-choosing" function $c : V_T \times C \to \bigcup_{\hat{\alpha} \in C} Q_\alpha$ such that the following holds:*

(1) For each species $s \in S$ there is a vertex v_s in T such that for each $\hat{\alpha} \in C$, $c(v_s, \hat{\alpha}) \in \alpha(s)$.

(2) For every $\hat{\alpha} \in C$ and $i \in Q_\alpha$, the set $\{v \in V_T \mid c(v, \hat{\alpha}) = i\}$ is a connected component of T.

(3) For every $\hat{\alpha} \in C$, the tree $T(\alpha)$ is an induced subtree of T_α, where $T(\alpha)$ is the tree obtained from T by labelling the nodes of T only with their α-states (as chosen by c), and then contracting edges having the same α-state at their endpoints.

Essentially, the first condition is that each species is represented somewhere in the tree T, and the second condition is that the set of nodes labelled by a given state of a given character form a connected subtree of T, just as with the Character

Compatibility Problem. Finally, condition three is that the state transitions for each character $\hat{\alpha}$ must respect its character tree T_α.

The GCCC Problem is NP-complete [11, 12], however it is polynomial for many special cases of the problem [11, 12, 98]. In particular, in Benham et al. [11] it was shown to be NP-complete for a case where for each species s and character $\hat{\alpha}$, $\alpha(s) \in \{\{1\},\{0,2\},\{0,1,2\}\}$, and T_α is $0 \to 1 \to 2$. It was also shown to be polynomial-time solvable in the case where for each species $s \in S$, $\alpha(s)$ is a directed path in T_α for each $\hat{\alpha} = (\alpha, T_\alpha) \in C$ [12]. We will consider the following variants of the GCCC Problem. The GCCC with non-branching character trees (GCCC-NB) Problem is a special case of the GCCC Problem in which character trees have a single branch, i.e., each character tree T_α is $0 \to 1 \to \cdots \to |T_\alpha| - 1$. If we restrict the solution of the GCCC-NB Problem (a phylogeny tree) to have only one, or two branches starting at the root, we will call this problem the Single-Branch GCCC-NB (SB-GCCC-NB) Problem, and the Path GCCC-NB (P-GCCC-NB) Problem, respectively. In addition, if in any of these problems, say in problem X, we restrict the set of states to be from the set $\mathcal{Q}$, we will call this problem the $\mathcal{Q}$-X Problem. Table 5.1 summarizes the cases studied here.

5.2 Ordering Problems

In this section, we discuss several different types of ordering problems. These problems are related to the Single-Branch and P-GCCC-NB Problems. We will use one of these variants to obtain a hardness result in Section 5.4.

The PTC Problem is a simplified version of the extensively studied Quartet Consistency (QC) Problem [138]. In the QC Problem, given a set S and the collection of quartets $(a_i, b_i : c_i, d_i)$, where $a_i, b_i, c_i, d_i \in S$, the task is to construct a tree T containing vertices S such that for each quartet there is an edge of T whose removal separates vertices $\{a_i, b_i\}$ from vertices $\{c_i, d_i\}$. This problem was shown to be NP-complete in Steel [138]. Here, we show that the problem remains NP-complete when we restrict the tree to be a path. In this case it is easy to see that (i) we can assume the path contains only vertices in S and (ii) each quartet $(a_i, b_i : c_i, d_i)$ can be replaced with the three triples $(a_i, b_i : c_i)$, $(a_i, b_i : d_i)$ and $(c_i, d_i : a_i)$. The PTC Problem can be viewed as the Total Ordering (TO) Problem with negative constraints $c_i \notin [a_i, b_i]$, where $[a_i, b_i]$ is the set of all elements between a_i and b_i in the total order. The TO Problem with positive constraints $c_i \in [a_i, b_i]$ was shown to be NP-complete in Opatrny [114]. The formal definition of the PTC Problem is as follows.

Problem 54 (Path Triple Consistency (PTC) Problem)**.** *Given a set $S = \{1, \ldots, n\}$, and a set of triples $\{a_i, b_i : c_i | i = 1, \ldots, k\}$, where $a_i, b_i, c_i \in S$ for every $i = 1, \ldots, k$, is there a path (order) P on vertices S such that for each $i = 1, \ldots, k$, there is an edge e_i of P whose removal separates vertices $\{a_i, b_i\}$ from vertex c_i.*

Lemma 55. *The PTC Problem is NP-complete.*

Proof. The PTC Problem is actually complementary to the TO Problem, which was shown to be NP-hard by Opatrny in 1979 [114]. The TO Problem is, given a set $Q = \{1, \ldots, n\}$ and a set of triples $\{a_i, b_i, c_i | i = 1, \ldots, k\}$, where for $i = 1, \ldots, k$, $a_i, b_i, c_i \in S$, is there a path (order) on Q such that for each $i = 1, \ldots, k$, either $a_i < b_i < c_i$ or $c_i < b_i < a_i$. It is easy to see that the NP-completeness of the TO Problem implies the NP-completeness of the PTC Problem. Given instance of TO Problem $Q = \{1, \ldots, n\}$ and $\{a_i, b_i, c_i | i = 1 \ldots, k\}$, for the corresponding instance of the PTC Problem we let $S = Q$, and for each triple a, b, c of the instance of the TO Problem, we introduce the triples $a, b : c$ and $c, b : a$. $\square$

Now, we study two subclasses of the PTC Problem and one subclass of the TO Problem in which one element of each constraint is fixed.

Problem 56 (Left Element Fixed Path Triple Consistency (LEF-PTC) Problem)**.** *Given a set $S = \{1, \ldots, n\}$, an element $r \notin S$, and a set of triples $\{(a_i, r : c_i)\}_{i=1}^{k}$ where $a_i, c_i \in S$ for every $i \in \{1, \ldots, k\}$, is there a path (an order) P on vertices $S \cup \{r\}$ such that for each $i \in \{1, \ldots, k\}$, there is an edge of P whose removal separates $\{r, a_i\}$ from c_i.*

Problem 57 (Right Element Fixed Path Triple Consistency (REF-PTC) Problem)**.** *Given a set $S = \{1, \ldots, n\}$, an element $r \notin S$, and a set of triples $\{(a_i, b_i : r)\}_{i=1}^{k}$ where $a_i, b_i \in S$ for every $i \in \{1, \ldots, k\}$, is there a path (an order) P on vertices $S \cup \{r\}$ such that for each $i \in \{1, \ldots, k\}$, there is an edge of P whose removal separates $\{a_i, b_i\}$ from r.*

Problem 58 (One Element Fixed Total Ordering (OEF-TO) Problem)**.** *Given a set $S = \{1, \ldots, n\}$, an element $r \notin S$, and a set of triples $\{(a_i, b_i, c_i)\}_{i=1}^k$ where for every $i \in \{1, \ldots, k\}$, either $a_i, c_i \in S$ and $b_i = r$, or $a_i, b_i \in S$ and $c_i = r$, is there a path (a Total Ordering) P on vertices $S \cup \{r\}$ such that for each $i \in \{1, \ldots, k\}$, b_i appears between a_i and c_i on P.*

In what follows, we will show that the first problem LEF-PTC is NP-complete, while the other two problems REF-PTC and OEF-TO are solvable in polynomial time. Thus, the LEF-PTC Problem seems to be the simplest version of the problem which is still intractable.

Lemma 59. *The LEF-PTC Problem is NP-complete.*

Proof. Here, we give a reduction from the Not-All-Equal-3SAT (NAE-3SAT) Problem [53]. The NAE-3SAT Problem is: given a set of Boolean variables $X = \{x_1, \ldots, x_n\}$ and a set of clauses $\mathscr{C} = \{C_1, \ldots, C_m\}$, where each clause contains three literals, is there a truth assignment to the set of variables such that in no clause, its three literals are all true or all false. Given an instance of NAE-3SAT, let S be the union of variable symbols $\{x_1, \overline{x_1}, \ldots, x_n, \overline{x_n}\}$ and literal symbols $\{\ell_1^1, \ell_1^2, \ell_1^3, \ldots, \ell_m^1, \ell_m^2, \ell_m^3\}$.

The basic principle of the reduction is the following observation. The triple $(a_i, r : c_i)$ is equivalent to the following condition on the elements in $S \cup \{r\}$:

$$r < c_i \Leftrightarrow a_i < c_i\,. \tag{5.1}$$

The Boolean value of predicate $r < x_i$ will represent the value of variable x_i, for $i \in \{1, \ldots, n\}$. First, we introduce the triples $(x_i, r : \overline{x_i})$ and $(\overline{x_i}, r : x_i)$, for $i \in \{1, \ldots, n\}$. These triples are equivalent to the following logical statement: $r < \overline{x_i} \Leftrightarrow x_i < \overline{x_i} \Leftrightarrow x_i < r$. Hence, they enforce $\overline{x_i} < r$ iff $r < x_i$, and hence the Boolean value of predicate $r < \overline{x_i}$ represents the value of $\neg x_i$.

Now, let clause C_j contain variables x_{k_1}, x_{k_2} and x_{k_3}. We will use symbols $\ell_j^1, \ell_j^2, \ell_j^3$ to represent the values of the three literals of C_j: the Boolean value of the i-th literal of C_j will be equal to the value of predicate $r < \ell_j^i$. To achieve this, we will reuse the above constraints. For each variable x_{k_i} with positive occurrence in C_j, we introduce the triples $(\ell_j^i, r : \overline{x_{k_i}})$ and $(\overline{x_{k_i}}, r : \ell_j^i)$, and for each variable x_{k_i}

with a negated occurrence in C_j, triples $(\ell_j^i, r : x_{k_i})$ and $(x_{k_i}, r : \ell_j^i)$. These triples will guarantee that predicate $r < \ell_j^i$ represents the Boolean value of the i-th literal of C_j. The reason why we have a symbol for each literal is that the position of the literal symbol ℓ_j^i and the position of the variable symbol x_{k_i} (or $\overline{x_{k_i}}$) are only very weakly dependent: one is smaller than r if and only if the other is, but otherwise they are independent. This is important, since the clause gadgets introduced in the next paragraph might put some ordering restrictions on its literal symbols, and hence if we would use the variable symbols x_{k_i} (or $\overline{x_{k_i}}$) in several clause gadgets, the ordering restrictions from different clause gadgets might not be compatible.

The clause gadget for clause C_j will contain the three triples $(\ell_j^1, r : \ell_j^2)$, $(\ell_j^2, r : \ell_j^3)$ and $(\ell_j^3, r : \ell_j^1)$. The purpose of these constraints is to guarantee that in any order at least one and not all literals in the clause C_j are true. For instance, assume that all literals are true, i.e., $r < \ell_j^i$ for $i \in \{1,2,3\}$. By (5.1), this is equivalent to $\ell_j^1 < \ell_j^2$, $\ell_j^2 < \ell_j^3$ and $\ell_j^3 < \ell_j^1$, which leads to a contradiction. Similarly, if literals are false in the order, all three inequalities will reverse their direction, and we get a contradiction again. Hence, each clause is satisfied and predicates $r < v_i$ define a solution to the instance of NAE-3SAT.

Now, assume that the instance of NAE-3SAT has a solution $\psi : X \to \{\mathit{false}, \mathit{true}\}$. Consider the order of elements of $S \cup \{r\}$ satisfying the following conditions:

(a) for each $v_i \in \{v_1, \ldots, v_n\}$, v_i appears to the right of r, i.e., $r < v_i$ in the order, if and only if $\psi(x_i) =$ true for the x_i corresponding to v_i;

(b) for each clause C_j, the relative order of the literal symbols $\ell_j^1, \ell_j^2, \ell_j^3$ and r is one of the following: $(\ell_j^1, r, \ell_j^2, \ell_j^3)$, $(\ell_j^3, \ell_j^2, r, \ell_j^1)$, $(\ell_j^2, r, \ell_j^3, \ell_j^1)$, $(\ell_j^1, \ell_j^3, r, \ell_j^2)$, $(\ell_j^3, r, \ell_j^1, \ell_j^2)$ and $(\ell_j^2, \ell_j^1, r, \ell_j^3)$.

Note that for any valid combination of truth assignments to the literals of C_j, there is one order in the list above. This order imposes a restriction on the relative order of the two literal symbols appearing on the same side of r, the reason why we created the literal symbols. It is easy to see that for each $s \in S$, other than on which side of r the s appears, there is at most one constraint specifying its relative order to another element. Hence, it is always possible to find an order satisfying the above conditions.

Let us verify that this order satisfies all triple constraints. The constraints $(x_i, r : \overline{x_i})$ and $(\overline{x_i}, r : x_i)$ (respectively, $(\ell^i_j, r : x_{k_i})$ and $(x_{k_i}, r : \ell^i_j)$; $(\ell^i_j, r : \overline{x_{k_i}})$ and $(\overline{x_{k_i}}, r : \ell^i_j)$) are satisfied just by the placement of symbols to the correct sides of r. For instance, if $r < x_i$ then the relative order of $x_i, \overline{x_i}, r$ is $\overline{x_i}, r, x_i$ and this order satisfies both triples. For the constraints for clause C_j, only the relative order of elements $\ell^1_j, \ell^2_j, \ell^3_j$ and r is important. It is easy to check that any of the six orders of these elements listed above satisfies all three triples for C_j. Hence, the constructed order is a solution to the corresponding instance of the LEF-PTC Problem. $\square$

Lemma 60. *Any instance of the REF-PTC Problem always has a solution, and thus the problem is solvable in constant time.*

Proof. Consider any order of $S \cup \{r\}$ with r as the first (resp., last) element. Then the first (resp., last) edge separates r from any pair of elements in S. Thus, such an order is a solution to any instance of the REF-PTC Problem. $\square$

Lemma 61. *The OEF-TO Problem can be solved in linear time.*

Proof. The algorithm will work in two stages. In the first stage the elements will be clustered into parts each appearing on different sides of r. In the second stage, we will determine the ordering of the elements in each part.

Constraint (a_i, r, c_i) is satisfied if and only if a_i and c_i appear on opposite sides of r. Constraint (a_i, b_i, r) is satisfied if and only if (i) a_i and b_i appear on the same side of r, and (ii) b_i is closer to r than a_i, which we write as $b_i \prec a_i$. Consider the graph with vertex set S and edges between any two vertices $u, v \in$ such that u and v appear together in some triple (a_i, b_i, c_i). Let C be a connected component of this graph. It is easy to see that once we fix the side of one element in the component, the side of all elements in the component will be determined. Hence, we can uniquely partition C into two (paired) clusters such that all edges from constraints of type (a_i, r, c_i) are between two clusters and all edges from constraints of type (a_i, b_i, r) are inside one of the two clusters. Now, pick one cluster from each pair and place all its elements on one side of r and all other clusters to the other side. Note that there can many ways how to do this, the number of ways is exponential in the number of pairs of clusters.

It remains to satisfy the precedence conditions. These conditions ($b_i \prec a_i$) define a partial order on each side of r. Any total order compatible with these partial orders will form a solution to the problem. Such an order can be found in time $O(n+k)$. $\square$

5.3 Tractability Results

5.3.1 An Algorithm for Cases of the Single-Branch GCCC Problem

Here we show that when each $\alpha(s)$ induces a directed path in T_α, for each $\hat{\alpha} \in C$, $s \in S$, the Single-Branch GCCC (SB-GCCC) Problem is polynomial-time solvable. The algorithm we use, while much simpler, is based on the algorithm given in Benham et al. [11].

Theorem 62. *The SB-GCCC Problem is solvable in time $O(|S| \sum_{\hat{\alpha} \in C} |Q_\alpha|)$, if each $\alpha(s)$ induces a directed path in T_α, for each $\hat{\alpha} \in C$, $s \in S$.*

Proof. Consider an instance of the SB-GCCC Problem (S,C) with the required property. Let $\mathrm{start}_\alpha(s)$ and $\mathrm{end}_\alpha(s)$ be the first and the last node on the directed path induced by $\alpha(s)$. We define the partial order on the nodes of T_α by saying $v \preccurlyeq_\alpha w$ if the directed path from the root r_α of T_α to w passes through v. Similarly, for each solution (T,c) we define the partial order $\preccurlyeq_T$ on S based on T. Since T has a single branch, $\preccurlyeq_T$ is a total order, i.e., for every $s_1, s_2 \in S$, s_1 and s_2 are comparable by $\preccurlyeq_T$. Hence, for every $\hat{\alpha} \in C$, $c(s_1, \hat{\alpha})$ and $c(s_2, \hat{\alpha})$ are comparable by $\preccurlyeq_\alpha$. Therefore, for all $s \in S$, $c(s, \hat{\alpha})$ lie on a single branch (directed path starting in the root) P_α of T_α. Since $\mathrm{start}_\alpha(s_1) \preccurlyeq_\alpha c(s_1, \hat{\alpha})$ and $\mathrm{start}_\alpha(s_2) \preccurlyeq_\alpha c(s_2, \hat{\alpha})$, we can assume that for all $s \in S$, $\mathrm{start}_\alpha(s)$ lie on a subpath P'_α of P_α starting in the root r_α of T_α and ending in $\mathrm{start}_\alpha(\ell_\alpha)$, where $\ell_\alpha \in S$ and $\mathrm{start}_\alpha(s) \preccurlyeq_\alpha \mathrm{start}_\alpha(\ell_\alpha)$ for every $s \in S$. If that is not the case, there is no solution. This can be checked in time $O(|S||Q_\alpha|)$ for each $\hat{\alpha} \in C$.

Next, we will argue that it is enough to consider only solutions in which c maps all elements in S to P'_α. Consider a solution (T,c). Any $c(s, \hat{\alpha}) \notin P'_\alpha$ must lie on the subpath of P_α ending at vertex $\mathrm{start}_\alpha(\ell_\alpha)$. Since $\mathrm{start}_\alpha(s) \preccurlyeq_\alpha \mathrm{start}_\alpha(\ell_\alpha)$, we can remap $c(s, \hat{\alpha})$ to $\mathrm{start}_\alpha(\ell_\alpha)$. It is easy to check that conditions (1)–(3) of the GCCC

Problem remain satisfied after mapping all such $c(s,\hat{\alpha})$ to $\text{start}_\alpha(\ell_\alpha)$. Hence, we can assume that $c(s,\hat{\alpha}) \in \alpha'(s) = \alpha(s) \cap P'_\alpha$, for each $\hat{\alpha} \in C$ and $s \in S$. Note that for all $s \in S$, $\alpha'(s)$ induce directed subpaths of P'_α.

Now, we are ready to present the algorithm for solving the SB-GCCC Problem with the required property. First, we will build a set $\mathscr{C}$ of constraints on the ordering of the nodes of T which have to be satisfied in any solution (T,c). If for $s_1, s_2 \in S$ and $\hat{\alpha} \in C$, the paths induced by $\alpha'(s_1)$ and $\alpha'(s_2)$ are disjoint, and the path induced by $\alpha'(s_1)$ is closer to the root r_α, then we must have $s_1 \prec_T s_2$. Therefore, we add this constraint to the set $\mathscr{C}$. Let T be a single branch tree that satisfies all these constraints in $\mathscr{C}$ and let $s_1 \prec_T s_2 \prec_T \cdots \prec_T s_{|S|}$ be the elements of S ordered according to this tree. (If such a tree does not exist, there is no solution.) For each character $\hat{\alpha} \in C$, we will map $c(s_i,\hat{\alpha})$ to $\alpha'(s_i)$ using Algorithm 4, where $\max(a,b)$ is the element (a or b) further from the root if a and b are comparable, and undefined otherwise.

Algorithm 4 Iterative algorithm that assigns to each species a state.

```
1: $c(s_1,\hat{\alpha}) \leftarrow \text{start}_\alpha(s_1)$
2: **for** $i = 2$ up to $|S|$ **do**
3:     $c(s_i,\hat{\alpha}) \leftarrow \max(\text{start}_\alpha(s_i), c(s_{i-1},\hat{\alpha}))$
4: **end for**
```

Let us verify that (T,c) is indeed a solution. First, note that since all $\text{start}_\alpha(s_i)$ lie on the path P'_α, the arguments of the max function are always comparable. Furthermore, it is easy to see that all $c(s_i,\hat{\alpha})$ are assigned to the set $\{\text{start}_\alpha(s); s \in S\}$, and that $c(s_1,\hat{\alpha}) \preccurlyeq_\alpha c(s_2,\hat{\alpha}) \preccurlyeq_\alpha \ldots \preccurlyeq_\alpha c(s_{|S|},\hat{\alpha})$. It remains to show that for each i, $c(s_i,\hat{\alpha}) \in \alpha'(s_i)$. Let i be the smallest index for which $c(s_i,\hat{\alpha}) \notin \alpha'(s_i)$. We must have that $\text{end}_\alpha(s_i) \prec_\alpha c(s_i,\hat{\alpha})$. Since $c(s_i,\hat{\alpha}) = \text{start}_\alpha(s_j)$ for some $j < i$, the subpath of P'_α induced by $\alpha'(s_i)$ is closer to the root than the subpath induced by $\alpha'(s_j)$. Hence, $\mathscr{C}$ must contain the constraint $s_i \prec_T s_j$, which contradicts the fact that T satisfies all these constraints. It follows that (T,c) is a solution.

Finally, let us analyze the running time of the algorithm. We can verify whether this set $\mathscr{C}$ of constraints defines a partial order and find a total order T compatible with this partial order in time $O(|S|+m)$, where m is the number of constraints. For each $\hat{\alpha} \in C$, we can have at most $|Q_\alpha|$ disjoint induced paths, and it is enough to consider the constraint between the neighbouring disjoint induced paths only. Hence, $m = O(\sum_{\hat{\alpha} \in C} |Q_\alpha|)$. ☐

We remark that this type of theorem does not hold for the case of path phylogeny, cf. Table 5.1.

5.3.2 The BKW Case of the SB-GCCC-NB Problem is Polynomial-Time Solvable

First, we show that the $\{\{1\},\{0,2\}\}$-SB-GCCC-NB and $\{\{1\},\{0,2\}\}$-P-GCCC-NB Problems are polynomial-time solvable, by showing that they are equivalent to deciding the C1P. We then build on the algorithm for constructing a PQ-tree for a binary C1P matrix [21] to show that the $\{\{0\},\{1\},\{2\},\{0,2\}\}$-SB-GCCC-NB Problem (the BKW Case of the SB-GCCC-NB Problem) is also polynomial-time solvable.

Lemma 63. *The* $\{\{1\},\{0,2\}\}$*-SB-GCCC-NB and* $\{\{1\},\{0,2\}\}$*-P-GCCC-NB Problems are polynomial-time solvable.*

Proof. The solutions to the $\{\{1\},\{0,2\}\}$-SB-GCCC-NB and $\{1\},\{0,2\}\}$-P-GCCC-NB Problems must fall on a single-branch tree and path, respectively. Because T_α is $0 \to 1 \to 2$ for any character $\hat{\alpha}$, all species where $\hat{\alpha}$ has state 1 must appear consecutively in this single-branch tree (resp., path), otherwise there would be more than one transition from 0 to 1 in the phylogeny, for some character $\hat{\alpha}$. In this case of the SB-GCCC-NB Problem, all other species can appear before (resp., after) this consecutive set of ones, because the “state-choosing” function c can map these species to 0 (resp., 2). Hence, this problem is exactly the problem of determining whether or not a binary (0/1-) matrix has the C1P, where each species is a column in this matrix. In this case of the P-GCCC-NB Problem, if there does exist a solution P, then there is always a “state-choosing” function c' that reflects the fact that the corresponding matrix has the C1P. Therefore these cases are polynomial-time solvable. ☐

We now consider the $\{\{0\},\{1\},\{2\},\{0,2\}\}$-SB-GCCC-NB Problem, the BKW case of the SB-GCCC-NB Problem. Here, for any character $\hat{\alpha}$, a species s with $\alpha(s) = \{0,2\}$ can still appear before or after the consecutive set of ones (on this single-branch tree), however a species s with $\alpha(s) = 0$ has to appear *before* this set, while the species s with $\alpha(s) = 2$ has to appear *after* this set. So essentially, this is again the problem of determining whether a binary (0/1-) matrix has the C1P, however the matrix, in addition to containing zeros and ones, contains some special zeros, we call them 0^- (0^+), that must appear before (resp., after) the set of consecutive ones of its row, in any C1 order. Hence, this case is equivalent to deciding the following generalized version of the C1P.

Property 64 (Extended Consecutive-Ones Property (E-C1P))**.** *A matrix M on m rows and n columns with entries from set $\{0,1,0^-,0^+\}$ has the E-C1P if there is an order of the n columns such that, for any row, the set of columns that have entry* 1 *in that row are consecutive in the order, and any column that has entry* 0^- *(resp.,* 0^+*) in that row appears before (resp., after) this consecutive set of ones.*

Lemma 65. *The E-C1P can be decided in polynomial-time.*

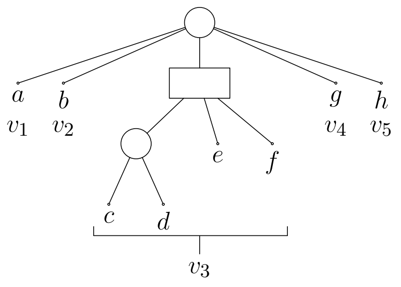
Proof. We prove this by showing that a structure that encodes all extended consecutive-ones (E-C1) orders of a matrix with entries from set $E = \{0,1,0^-,0^+\}$ can be constructed in polynomial-time. Given matrix M on m rows and n columns with entries from set E, we first construct PQ-tree PQ_M for matrix M, where we have "forgotten" the labels of the special zeros (we treat 0^- and 0^+ simply as 0). This can be done in time $O(m+n)$ [21]. It is clear that PQ_M encodes a superset of the E-C1 orders of M. We then associate to each P-node, the empty partial order on its children, and to each Q-node, the set of directions $\{\text{left},\text{right}\}$. Next, we obtain a list of order constraints imposed by the special zeros of M, by processing each pair $(0^-,1)$, $(0^+,1)$ and $(0^-,0^+)$. For instance, if column i has 0^- and j has 1 in some row r, then we add constraint $i < j$ to the list. We now update these sets that are associated with each P- and Q-node, one-by-one from the list, to incorporate these ordering constraints. The idea is that these sets will restrict the configurations each node in PQ_M can have to the set of E-C1 orders of M.

When adding constraint $i < j$ from the list to PQ_M, we find the least common ancestor $a_{i,j}$ of i and j in PQ_M, which takes $O(n)$ steps. For $a_{i,j}$, one of the two cases holds:

1. $a_{i,j}$ is a Q-node. Then we eliminate from the set at this Q-node, the direction that places j before i. If the set of directions is now empty, then the algorithm

$$\begin{array}{cccccccc}
a & b & c & d & e & f & g & h \\
\hline
0 & 0^{-} & 1 & 1 & 0 & 0^{+} & 0^{+} & 0 \\
0^{+} & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0
\end{array}$$

(a)



(b)

Figure 5.1: (a) A matrix M with entries from set $\{0,1,0^-,0^+\}$. (b) PQ-tree PQ_M for M where the labels of the special zeros (0^- and 0^+) have been "forgotten".

halts, outputting that M does not have an E-C1 order. This can be done in constant time.

2. $a_{i,j}$ is a P-node. This P-node stores some partial order on its children $\{v_1,\ldots,v_k\} = V$. First, we find the children of $a_{i,j}$: v_x and v_y such that the subtrees rooted at them contain i and j, respectively. We add the constraint $v_x < v_y$ to the existing partial order at this P-node. If this constraint is not consistent with the existing partial order then the algorithm halts, outputting that M does not have an E-C1 order. This partial order can be updated in time $O(k^2)$. Hence, this step takes time $O(k^2) \subseteq O(n^2)$.

Since there are $O(mn^2)$ order constraints, and it takes time $O(n^2)$ to process each constraint, the algorithm takes time $O(mn^4)$. Furthermore, since the tree has $O(n)$ internal nodes, and each one stores $O(mn^2)$ information, this structure is of size $O(mn^3)$. $\square$

For example, let M be the matrix with entries from set $E = \{0,1,0^-1,0^+\}$ shown in Figure 5.1a. The PQ-tree PQ_M for M, where we have "forgotten" the labels of the special zeroes is given in Figure 5.1b.

The special zeros of M then give rise to the following order constraints. In the first row of M, for example, since column b has entry 0^- and columns c and g have entries 1 and 0^+ respectively, this introduces the order constraints $b < c$

and $b < g$. The list of order constraints given by the first row of M is $\{b < c, b < d, b < f, b < g, c < f, c < g, d < f, d < g\}$, while the list given by the second row is $\{a > c, a > d, a > e\}$. The third row introduces no order constraints.

After adding the above order constraints to PQ_M of Figure 5.1b, the P-node that is the root of this tree, with children $\{v_1, \ldots, v_5\}$ stores the partial order $\{v_2 < v_3, v_2 < v_4, v_3 < v_4, v_1 > v_3\}$ on its children. The only Q-node of PQ_M has associated with it the set $\{\text{right}\}$, and the other P-node stores the empty partial order on its children $\{c, d\}$. Here, PQ_M, with the following sets (resp., partial orders) associated with its Q- (resp., P-) nodes encodes all E-C1 orders of M.

For example, in the P-node that is the root of PQ_M, order constraint $v_2 < v_3$ guarantees that $b < d$, however this is also necessary. The Q-node has associated with it set $\{\text{right}\}$ to enforce $c < f$, or even $d < f$. Note that if instead of constraint $d < f$, we had $d > f$ that this matrix would not have an E-C1 order. Finally, the P-node of PQ_M with children $\{c, d\}$ stores the empty partial order because there are no constraints involving c and d.

Theorem 66. *The* $\{\{0\},\{1\},\{2\},\{0,2\}\}$*-SB-GCCC-NB Problem is polynomial-time solvable.*

Proof. This follows from equivalence to deciding the E-C1P and Lemma 65. ☐

Since the $\{\{0\},\{1\},\{2\},\{0,2\}\}$-SB-GCCC-NB Problem is the BKW Case of the SB-GCCC-NB Problem, we have the following corollary.

Corollary 67. *The BKW Case of the SB-GCCC-NB Problem is polynomial-time solvable.*

Note that the constructed structure of Theorem 66 encodes all solutions to the problem, even if there are exponentially many of them.

5.3.3 The $\{\{1\},\{2\},\{0,2\}\}$-P-GCCC-NB Problem

We will show that if there is a solution to an instance (S,C) of the $\mathscr{Q}^*$-P-GCCC-NB Problem then there is a solution to the instance (S,C) of the $\mathscr{Q}^*$-SB-GCCC-NB Problem, and vice versa, where $\mathscr{Q}^* = \{\{1\},\{2\},\{0,2\}\}$. Since the single branch version of this problem can be solved in polynomial time by Theorem 66, it follows that also the path version is polynomial-time solvable.

Lemma 68. *An instance (S,C) of the $\{\{1\},\{2\},\{0,2\}\}$-P-GCCC-NB Problem has a solution if and only if the instance (S,C) of the $\{\{1\},\{2\},\{0,2\}\}$-SB-GCCC-NB Problem has a solution.*

Proof. Let $\mathscr{Q}^* = \{\{1\},\{2\},\{0,2\}\}$. Obviously, a solution to the instance (S,C) of the $\mathscr{Q}^*$-SB-GCCC-NB Problem is also a solution to the instance (S,C) of the $\mathscr{Q}^*$-P-GCCC-NB Problem. Now, assume that (T,c) is a solution to the instance (S,C) of the $\mathscr{Q}^*$-P-GCCC-NB Problem. Let P_1 and P_2 be two branches of T starting at the root r. Let T' be the tree obtained by attaching P_2 to the last vertex of P_1. To define the state-choosing function c' we only need to determine the values of $c'(s,\hat{\alpha})$ when $\alpha(s) = \{0,2\}$. Consider $s \in S$ and $\hat{\alpha} \in C$ such that $\alpha(s) = \{0,2\}$. If there is a species $s' \prec_T s$ such that $\alpha(s') = \{1\}$ then we set $c'(s,\hat{\alpha}) = 2$, otherwise we set $c'(s,\hat{\alpha}) = 0$. We will show that (T',c') is a solution to the instance (S,C) of the $\mathscr{Q}^*$-SB-GCCC-NB Problem.

For each $\hat{\alpha} \in C$, the set of species $S_{\hat{\alpha},\{1\}} = \{s \in S | \alpha(s) = \{1\}\}$ must induce a connected component in T. Since $\alpha(r) = 0$, this component lies entirely in P_1 or in P_2. Hence, the set $S_{\hat{\alpha},\{1\}}$ induces a connected component K in T' as well. By the definition of c', all species that lie below K in T' are assigned value 2 and all species s such that $\alpha(s) = \{0,2\}$ that lie above K in T' are assigned value 0. Hence, the only possible violation is if there is a species s such that $\alpha(s) = \{2\}$ that lies above K in T'. This species s either lies above K in T or lies in the branch that does not contain K in T. In either case, (T,c) cannot be a solution to the instance (S,C) of the $\mathscr{Q}^*$-P-GCCC-NB Problem, a contradiction. □

Corollary 69. *The $\{\{1\},\{2\},\{0,2\}\}$-P-GCCC-NB Problem is polynomial-time solvable.*

5.4 Hardness Results

We first show that the $\{\{1\},\{0,2\},\{0,1,2\}\}$-SB-GCCC-NB and $\{\{1\},\{0,2\},\{0,1,2\}\}$-P-GCCC-NB Problems are NP-complete by by reduction from the PTC Problem (Lemma 55).

Theorem 70. *The $\{\{1\},\{0,2\},\{0,1,2\}\}$-SB-GCCC-NB and $\{\{1\},\{0,2\},\{0,1,2\}\}$-P-GCCC-NB Problems are NP-complete.*

Proof. Let $\mathscr{Q}^{\triangle} = \{\{1\}, \{0,2\}, \{0,1,2\}\}$. Let S and $\{(a_i, b_i : c_i)\}_{i=1}^{k}$ be an instance of the PTC Problem. We will construct an instance of the $\mathscr{Q}^{\triangle}$-SB-GCCC-NB (resp., $\mathscr{Q}^{\triangle}$-P-GCCC-NB) Problem as follows. Let S be the set of species and $C = \{\hat{\alpha}_1, \ldots, \hat{\alpha}_k\}$ the set of characters. For every $\hat{\alpha} \in C$, we let $\alpha_i(a_i) = \alpha_i(b_i) = \{1\}$, $\alpha_i(c_i) = \{0,2\}$ and for all $s \in S \setminus \{a_i, b_i, c_i\}$, $\alpha_i(s) = \{0,1,2\}$.

We will show that the instance of the PTC Problem has a solution if and only if the constructed instance of the $\mathscr{Q}^{\triangle}$-SB-GCCC-NB (resp., $\mathscr{Q}^{\triangle}$-P-GCCC-NB) Problem has a solution. First, consider a single-branch tree (resp., path) P containing vertices S which is a solution to the constructed instance. Consider the order of elements in S as they occur on P starting from the root (resp., leaf on one branch) of P and ending with the leaf (resp., leaf on the other branch). For every $i \in \{1, \ldots, k\}$, all elements in $[a_i, b_i]$ must have state 1 for character $\hat{\alpha}_i$, hence, $c_i \notin [a_i, b_i]$, i.e., this order is a solution to the PTC Problem.

On the other hand, let order O be a solution to the PTC Problem. Consider a tree T with a single branch consisting of the all-zero root followed by vertices in S ordered by O. Note that, for every $i \in \{1, \ldots, k\}$, c_i appears either above both a_i and b_i, or below them. The state-choosing function is defined as follows. For every node in S, we choose for character $\hat{\alpha}_i$ state 0 if they are above both a_i and b_i, state 1 if they are between a_i and b_i, and state 2 otherwise. Clearly, this tree is compatible with all character trees and it is easy to see that each $c(s, \hat{\alpha}) \in \alpha(s)$, i.e., T is a solution to the $\mathscr{Q}^{\triangle}$-SB-GCCC-NB (resp., $\mathscr{Q}^{\triangle}$-P-GCCC-NB) Problem. □

Next, we show that if for $\mathscr{Q} \subseteq 2^{\{0,\ldots,m\}}$, the $\mathscr{Q}$-SB-GCCC-NB Problem is NP-complete, then the $\mathscr{Q} \cup \{\{m\}\}$-(P-)GCCC-NB Problems are NP-complete.

Theorem 71. *If for $\mathscr{Q} \subseteq 2^{\{0,\ldots,m\}}$, the $\mathscr{Q}$-SB-GCCC-NB Problem is NP-complete, then the $\mathscr{Q} \cup \{\{m\}\}$-P-GCCC-NB and $\mathscr{Q} \cup \{\{m\}\}$-GCCC-NB Problems are NP-complete.*

Proof. We will prove the claim by reduction from the $\mathscr{Q}$-SB-GCCC-NB Problem. An instance of the SB-GCCC-NB Problem can be considered as an instance of the (P-)GCCC-NB Problem, provided that we can force all species to be on a single branch. This can be done easily by adding the extra species x that has state set $\{m\}$ on all characters, and showing that all other species must have x as a descendant, which forces any solution to this instance of the (P-)GCCC-NB Problem to be a single-branch tree. We omit the details. $\square$

As a corollary, we have that the $\{\{1\},\{2\},\{0,2\},\{0,1,2\}\}$-(P-)GCCC-NB Problem is NP-complete. However, the complexity of the BKW case posed in Benham et al. [12] remains open.

Finally, we show that the $\{\{0\},\{1\},\{0,1\}\}$-P-GCCC-NB Problem is NP-complete by reduction from the LEF-PTC Problem (Lemma 59).

Theorem 72. *The $\{\{0\},\{1\},\{0,1\}\}$-P-GCCC-NB Problem is NP-complete.*

Proof. Given an instance of the LEF-PTC Problem $S = \{1,\dots,n\}$, r, and the set of k triples $(a_i, r : c_i)$, let S be the set of species, and $C = \{\hat{\alpha}_1,\dots,\hat{\alpha}_k\}$ be the set of characters. For each $\hat{\alpha}_i \in C$, we let $\alpha_i(a_i) = \{0\}$ and $\alpha_i(c_i) = \{1\}$, while for all other $s \in S \setminus \{a_i, c_i\}$ we let $\alpha_i(s) = \{0,1\}$.

Let path phylogeny T be a solution to this instance of the $\{\{0\},\{1\},\{0,1\}\}$-P-GCCC-NB Problem. Let r be the root of T, i.e., r is the all-zero vertex. Consider the ordering of elements in $S \cup \{r\}$ based on the ordering of vertices on path T starting in the leaf of one branch and ending in the leaf of the other branch. Assume the triple $(a_i, r : c_i)$ is not valid, i.e., c_i appears between a_i and r. However, this is not possible since vertex a_i is then below c_i in T and we have a transition from 1 to 0 somewhere on the path from c_i to a_i for character $\hat{\alpha}_i$. Hence, the order is a solution to the LEF-PTC Problem.

Conversely, let path/order P be a solution to the LEF-PTC Problem. Consider the path phylogeny obtained from P by rooting it at r and the state-choosing function assigning 1 to c_i and all nodes below c_i and 0 to all other nodes for character $\hat{\alpha}_i$. Clearly, this tree is compatible with all character trees. The state choosing function could only fail, if a_i is below c_i, in which case $c(a_i, \hat{\alpha}_i) = 1$, but $\alpha_i(a_i) = \{0\}$. However, this is not possible as then c_i would be between r and a_i on P which violates the constraint $(a_i, r : c_i)$. The claim follows by Lemma 59. $\square$

Note that Theorem 72 implies NP-completeness of several cases of the P-GCCC-NB Problem. In fact, any case of the problem in which set $\mathcal{Q}$ contains two distinct state singletons $\{a\}$ and $\{b\}$, and a set containing states 0, c and d such that $a \preccurlyeq_\alpha c \preccurlyeq_\alpha b$ and $b \preccurlyeq_\alpha d$ in T_α is NP-complete. For instance, for $a = c = 0$,

$b = 1$ and $d = 2$, we have that the $\{\{0\},\{1\},\{0,2\}\}$-P-GCCC-NB Problem is NP-complete ((6b) in Table 5.1).

Chapter 6

Conclusion

In this thesis, we have defined and studied several variants of the Consecutive-Ones Property (C1P) in order to model or solve several problems that arise in the reconstruction of ancestral species.

We first define in Chapter 2 a way of relaxing the C1P of binary matrices, namely the (k,δ)-C1P, to model the problem of reconstructing AGOs in the presence of small errors [27, 96]. We show that for most values of k and δ, deciding the (k,δ)-C1P is NP-complete, as well as give a tractability result for a relevant case of the (2,1)-C1P. In light of this result, and the fact that matrices from real data generally have low degree [27], in Chapter 3, we then consider the (k,δ)-C1P for matrices of bounded degree d (the (d,k,δ)-C1P). We then show that the (d,k,δ)-C1P is polynomial-time solvable when all three parameters are fixed constants, while other cases are NP-complete.

In Chapter 4, we then study a slightly different way to relax the C1P: by allowing columns to appear multiple times in an order, or the mC1P, which was first introduced in Wittler and Stoye [151]. We improve upon the hardness results of Wittler and Stoye to show that this problem is NP-complete in most cases, while also finding a tractable case of interest to handling telomeres in the reconstruction of AGOs.

Finally, in Chapter 5, we use the C1P, or more specifically, its associated data structure, the PQ-tree, to develop algorithms for several cases of the Generalized Cladistic Character Compatibility (GCCC) Problem. We now summarize our results for these four chapters in more detail, along with relevant future work.

6.1 Chapter 2: The (k,δ)-C1P

In Section 2.3 of this chapter, we show that for every $k \geq 2, \delta \geq 1, (k,\delta) \neq (2,1)$, deciding the (k,δ)-C1P is NP-complete by first showing in Subsection 2.3.1 that for every $k,\delta \geq 2$, deciding the (k,δ)-C1P is NP-complete, and then in Subsection 2.3.2 that for every $k \geq 3$, deciding the $(k,1)$-C1P is NP-complete. Note that this leaves open the case of the (2,1)-C1P, one that is interesting for real applications such as the reconstruction of AGOs [27]. In Section 2.4, we give an algorithm that, given a binary matrix M, either (a) decides if M has the (2,1)-C1P when the orders of the columns of M are restricted according to the block construction with blocks of fixed constant size of the type which the two above-mentioned constructions are, or (b) finds a proof that deciding the (2,1)-C1P is NP-complete. In fact, this algorithm is FPT in the maximum size of any block. We then show that for every $\delta \geq 1$, deciding the (∞,δ)-C1P is NP-complete in Section 2.5. We note that deciding the k-C1P, or equivalently, the (k,∞)-C1P for $k \geq 2$ has been proved NP-complete in Goldberg et al. [55]. This set of results implies that deciding the (k,δ)-C1P is NP-complete for all bounded and unbounded values of k and δ except for $(k,\delta) = (2,1)$.

The above study of this particular gapped C1P of binary matrices, namely the (k,δ)-C1P, immediately raises some open questions about closely related properties. A more restricted version would be the (k,δ)-C1P where the number of gaps in the entire matrix M is bounded by some $K \leq m(k-1)$ where m is the number of rows of M. Is such a property polynomial-time decidable? The (k,δ)-C1P is known to be NP-complete for all values of k and δ except for $(k,\delta) = (2,1)$: are there any natural parameters such that the (k,δ)-C1P is FPT? One drawback of the (k,δ)-C1P (and the k-C1P, for that matter) is that it has the rigid limit of $k-1$ gaps per row. What if we allowed allowed rows to “share a pool” of gaps, in the sense that if one row has only $k-2$ gaps, then another may have k gaps? A more general version of the (k,δ)-C1P is to bound the total number of `0`’s in the gaps in all of M. For example, given a matrix M with m rows and at most $N \leq m(k-1)\delta$ `0`’s can be in the gaps of M (an average of one gap per row when $(k,\delta) = (2,1)$, which, in a way, generalizes the (2,1)-C1P). Is this problem FPT for some natural parameter?

From a purely combinatorial point of view, there has been a renewed interest in the characterization of matrices that do not have the C1P in terms of forbidden submatrices introduced by Tucker [145]. It has recently been shown that this characterization could be used in the design of algorithms related to the C1P [18, 28, 36]. This then raises the following natural question: is there a nice characterization of matrices that do not have the (k,δ)-C1P in terms of forbidden submatrices? This is of particular interest to the open (2,1)-C1P case: if it is indeed polynomial-time decidable, trying to find a forbidden submatrix characterization may lead to an algorithm for this case. If such a characterization does not exist, given a matrix that is not C1P, can the (k,δ)-C1P be quickly determined if the set of all Tucker patterns is known?

Finally, it is also natural to ask if there exists a structure that can represent all orders that satisfy some gaps conditions related to the C1P. Such a structure exists for the C1P with no gaps: for a matrix that has the C1P, its PQ-tree represents all its C1 orders, and can be computed in linear time [21]. This has even been extended to matrices that do not have the C1P through the notion of the PQR-tree [106, 107], or the Generalized PQ-tree of McConnell [102]. Although the existence of such a structure with nice algorithmic properties is ruled out by the hardness of deciding the (k,δ)-C1P (except for maybe the (2,1)-C1P), it remains open to find classes of matrices such that testing for this property is tractable, and in such case, to represent all possible orders in a compact way. Here again, this question is motivated both by theoretical considerations (for example representing all possible layouts of a graph of bandwidth 2), but also by problems in computational genomics, such as the reconstruction of AGOs [27, 96].

Recall that, in the approach of Chauve and Tannier [27], they discard the minimum number of rows of a given matrix M using a branch-and-bound procedure, until the remaining matrix has the C1P. In Chauve and Tannier's experiments, the number of rows discarded is generally a very small fraction of the number of rows of M. This motivates the following question. Given a PQ-tree T and a set of rows R of bounded size, is there a permutation π that is generated by T to which all $r \in R$ map with at most k gaps of size δ? A variant of this would be to try and map the set R onto π while trying to minimize the number of gaps, or the number of `0`'s in the gaps (cf. a previous paragraph). In either case, perhaps there is a way to refine a PQ-tree T, as in Section 5.3 of Chapter 5, or even to *partially* refine T, as in Section 4.3 of Chapter 4 to come up with a new structure T' that encodes or (partially encodes) all permutations π that meet the gaps constraints of R. Either one would be a weak notion of a structure that encodes (k,δ)-C1 orders of a matrix that has the (k,δ)-C1P.

6.2 Chapter 3: The (d,k,δ)-C1P

In this chapter we study the (k,δ)-C1P for matrices of bounded degree d, or the (d,k,δ)-C1P. This is motivated by the fact that we have observed that matrices from experimental data of the reconstruction of AGOs [27] tend to have low degree. In Section 3.1 we show that when all three parameters are fixed constants, the (d,k,δ)-C1P is related to the classical Graph Bandwidth Problem, and can hence be solved in polynomial-time using a variant of a relatively brute-force algorithm of Saxe [135].

Then, in Section 3.2.4 we show that, for every $d > k \geq 2$, deciding the (d,k,∞)-C1P is NP-complete, by reducing from an NP-complete hypergraph covering problem which is defined in Section 3.2.1, and then is shown, in Sections 3.2.2 and 3.2.3, to be NP-complete. We comment that here we have studied the weakest formulation of the C1P with gaps: indeed, in the $(d,d-1,\infty)$-C1P case, it is required that only two of the d `1`'s in each row are adjacent in any order, while the other `1`'s can end up arbitrarily far away from this pair. It is thus surprising that deciding this property is still NP-complete for any $d \geq 3$ as implied by the general result above. This chapter closes the case of the complexity of deciding the (d,k,δ)-C1P, with the exception of the $(\infty,2,1)$-C1P case, or just the (2,1)-C1P case (cf. Chapter 2), which remains open.

We comment here that Goldberg et al. [55] poses the open question about the complexity of deciding the 2-C1P for sparse matrices (matrices where there is a limit on the number of ones per row and per column). The $(d,2,\infty)$-C1P limits the number of `1`'s per row only, that is, it is equivalent to the 2-C1P for bounded degree matrices. If we could determine the complexity of deciding the $(d,2,\infty)$-C1P for matrices with a bounded number of `1`'s per column, we could close this open question of Goldberg et al.. We do show, as a corollary of Theorem 32, that deciding the $(d,2,\infty)$-C1P is NP-complete for matrices with at most 7 1's per column, closing this open question of Goldberg et al. [55].

There are several open questions and directions we would like to follow in the future work, some of them being parallel to open questions posed in the context of just the (k,δ)-C1P. One such question: is it possible to find a nice characterization of matrices that do not have the (d,k,δ)-C1P in terms of forbidden structures, such as Tucker submatrices [145], especially for small values of d? Can the (d,k,δ)-C1P be quickly determined if the set of all Tucker patterns is known?

When all three parameters are fixed, the (d,k,δ)-C1P is related to the classical Graph Bandwidth Problem, and can hence be solved in polynomial time [29] using a variant of a relatively brute-force algorithm of Saxe [135] for deciding if a graph has bandwidth $d+(k-1)\delta-1$. This algorithm of Saxe decides if a given graph has bandwidth b in time $O(n^{b+1})$. Caprara et al. [25] provide a linear time algorithm for the special case of deciding if a graph G has bandwidth 2. In this algorithm, Caprara et al. first reduce G to a skeleton (called an auxiliary graph) that all bandwidth 2 layouts must contain. The bandwidth 2 layouts of each component of this auxiliary graph, irreducible subgraphs of G that are independent of each other, then determine the set of bandwidth 2 layouts of G.

Indeed, this auxiliary graph resembles somewhat a PQ-tree: given a C1P matrix M, and its graph G_M as defined in Section 3.1, how does the PQ-tree for M relate to the auxiliary graph for G_M? If M does not have the C1P, how does the auxiliary graph relate to the set of (d,k,δ)-C1 orders of M? How does the auxiliary graph relate to the *active regions* computed in the algorithm of Saxe? Indeed, since Caprara et al.'s algorithm is linear for graphs of bandwidth 2, perhaps improvements can be made in the general bandwidth b case (this is one of the open questions posed by Saxe). Even for small values of b, this would be useful in applications involving the reconstruction of AGOs [27]. Can the auxiliary graph be extended to a structure that all bandwidth b layouts must contain, even if computing it involves a large time overhead? This could lead to a weak notion of a PQ-tree for all (d,k,δ)-C1 orders of a matrix that has the (d,k,δ)-C1P.

Finally, assuming that k is close to d, for each row there are many orders of columns which make this row (d,k,∞)-consecutive. Hence, for a small number of rows, random instances of matrices have the (d,k,∞)-C1P almost always. Conversely, for a large number of rows, random instances of matrices that have the (d,k,∞)-C1P would have very few column orders that witness this property. We would like to investigate the ratios between the number of rows and columns for which one or the other type of instance occurs, with the goal of developing heuristics for both of these types of instances.

6.3 Chapter 4: The *m*C1P

In Section 4.1 of this chapter, we have shown that deciding the *m*C1P is NP-complete for matrices with degree at most 3 and $\mathbf{m}(s) \leq 2$ for each $s \in S$, where S is the set of columns of M. In Section 4.2 we then show that the two restricted variants of the *m*C1P given in Wittler and Stoye [151], namely the *m*C1P(fr) and the *m*C1P(ne) are NP-complete for matrices with degree at most 3 (6 for the *m*C1P(fr) case) and $\mathbf{m}(s) \leq 2$ for each $s \in S$, where S is the set of columns of M. In Section 4.3, we have shown that, given a matrix M and a multiplicity vector $\mathbf{m}$ such that (1) M has matched multirows, and (2) each row contains either (i) at most one entry 1 in multicolumns, or (ii) two entries 1 in multicolumns and no other entries, that deciding if M has the *m*C1P for $\mathbf{m}$ can be done in polynomial time and space (cf. Theorem 51).

In light of the result of Section 4.3, we extend the domain of tractable instances of deciding the *m*C1P for binary matrices. This approach relies on previously used techniques to decide the C1P and simpler instances of the *m*C1P, and answers a natural problem in reconstructing ancestral gene orders. Several questions remain open. Naturally, one can ask to relax the condition that M has matched multirows, which is crucial in our proofs. It seems however that the problem becomes hard in this case, and some less rigid constraints on M would then have to be introduced to recover tractability. Also it is open to exhibit an extension of the notion of the PQ-tree that could encode all *m*C1P orders of a binary matrix that satisfies this property. Even for the case of a matrix with matched multirows, our techniques lead to a data structure which only captures the consecutivity requirement (cf. Section 4.3) but not the multiplicity requirement. From an algorithmic complexity point of view, our algorithm has an $O(mn)$ time complexity, and it remains open to see if this case can be solved in $O(m+n+\ell)$ time, where ℓ is the number of `1`'s in the entire matrix M.

The problem of covering hypergraphs with a collection of paths played a key role in the hardness results of Chapter 3, and, with slightly different conditions on this collection of paths, played a key role in the hardness results of this chapter. Other variants of hypergraph covering were also used to show both hardness and algorithmic results for the haplotyping problem via galled tree networks [59–61]. Perhaps considering other conditions on the covering could give rise to other new and interesting problems. In fact, one could do a systematic study of the covering of hypergraphs with graphs to see which conditions (on both hypergraph and graph) lead to interesting results.

6.4 Chapter 5: The GCCC Problem

In Section 5.2 we show that the PTC and the LEF-PTC Problems are NP-complete, while the OEF-TO Problem is polynomial-time solvable, and the REF-PTC Problem always has a solution. In Section 5.3, we present some tractable cases of the GCCC Problem, while in Section 5.4 we present some hardness results.

Here, we have characterized the complexity of cases of the $\mathscr{Q}$-SB-GCCC-NB and $\mathscr{Q}$-P-GCCC-NB Problems for $\mathscr{Q} \subseteq \{\{0\},\{1\},\{2\},\{0,2\},\{0,1,2\}\}$. This leaves open, however, some interesting cases of the GCCC-NB Problem. Here we show that when $\mathscr{Q}' = \{\{1\},\{0,2\}\}$, the input corresponds to a binary matrix M, hence the $\mathscr{Q}'$-SB-GCCC-NB Problem is equivalent to the C1P Problem. That is, the $\mathscr{Q}'$-SB-GCCC-NB (resp., $\mathscr{Q}'$-GCCC-NB) Problem is to find a single-branch path (resp., tree) with vertex set containing the columns of M (and possibly other columns) such that for each row of M, the set of vertices labelled `1` by this row forms a connected subpath (resp., subtree), i.e., M has the C1P (resp., a "connected-ones property" of trees). Note also that for a tree to have this connected-ones property, that sets of vertices labelled `0` by any row must form at most 2 connected subtrees, so that this tree can be contracted to $0 \rightarrow 1 \rightarrow 2$ for each row (this is automatically enforced in the case of the C1P, since the set of vertices labelled by `1` in each row is a path). If we can determine in polynomial-time that this connected-ones property holds (like we can for the C1P), it might provide an answer to the

BKW Case. Preliminary study has shown that the set of such matrices corresponds to a special class of chordal graphs: deeper study into this connection could be useful.

Finally, it would be interesting to systematically study these problems for all subsets of $2^{\{0,1,2\}}$, as it would complete the study for all possible inputs to the GCCC-NB Problem when character trees are $0 \to 1 \to 2$.

Bibliography

[1] Z. Adam, M. Turmel, C. Lemieux, and D. Sankoff. Common intervals and symmetric difference in a model-free phlogenomics, with an application to streptophyte evolution. *Journal of Computational Biology*, 14:436–445, 2007. $\rightarrow$ pages 10, 13, 16, 17

[2] R. Agarwala and D. Fernandez-Baca. A polynomial-time algorithm for the perfect phylogeny problem when the number of character states is fixed. *SIAM Journal on Computing*, 26(6):1216–1224, 1994. $\rightarrow$ pages 30, 102

[3] M. Alekseyev and P. Pevzner. Colored de bruijn graphs and the genome halving problem. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4:98–107, 2007. $\rightarrow$ pages 27

[4] F. Alizadeh, R. Karp, L. Newberg, and D. Weisser. Physical mapping of chromosomes: A combinatorial problem in molecular biology. In *Proceedings of the 4th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1991. $\rightarrow$ pages 7

[5] F. Alizadeh, R. Karp, D. Weisser, and G. Zweig. Physical mapping of chromosomes using unique probes. *J. Comput. Biol.*, 2(2):159–184, 1995. $\rightarrow$ pages 7, 11

[6] E. Althaus, S. Canzar, M. Emmett, A. Karrenbauer, A. Marshall, A. Meyer-Baese, and H.-M. Zhang. Computing H/D-exchange speeds of single residues from data of peptic fragments. In *Proceedings of the 23rd ACM Symposium on Applied Computing (SAC 2008)*, pages 1273–1277. ACM Press, 2008. $\rightarrow$ pages 6

[7] J. Atkins and M. Middendorf. On physical mapping and the consecutive ones property for sparse matrices. *Discrete Applied Mathematics*, 71(1-3): 23–40, 1996. $\rightarrow$ pages 7, 21

[8] J. Atkins, E. Boman, and B. Hendrickson. A spectral algorithm for seriation and the consecutive ones problem. *SIAM Journal on Computing*, 28(1):297–310, 1998. → pages 7, 21

[9] M. Beal, A. Bergeron, S. Corteel, and M. Raffinot. An algorithmic view of gene teams. *Theoretical Computer Science*, 320:395–418, 2004. → pages 17

[10] M. Belcaid, A. Bergeron, A. Chateau, C. Chauve, Y. Gingras, G. Poisson, and M. Vendette. Exploring genome rearrangments using virtual hybridization. In D. Sankoff, L. Wang, and F. Chin, editors, *Proceedings of the 5th Asia-Pacific Bioinformatics Conference (APBC)*, volume 5 of *Advances in Bioinformatics and Computational Biology*, pages 205–214. Imperial College Press, 2007. → pages 17

[11] C. Benham, S. Kannan, M. Paterson, and T. Warnow. Hen's teeth and whale's feet: Geralized characters and their compatibility. *Computational Biology*, 2(4):515–525, 1995. → pages 30, 31, 100, 101, 102, 103, 108

[12] C. Benham, S. Kannan, and T. Warnow. Of chicken teeth and mouse eyes, or generalized character compatibility. *Combinatorial Pattern Matching*, pages 17–26, 1995. → pages iii, v, 1, 30, 31, 100, 101, 102, 103, 116

[13] S. Benzer. On the topology of genetic fine structure. In *Proceedings of the National Academy of Sciences*, volume 45, pages 1607–1620, U.S.A., 1959. → pages 3, 7

[14] A. Bergeron and J. Stoye. On the similarity of sets of permutations and its applications to genome comparison. *Journal of Computational Biology*, 13 (7):1340–1354, 2006. → pages 78

[15] A. Bergeron, M. Blanchette, A. Chateau, and C. Chauve. Reconstructing ancestral genomes using conserved intervals. In I. Jonassen and J. Kim, editors, *Proceedings of the 4th International Workshop on Algorithms in Bioinformatics*, volume 3240 of *Lecture Note in Bioinformatics*, pages 14–25, 2004. → pages 16, 27, 28, 78

[16] A. Bergeron, Y. Gingras, and C. Chauve. *Bioinformatics Algorithms: Techniques and Applications*, chapter 8 Formal Models of Gene Clusters, pages 177–202. 2008. → pages 21, 27

[17] G. Blin, D. Faye, and J. Stoye. Finding nested common intervals efficiently. *Journal of Computational Biology*, 17(9):1183–1194, 2010. → pages 82

[18] G. Blin, R. Rizzi, and S. Vialette. A faster algorithm for finding minimum tucker submatrices. In F. Ferreira, B. Löwe, E. Mayordomo, and L. Gomes, editors, *In Proceedings of Program, Proofs, Processes, the Sixth Conference on Computability in Europe (CiE)*, volume 6158 of *LNCS*, pages 69–77. Springer, 2010. → pages 3, 5, 120

[19] S. Böcker, K. Jahn, J. Mixtacki, and J. Stoye. Computation of median gene clusters. *Journal of Computational Biology*, 16(8):1085–1099, 2009. → pages 78

[20] H. Bodlaender, M. Fellows, and T. Warnow. Two strikes against perfect phylogeny. In *ICALP*, pages 273–283, 1992. → pages 30, 102

[21] K. S. Booth and G. S. Lueker. Testing for the consecutive ones property of, interval graphs, and graph planarity using PQ-tree algorithms. *Journal of Computer and System Sciences*, 13(3):335–379, 1976. → pages iii, v, 3, 4, 6, 18, 88, 101, 110, 111, 120

[22] G. Bourque and P. Pevzner. Genome-scale evolution: reconstructing gene orders in the ancestral species. *Genome Research*, 12:26–36, 2002. → pages 11, 12

[23] G. Bourque, P. Pevzner, and G. Tesler. Reconstructing the genomic architecture of ancestral mammals: lessons from human, mouse and rat genomes. *Genome Research*, 14:507–516, 2004. → pages

[24] G. Bourque, E. Zdobnov, P. Bork, P. Pevzner, and G. Tesler. Comparative architectures of mammalian and chicken genomes reveal highly rates of genomic rearrangements across different lineages. *Genome Research*, 15: 98–110, 2005. → pages 11

[25] A. Caprara, F. Malucelli, and D. Petrolani. On bandwidth-2 graphs. *Discrete Applied Mathematics*, 34:477–495, 2002. → pages 25, 122

[26] B. Chang, K. Jönsson, M. Kazmi, M. Donoghue, and T. Sakmar. Recreating a functional ancestral archosaur visual pigment. *Molecular Biology and Evolution*, 19(9):1483–1489, 2002. → pages 10

[27] C. Chauve and E. Tannier. A methodological framework for the reconstruction of contiguous regions of ancestral genomes and its application to mammalian genomes. *PLoS Comput. Biol.*, 4(e1000234), 2008. → pages 1, 9, 17, 18, 19, 20, 21, 22, 24, 25, 26, 27, 29, 118, 119, 120, 121, 122

[28] C. Chauve, U.-W. Haus, T. Stephen, and V. You. Minimal conflicting sets for the consecutive-ones property in ancestral genome reconstruction. In *Proc. of RECOMB-CG*, volume 5817 of *LNBI*, pages 48–58, 2009. → pages 120

[29] C. Chauve, J. Maňuch, and M. Patterson. On the gapped consecutive-ones property. In *Proc. of European Conference on Combinatorics, Graph Theory and Applications (EUROCOMB)*, volume 34 of *ENDM*, pages 121–125, 2009. → pages iv, 122

[30] C. Chauve, U.-W. Haus, T. Stephen, and V. You. Minimal conflicting sets for the consecutive-ones property in ancestral genome reconstruction. *Journal of Computational Biology*, 17(9):1167–1181, 2010. → pages 3

[31] C. Chauve, J. Maňuch, M. Patterson, and R. Wittler. Tractability results for the consecutive-ones property with multiplicity. In *Proceedings of the 22nd Annual Symposium on Combinatorial Pattern Matching (CPM)*, volume 6661 of *Lecture Notes in Computer Science*, pages 90–103. Springer, 2011. → pages v

[32] T. Christof, M. Jnger, J. Kececioglu, P. Mutzel, and G. Reinelt. A branch-and-cut approach to physical mapping of chromosomes by unique end-probes. *Journal of Computational Biology*, 4:433–447, 1997. → pages 7

[33] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2001. → pages 7

[34] D. Corneil, S. Olariu, and L. Stewart. The ultimate interval graph recognition algorithm? In *Proceedings of the 9th Symposium on Discrete Algorithms (SODA)*, pages 175–180. ACM/SIAM, 1998. → pages 6

[35] T. Dandekar, B. Snel, M. Huynen, and P. Bork. Conservation of gene order: A fingerprint of proteins that physically interact. *Trends in Biochemical Sciences*, 23(9):324–328, 1998. → pages 8

[36] M. Dom. *Recognition, generation, and application of binary matrices with the consecutive-ones property*. PhD thesis, Institut für Informatik, Friedrich-Schiller-Universität, Jena, 2008. → pages 6, 7, 21, 24, 120

[37] M. Dom. Algorithmic aspects of the consecutive-ones property. *Bullentin of the European Association of Theoretical Computer Science (EATCS)*, 98 (2759), 2009. → pages 3, 5, 6, 24

[38] M. Dom, J. Guo, and R. Niedermeier. Approximability and parameterized complexity of the consecutive ones submatrix problems. In *Proceedings of the 4th International Conference on the Theory and Applications of Models of Computation (TAMC)*, volume 4484 of *LNCS*, pages 680–691. Springer-Verlag, 2007. → pages 21

[39] M. Dom, J. Guo, and R. Niedermeier. Approximation and fixed-parameter algorithms for consecutive ones submatrix problems. *Journal of Computer and System Sciences*, 2009. → pages 3

[40] D. Durand and D. Sankoff. Tests for gene clustering. pages 144–154. ACM Press, 2002. → pages 78

[41] N. El-Mabrouk and D. Sankoff. The reconstruction of doubled genomes. *SIAM Journal of Computing*, 32:754–792, 2003. → pages 27

[42] G. Estabroowk and F. McMorris. When is one estimate of evolutionary relationships a refinement of the another? *J. Math. Biosci.*, 10:327–373, 1980. → pages 30

[43] G. Even, R. Levi, D. Rawitz, B. Schieber, S. Shahar, and M. Sviridenko. Algorithms for capacitated rectangle stabbing and lot sizing with joint set-up costs. *ACM Transactions on Algorithms*, 4(3), 2008. Article 34. → pages 7

[44] T. Faraut. Adressing chromosome evolution in the whole-genome sequence era. *Chromosome Research*, 16:5–16, 2008. → pages 12

[45] J. Felsenstein. *Inferring Phylogenies*. Sinauer Associates, 2003. → pages 10, 30

[46] A. Ferreria and S. Song. Achieving optimality for gate matrix layout and PLA folding: a graph theoretic approach. In I. Simon, editor, *Proceedings of the 1st Latin American Symposium on Theoretical Informatics (LATIN)*, volume 583 of *LNCS*, pages 139–153, São Paulo, Brasil, 1992. Springer-Verlag. → pages 6

[47] L. Figuera, M. Pandolfo, P. Dunne, J. Cantu, and P. Patel. Mapping the congenital generalized hypertrichosis locus to chromosome Xq24-q27.1. *Nature (London)*, 10:202–207, 1995. → pages 30

[48] W. Fitch. Towards defining the course of evolution: Minimum change for a specific tree topology. *Systematic Zoology*, 20:406–416, 1971. → pages 12, 16

[49] R. Friedman and A. Hughes. Gene duplication and the structure of eukaryotic genomes. *Genome Research*, 11(3):373–381, 2001. → pages 78

[50] L. Froenicke, J. Wienberg, G. Stone, L. Adams, and R. Stanyon. Towards the delineation of the ancestral eutherian genome organization: comparitive genome maps of human and the african elephant (loxodonta africana) generated by chromosome painting. In *Proceedings of the Royal Society B Biological Sciences*, volume 270, pages 1331–1340, 2003. → pages 11, 19

[51] L. Froenicke, M. Caldés, A. Graphodatsky, S. Müller, L. Lyons, T. Robinson, M. Volleth, F. Yang, and J. Wienberg. Are molecular cytogenetics and bioinformatics suggesting diverging models of ancestral mammalian genomes? *Genome Research*, 16:306–310, 2006. → pages 12, 13

[52] D. Fulkerson and O. Gross. Incidence matrices and interval graphs. *Pacific Journal of Mathematics*, 15:835–855, 1965. → pages ii, 2, 3, 4, 6, 7

[53] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979. → pages 105

[54] S. Ghosh. File organization: the consecutive retrieval property. *Communcations of the ACM*, 15(9):802–808, 1972. → pages 6

[55] P. Goldberg, M. Golumbic, H. Kaplan, and R. Shamir. Four strikes against physical mapping of DNA. *J. Comput. Biol.*, 2(1):139–152, 1995. → pages ii, 7, 21, 22, 23, 26, 35, 36, 71, 72, 119, 121, 122

[56] J. Gordon, K. Byrne, and K. Wolfe. Additions, losses, and rearrangements on the evolutionary route from a reconstructed ancestor to the modern Saccharomyces cerevisiae genome. *PLoS Genetics*, 5(5), 2009. → pages 10

[57] J. Gramm, T. Nierhoff, R. Sharan, and T. Tantau. Haplotyping with missing data via perfect path phylogenies. *Discrete Applied Mathematics*, 155: 788–805, 2007. → pages 101

[58] D. Greenberg and S. Istrail. Physical mapping by STS hybridization: algorithmic strategies and the challenge of software evaluation. *Journal of Computational Biology*, 2(2):219–274, Summer 1995. → pages 7

[59] A. Gupta, J. Maňuch, L. Stacho, and X. Zhao. Algorithm for haplotype inferring via galled-tree networks with simple galls. In *Proc. of Int. Symposium on Bioinformatics Research and Applications (ISBRA)*, volume 4463 of *LNBI*, pages 121–132, 2007. → pages 62, 124

[60] A. Gupta, J. Maňuch, L. Stacho, and X. Zhao. Haplotype inferring via galled-tree networks is NP-complete. In *Proc. of Annual Int. Computing and Combinatorics Conference (COCOON)*, volume 5092 of *LNCS*, pages 287–298, 2008. → pages 62

[61] A. Gupta, J. Maňuch, L. Stacho, and X. Zhao. Haplotype inferring via galled-tree networks using a hypergraph covering problem for special genotype matrices. *Discr. Appl. Math.*, 157(10):2310–2324, 2009. → pages 62, 124

[62] D. Gusfield. Efficient algorithms for inferring evolutionary trees. *Networks*, 21:19–28, 1991. → pages 30

[63] D. Gusfield. The multi-state perfect phylogeny problem with missing and removable data: Solutions via integer-programming and chordal graph theory. In *Proc. of RECOMB 2009*, volume 5541 of *LNCS*, pages 294–310, 2009. → pages 31

[64] M. Habib, R. M. McConnell, C. Paul, and L. Viennot. Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. *Theoretical Computer Science*, 234(1-2):59–84, 2000. → pages 5, 6

[65] S. Haddadi. A note on the NP-hardness of the consecutive block minimization problem. *International Transactions on Operational Research*, 9(6):775–777, 2002. → pages 24

[66] M. T. Hajiaghayi and Y. Ganjali. A note on the consecutive ones submatrix problem. *Information Processing Letters*, 83(3):163–166, 2002. → pages 21

[67] R. Hassin and M. Megiddo. Approximation algorithms for hitting objects with straight lines. *Discrete Applied Mathematics*, 30:29–42, 1991. → pages 6

[68] X. He and M. Goldwasser. Identifying conserved gene clusters in the presence of homology families. *Journal of Computational Biology*, 12(6): 638–656, 2005. → pages 78

[69] R. Hoberman and D. Durand. The incompatible desiderata of gene cluster properties. In *Proceedings of RECOMB Comparitive Genomics*, volume 3678 of *Lecture Notes in Bioinformatics*, pages 73–87. Springer Verlag, 2005. → pages 78, 82

[70] D. Hochbaum and A. Levin. Cyclical scheduling and multi-shift scheduling: Complexity and approximation algorithms. *Discrete Optimization*, 3(4):327–340, 2006. → pages 6

[71] D. Hochbaum and P. Tucker. Minimax problems with bitonic matrices. *Networks*, 40(3):113–124, 2002. → pages 6

[72] F. Hole and M. Shaw. *Computer analysis of chronological seriation*, volume 53. 1967. → pages 6

[73] W.-L. Hsu. A simple test for the consecutive ones property. In T. Ibaraki, Y. Inagaki, and K. Iwama, editors, *ISAAC*, volume 650 of *LNCS*, pages 459–468, 1992. → pages 4

[74] W.-L. Hsu. A simple test for interval graphs. In *Proceedings of the 18th International Workshop on Graph-Theoretic Concepts in Computer Science (WG)*, volume 657 of *LNCS*, pages 11–16. Springer, 1992. → pages 6

[75] W.-L. Hsu. On physical mapping algorithms – an error-tolerant test for the consecutive-ones property. volume 1276 of *Lecture Notes in Computer Science*, pages 242–250. Springer, 1997. → pages 24

[76] W.-L. Hsu. A simple test for the consecutive ones property. *Journal of Algorithms*, 43(1):1–16, 2002. → pages 4

[77] W.-L. Hsu and T.-H. Ma. Fast and simple algorithms for recognizing chordal comparability graphs and interval graphs. *SIAM Journal on Computing*, 28(3):1004–1020, 1999. → pages 6

[78] W.-L. Hsu and R. McConnell. PC tress and circular-ones arragements. *Theoretical Computer Science*, 296(1):99–116, 2003. → pages 5

[79] C. Janis. The sabertooth's repeat performances. *Natural History*, 103: 78–82, 1994. → pages 30

[80] T. Jermann, J. Opitz, J. Stackhouse, and S. Benner. Reconstructing the evolutionary history of the artiodactyl ribonuclease superfamily. *Nature*, 374(6517):57–59, 1995. → pages 10

[81] S. Jinks-Robertson and T. Petes. Chromosomal translocations generated by high-frequency meiotics recombination between repeated yeast genes. *Genetics*, 114(3):731–752, 1986. → pages 20

[82] S. Kannan and T. Warnow. Inferring evolutionary history from DNA sequences. *SIAM Journal on Computing*, 23(4):713–737, 1994. → pages 30

[83] S. Kannan and T. Warnow. A fast algorithm for the computation and enumeration of perfect phylogenies. In *SODA*, pages 595–603, 1995. → pages 30

[84] D. Kendall. Incidence matrices, interval graphs and seriation in archaeology. *Pacific Journal of Mathematics*, 2(28):219–274, 1995. → pages 2, 6

[85] W. Kent, R. Baertsch, A. Hinrichs, W. Miller, and D. Haussler. Evolutions's cauldron: Duplication, deletion, and rearrangement in the mouse and human genomes. In *Proceedings of the National Academy of Sciences USA*, volume 100, pages 11484–11489. → pages xii, 12, 14

[86] E. Kollar and C. Fisher. Tooth induction in chick epithelium: Expression of quiescent genes for enamel synthesis. *Science*, 207:993–995, 1980. → pages 30

[87] N. Korte and R. Möhring. An incremental linear-time algorithm for recognizing interval graphs. *SIAM Journal on Computing*, 18(1):68–81, 1989. → pages 4, 6

[88] L. Kou. Polynomial complete consecutive information retrieval problems. *SIAM Journal on Computing*, 6(1):67–75, 1977. → pages 6

[89] S. Kovaleva and F. Spieksma. Approximation of a geometric set covering problem. In *Proceedings of the 12th International Society for Analysis, Applications and Computation (ISAAC)*, volume 2223 of *LNCS*, pages 493–501. Springer, 2001. → pages 6

[90] D. Kratsch, R. McConnell, K. Mehlhorn, and J. Spinrad. Certifying algorithms for recognizing interval graphs and permutation graphs. *SIAM Journal on Computing*, 36(2):326–353, 2006. → pages 5, 6

[91] G. Landau, L. Parida, and O. Weimann. Gene proximity analysis across whole genomes via PQ trees. *Journal of Computational Biology*, 12(10): 1289–1306, 2005. → pages 16, 17, 26

[92] C. Lekkerkerker and J. Boland. Representation of a finite graph by a set of intervals on the real line. *Fundamentals of Mathematics*, 51:45–64, 1962. → pages 7

[93] H. Lewin, D. Larkin, J. Pontius, and S. O'Brien. Every genome sequence needs a good map. *Genome Research*, 19:1925–1928, 2009. → pages 7

[94] W.-F. Lu and W.-L. Hsu. A test for the consecutive ones property on noisy data – application to physical mapping and sequence assembly. *Journal of Computational Biology*, 10(5):709–735, 2003. → pages 7, 21, 24

[95] N. Luc, J. Risler, A. Bergeron, and M. Raffinot. Gene teams: A new formalization of gene clusters for comparative genomics. *Computational Biology and Chemistry*, 27:59–67, 2003. → pages 17

[96] J. Ma, L. Zhang, B. Suh, B. Raney, R. Burhans, W. Kent, M. Blanchette, D. Haussler, and W. Miller. Reconstructing contiguous regions of an ancestral genome. *Genome Research*, 16(12):1557–1565, 2006. → pages xii, xvii, 9, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 118, 120

[97] J. Ma, A. Ratan, B. Raney, B. Suh, L. Zhang, W. Miller, and D. Haussler. DUPCAR: Reconstructing contiguous ancestral regions with duplications. *Journal of Computational Biology*, 15:1007–1027, 2008. → pages 27

[98] J. Maňuch, M. Patterson, and A. Gupta. On the generalised character compatibility problem for non-branching character trees. In H. Q. Ngo, editor, *Proceedings of the 15th Annual International Conference on Computing and Combinatorics (COCOON)*, pages 268–276, 2009. → pages 103

[99] J. Maňuch and M. Patterson. The complexity of the gapped consecutive-ones property problem for matrices of bounded maximum degree. In *Proceedings of the 8th Annual RECOMB Satellite Workshop on Comparative Genomics (RECOMB-CG)*, volume 6398 of *Lecture Notes in Bioinformatics*, pages 278–289. Springer, 2010. → pages v

[100] J. Maňuch and M. 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. → pages v

[101] J. Maňuch, M. Patterson, and C. Chauve. Hardness results for the gapped consecutive-ones property. *Discrete Applied Mathematics*, 2011. to appear. → pages iv

[102] R. McConnell. A certifying algorithm for the consecutive-ones property. In *Proc. of the Fifth Annual Symposium on Discrete Algorithms (SODA)*, pages 761–770. SIAM, 2004. → pages 5, 18, 88, 94, 120

[103] F. McMorris, T. Warnow, and T. Wimer. Triangulating vertex colored graphs. *SIAM Journal of Discrete Mathematics*, 7(2):296–306, 1994. → pages 30

[104] S. Mecke and D. Wagner. Solving geometric covering problems by data reduction. In *Proceedings of the 12th European Symposium on Algorithms (ESA)*, volume 3221 of *LNCS*, pages 760–771. Springer, 2004. → pages 6, 7, 24

[105] S. Mecke, A. Schöbel, and D. Wagner. Station location – complexity and approximation. In *Proceedings of the 5th Algorithmic Methods and Models for Optimization of Railways (ATMOS)*, IBFI. Dagstuhl, Germany, 2005. → pages 6, 7, 24

[106] J. Meidanis, O. Porto, and G. Telles. On the consecutive ones property. *Discrete Applied Mathematics*, 88(1-3):325–354, 1998. → pages v, 4, 18, 88, 101, 120

[107] J. Meidanis, O. Porto, and G. P. Telles. On the consecutive ones property. *Discrete Applied Mathematics*, 155:788–805, 2007. → pages 120

[108] T. Mizukami, W. Chang, I. Garkavtsev, N. Kaplan, D. Lombardi, T. Matsumoto, O. Niwa, A. K. andM. Yanagida, T. Marr, and D. Beach. A 13kb resolution cosmid map of the 14mb fission yeast genome by nonrandom sequence-tagged site mapping. *Cell*, 73:121–132, 1993. → pages xvii, 8, 11

[109] T. Morgan. *The Theory of the Gene*. Yale University Press, New Haven, 1926. → pages 7

[110] M. Muffato and H. Roest-Crollius. Paleogenomics, or the recovery of lost genomes from the mist of times. *Bioessays*, 30:122–134, 2008. → pages 12

[111] W. Murphy, D. Larkin, A. E. van der Wind, G. Bourque, G. Tesler, L. Auvil, J. Beever, B. Chowdhary, F. Galibert, L. Gatzke, C. Hitte, S. Meyers, D. Milan, E. Ostrander, G. Pape, H. Parker, T. Raudsepp, M. Rogatcheva, L. Schook, L. Skow, M. Welge, J. Womack, S. O'Brien, P. Pevzner, and H. Lewin. Dynamics of mammalian chromosome evolution inferred from multispecies comparative maps. *Science*, 309(5734): 613–617, July 2005. → pages 11, 12, 17

[112] G. Nemhauser and L. Wolsey. *Integer and Combinatorial Optimization*. Discrete Mathematics and Optimization. Wiley, 1988. → pages 7

[113] M. Novick. Generalized pq-trees. Technical Report 89-1074, Cornell University, 1989. → pages 4

[114] J. Opatrny. Total ordering problem. *SIAM Journal of Computing*, 8(1): 111–114, 1979. → pages 104

[115] M. Oswald and G. Reinelt. Polyhedral aspects of the consecutive ones problem. In *Proceedings of the 6th Annual International Computing and Combinatorics Conference (COCOON)*, volume 1858 of *LNCS*, pages 373–382. Springer, 2000. → pages 6

[116] M. Oswald and G. Reinelt. Constructing new facets of the consecutive ones polytope. In *Proceedings of the 5th International Workshop on Combinatorial Optimization–"Eureka, You Shrink!"*, volume 2570 of *LNCS*, pages 147–157. Springer, 2003. → pages 6

[117] A. Ouangraoua, F. Boyer, A. McPherson, E. Tannier, and C. Chauve. Prediction of contiguous regions in the amniote ancestral genome. In *Proceedings of the International Symposium on Bioinformatics Research and Applications (ISBRA)*, Lecture Notes in Computer Science, pages 173–185. Springer, 2009. → pages 22

[118] R. Overbeek, M. Fonstein, M. D'Souza, G. Pusch, and M. Maltsev. The use of gene clusters to infer functional coupling. In *Proceedings of the National Academy of Sciences USA*, volume 96, pages 2896–2901, 1999. → pages 8

[119] M. Palazzolo, S. Sawyer, C. Martin, D. Smoller, and D. Hartl. Optimized strategies for sequence-tagged-site selection in genome mapping. In *Proceedings of the National Academy of Sciences*, volume 88, pages 8034–8038, U.S.A., 1991. → pages xvii, 8, 11

[120] C. Papadimitriou. *Computational Complexity*. Addison Wesley, 1994. → pages 46, 47, 62, 74

[121] L. Parida. Using PQ structures for genomic rearrangement phylogeny. *Journal of Computational Biology*, 13(10):1685–1700, 2006. → pages 17

[122] S. Pasek, A. Bergeron, J. Risler, A. Louis, E. Ollivier, and M. Raffinot. Identification of genomic features using microsyntenies of domains: domain teams. *Genome Research*, 15(6):867–874, 2005. → pages 21, 22, 23, 78

[123] I. Pe'er, T. Pupko, R. Shamir, and R. Sharan. Incomplete directed perfect phylogeny. *SIAM J. Computing*, 33:590–607, 2004. → pages 100

[124] G. Pontecorvo. *Trends in Genetic Analysis*. Columbia University Press, New York, 1958. → pages 7

[125] S. Rahmann and G. Klau. Integer linear programs for discovering approximate gene clusters. In *Proceedings of the Workshop on Algorithms in Bioinformatics (WABI)*, volume 4175 of *Lecture Notes in Bioinformatics*, pages 298–306. Springer Verlag, 2006. → pages 78

[126] V. Rascol, P. Pontarotti, and A. Levasseur. Ancestral animal genomes reconstruction. *Current Opinions in Immunology*, 19(5):542–546, 2007. → pages 12

[127] F. Richard, M. Lombard, and B. Dutrillaux. Reconstruction of the ancestral karyotype of eutherian mammals. *Chromosome Research*, 11:605–618, 2002. → pages 11, 16, 18, 19

[128] C. Richardson and M. Jasin. Frequent chromosomal translocations induced by DNA double-strand breaks. *Nature*, 405:697–700, 2000. → pages 20

[129] W. Robinson. A method for chronologically ordering archaeological deposits. *American Antiquity*, 16:293–301, 1951. → pages 2, 6

[130] M. Rocchi, N. Archidiacono, and R. Stanyon. Ancestral genome reconstruction: an integrated, multi-disciplinary approach is needed. *Genome Research*, 16:1441–1444, 2006. → pages 12, 13

[131] A. Ruf and A. Schöbel. Set covering with almost consecutive ones property. *Discrete Optimization*, 1(2):215–228, 2004. → pages 6, 7, 24

[132] H. Ryser. Combinatorial configurations. *SIAM Journal on Applied Mathematics*, 17(3):593–602, 1969. → pages 3

[133] M. Sadqi, E. de Alba, R. Pérez-Jiménez, J. Sanchez-Ruiz, and B. M. noz. A designed protein as experimental model of primordial folding. In *Proceedings of the National Academy of Sciences USA*, volume 106, pages 4127–4132, 2009. → pages 10

[134] D. Sankoff, C. Zheng, and Q. Zhu. Polyploids, genome halving and phylogeny. *Bioinformatics*, 23:433–439, 2007. → pages 27

[135] J. B. Saxe. Dynamic-programming algorithms for recognizing small-bandwidth graphs in polynomial time. *SIAM J. on Alg. and Discr. Meth.*, 1(4):363–369, 1980. → pages 25, 54, 55, 56, 57, 58, 59, 60, 121, 122

[136] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, 1986. → pages 6

[137] C. Semple and M. Steel. *Phylogenetics*. Oxford Lecture Series in Mathematics and its Applications. Oxford University Press, 2003. → pages iii, xviii

[138] M. Steel. The complexity of reconstructing trees from qualitative characters and subtrees. *Journal of Classification*, 9:91–116, 1992. → pages 30, 103

[139] J. Stoye and R. Wittler. A unified approach for reconstructing ancient gene clusters. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 2009. → pages 27

[140] A. Sturtevant and T. Dobzhansky. Inversions in the third chromosome of wild races of drosophilia pseudoobsura, and their use in the study of the history of the species. In *Proceedings of the National Academy of Sciences*, number 22, pages 448–450, 1936. → pages 11

[141] M. Svartman, G. Stone, J. Page, and R. Stanyon. A chromosome painting test of the basal eutherian karyotype. *Chromosome Research*, 12:45–53, 2004. → pages 11, 19

[142] M. Svartman, G. Stone, and R. Stanyon. The ancestral eutherian karyotype is present in xenarthra. *PLoS Genetics 2*, (e109), 2006. → pages 11, 19

[143] J. Tang and L. Zhang. The consecutive ones submatrix problem for sparse matrices. *Algorithmica*, 48:287–299, 2007. → pages 21

[144] J. Trowsdale. Genomic structure and function in the MHC. *Trends Genet.*, 9:117–122, 1993. → pages 30

[145] A. C. Tucker. A structure theorem for the consecutive 1's property. *J. of Comb. Theory, Series B*, 12:153–162, 1972. → pages x, 2, 3, 6, 120, 122

[146] Y. van de Peer. Computational approaches to unveiling ancient genome duplications. *Nature Reviews*, 5:752–763, 2004. → pages 27

[147] A. Veinott and H. Wagner. Optimal capacity scheduling. *Operational Research*, 10:518–547, 1962. → pages 6, 7

[148] T. Warnow. Tree compatibility and inferring evolutionary history. *J. Algorithms*, 16:388–407, 1994. → pages 30

[149] S. Weis and R. Reischuk. The complexity of physical mapping with strict chimerism. In *Proceedings of the Sixth Annual COCOON*, volume 1858 of *LNCS*, pages 383–395. Springer, 2000. → pages 7, 21, 22

[150] J. Wienberg. The evolution of eutherian chromosomes. *Current Opinion in Genetics and Development*, (6):657–666, 2004. → pages 11, 16, 18, 19

[151] R. Wittler and J. Stoye. Consistency of sequence-based gene clusters. In *Proceedings of RECOMB Comparitive Genomics*, volume 6398 of *Lecture Note in Bioinformatics*, pages 252–263. Springer, 2010. → pages ii, v, 27, 28, 73, 74, 82, 89, 118, 123

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

[153] F. Yang, E. Alkalaeva, P. Perelman, A. Pardini, W. Harrison, P. O'Brien, B. Fu, A. Graphodasky, M. Ferguson-Smith, and T. Robinson. Reciprocal chromosome painting among human, aardvark, and elephant (superorder afrotheria) reveals the likely eutherian ancestral karyotype. In *Proceedings of the National Academy of Sciences*, volume 100, pages 1062–1066, U.S.A., 2003. → pages 11, 16, 18, 19