Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property

Dissertation

zur Erlangung des akademischen Grades
Doctor rerum naturalium (Dr. rer. nat.)

vorgelegt dem Rat der Fakultät für Mathematik und Informatik
der Friedrich-Schiller-Universität Jena

von Dipl.-Inform. Michael Dom

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Gutachter:

- Prof. Dr. Rolf Niedermeier (Friedrich-Schiller-Universität Jena)
- Prof. Dr. Martin Mundhenk (Friedrich-Schiller-Universität Jena)
- Prof. Dr. Henning Fernau (Universität Trier)

Datum der letzten Prüfung des Rigorosums: 04. 12. 2008

Datum der öffentlichen Verteidigung: 10. 12. 2008

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Zusammenfassung

Diese Arbeit beschäftigt sich mit der Komplexität mehrerer kombinatorischer Probleme, die alle eng mit der „Consecutive-Ones"-Eigenschaft binärer Matrizen (d. h. 0/1-Matrizen) zusammenhängen: Eine binäre Matrix hat die *Consecutive-Ones-Eigenschaft (C1P)*, wenn ihre Spalten so geordnet werden können, dass in jeder Zeile sämtliche 1-Einträge direkt hintereinander stehen und somit in jeder Zeile nur ein „Einserblock" vorkommt.

Die C1P hat eine lange Geschichte und spielt eine wichtige Rolle in vielen Anwendungen, die von der algorithmischen Biologie bis hin zur Eisenbahnoptimierung reichen. Das Ziel der vorliegenden Arbeit ist es, einerseits zu untersuchen, wie die C1P erlangt werden kann, wenn eine Matrix diese Eigenschaft ursprünglich nicht hat, und andererseits zu analysieren, wie die C1P und „abgeschwächte" Varianten der C1P bei der Behandlung von Matrixproblemen helfen können, die im Allgemeinen nur schwer zu lösen sind.

Typischerweise gilt es bei der Analyse kombinatorischer Probleme zunächst zu untersuchen, ob ein vorliegendes Problem in *Polynomzeit* bezüglich der Eingabegröße gelöst werden kann, was für viele Anwendungen ausreichend schnell wäre. Kann man hingegen zeigen, dass das Problem NP-*schwer* ist, so bedeutet dies vermutlich, dass es keinen Polynomzeitalgorithmus geben kann. Doch auch für NP-schwere Probleme können oft positive Resultate erzielt werden. Dazu gibt es im Wesentlichen zwei Ansätze (Heuristiken, deren Laufzeit oder Lösungsgüte nicht bewiesen werden kann, außer Acht gelassen). Einerseits kann man versuchen, *Approximationsalgorithmen* für das Problem zu finden. Ein solcher Approximationsalgorithmus löst das Problem nicht exakt, er findet jedoch stets in Polynomzeit eine Lösung, die nur um einen konstanten Faktor von der optimalen Lösung abweicht. Bei manchen Problemen ist allerdings schon die Approximierung einer Lösung NP-schwer. Die zweite Möglichkeit ist, die zur Lösung des Problems benötigte Zeit exakter zu bestimmen, indem man diese nicht nur bezüglich der Eingabegröße analysiert, sondern einen zusätzlichen „Parameter" k wählt, den man ebenfalls in die Laufzeitanalyse mit einfließen lässt. Das Ziel ist dabei, die kombinatorische Explosion auf diesen Parameter zu beschränken. Demzufolge wird ein Problem als *festparameterhandhabbar* bezüglich der gewählten Parametrisierung bezeichnet, wenn jede Probleminstanz der Größe n und mit Parameter k innerhalb von höchstens $f(k) \cdot n^c$ Rechenschritten gelöst werden kann, wobei f eine berechenbare Funktion und c eine Konstante ist. Ein entsprechender Algorithmus heißt *Festparameteralgorithmus*. Ist ein Problem festparameterhandhabbar, so können selbst große Probleminstanzen schnell gelöst werden, vorausgesetzt, der Wert des Parameters ist klein. Kann man allerdings zeigen, dass ein Problem W[1]-*schwer* ist, so bedeutet dies vermutlich, dass es bezüglich der betrachteten Parametrisierung nicht festparameterhandhabbar ist.

Die Arbeit behandelt, nach einer breitgefächerten Übersicht über die C1P und ihre Verbindungen zur Graphentheorie und Algorithmik, vier Problembereiche.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Finden kleiner verbotener Teilmatrizen. Bei diesem Problem geht es darum, innerhalb einer Matrix M, die nicht die C1P hat, eine kleinstmögliche Teilmatrix ohne die C1P zu finden. Eine solche Teilmatrix wird *verbotene Teilmatrix* genannt, da sie der C1P in M entgegensteht. Das Problem, möglichst kleine verbotene Teilmatrizen zu finden, ergibt sich insbesondere, wenn eine Matrix durch Anwendung möglichst weniger Modifikationen die C1P erhalten soll.

Aus der Graphentheorie gibt es ein Ergebnis, welches die C1P mittels sogenannter asteroidaler Tripel charakterisiert. Mit Hilfe dieser Charakterisierung zeigen wir, dass man in Polynomzeit eine verbotene Teilmatrix finden kann, deren Anzahl an Zeilen und Spalten höchstens um einen konstanten Betrag größer ist als die einer optimalen Lösung, sprich, einer kleinsten verbotenen Teilmatrix. Darüberhinaus können wir, auf Kosten einer etwas höheren (aber immer noch polynomiellen) Laufzeit, auch verbotene Teilmatrizen mit einer minimalen Anzahl an Zeilen und Spalten finden.

Erzeugen der C1P mittels Zeilen- oder Spaltenlöschungen. Die zwei Probleme, eine gegebene Matrix durch eine kleinstmögliche Anzahl von Zeilen- bzw. Spaltenlöschungen in eine Matrix mit der C1P umzuformen, heißen Min-COS-R bzw. Min-COS-C. Die entsprechenden Maximierungsprobleme, bei denen es darum geht, eine Teilmatrix zu finden, welche die C1P hat und aus möglichst vielen Zeilen bzw. Spalten der gegebenen Matrix besteht, heißen Max-COS-R bzw. Max-COS-C. Alle diese vier Probleme sind bekanntermaßen selbst auf sehr dünn besetzten Matrizen, welche nur zwei Einsen pro Zeile und drei Einsen pro Spalte enthalten, NP-schwer.

Unsere Hauptergebnisse für diese Probleme sind Approximations- und Festparameteralgorithmen für Eingabematrizen, die nur eine beschränkte Anzahl Δ an Einsen pro Zeile enthalten: Sowohl Min-COS-R als auch Min-COS-C können auf solchen Matrizen in Polynomzeit approximiert werden, wobei sowohl der Grad des Polynoms der Laufzeit als auch der Approximationsfaktor von Δ abhängen. Daneben sind beide Probleme festparameterhandhabbar bezüglich der kombinierten Parameter Δ und „Lösungsgröße“. Sämtliche Ergebnisse beruhen auf einem neuen strukturellen Ergebnis, welches eine Verfeinerung einer bereits bekannten Charakterisierung der C1P darstellt.

Für Matrizen, die nur zwei Einsen pro Zeile oder pro Spalte enthalten, erhalten wir Nichtapproximierbarkeits- und W[1]-Schwerebeweise für Max-COS-R und Max-COS-C sowie diverse Resultate zur Approximierbarkeit und Festparameterhandhabbarkeit von Min-COS-R, Min-COS-C, Max-COS-R und Max-COS-C.

Abdeckungsprobleme bei Matrizen mit der C1P. Hier betrachten wir zwei Varianten des Set Cover-Problems. Genauso wie Set Cover können diese zwei Varianten als Matrixprobleme modelliert werden, welche auf Matrizen, die keiner Einschränkung unterliegen, NP-schwer sind. Wir untersuchen, ob die

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Probleme in Polynomzeit gelöst werden können, wenn in manchen oder gar in allen Zeilen der Eingabematrix nur jeweils ein Einserblock vorkommt.

Die beiden betrachteten Probleme heißen Minimum-Degree Hypergraph und Red-Blue Set Cover, und in beiden Fällen besteht die Eingabe aus einer Matrix, die aus „roten“ und „blauen“ Zeilen besteht. Während man bei Minimum-Degree Hypergraph nach einer Teilmenge der Spalten sucht, welche mindestens eine Eins aus jeder blauen Zeile, jedoch möglichst wenige Einsen aus jeder roten Zeile enthält, sucht man bei Red-Blue Set Cover nach einer Teilmenge der Spalten mit mindestens einer Eins aus jeder blauen Zeile, jedoch Einsen aus möglichst wenigen roten Zeilen. Durch Techniken aus dem Bereich des ganzzahligen linearen Programmierens bzw. durch dynamisches Programmieren können wir zeigen, dass sowohl Minimum-Degree Hypergraph als auch Red-Blue Set Cover in Polynomzeit gelöst werden können, wenn die Eingabematrix die C1P hat. Hat hingegen nur die aus den blauen oder den roten Zeilen bestehende Teilmatrix die C1P, so können wir zeigen, dass die entsprechenden Probleme NP-schwer sind. Indem wir zusätzlich die Anzahl der Einsen pro Zeile und pro Spalte in Betracht ziehen, können wir die Grenze zwischen polynomzeitlösbaren und NP-schweren Problemvarianten genau ausloten.

Schneiden von Rechtecken und Hyperrechtecken. (2-Dimensionales) Rectangle Stabbing ist ein geometrisches Problem, bei dem die Eingabe aus einer Menge von achsenparallelen Rechtecken und einer Menge von achsenparallelen Geraden besteht und die Aufgabe ist, eine kleinstmögliche Anzahl an Geraden auszuwählen, so dass jedes Rechteck mindestens einmal geschnitten wird. Die Verallgemeinerung dieses Problems auf d Dimensionen heißt d-Dimensionales Rectangle Stabbing, hier besteht die Eingabe aus Hyperrechtecken und Hyperebenen anstatt Rechtecken und Geraden. Das Problem kann für jeden Wert $d \geq 2$ als ein Spezialfall des Set Cover-Problems und somit als ein Spaltenauswahlproblem auf speziellen Matrizen aufgefasst werden: Die Eingabe besteht dann aus einer Matrix mit höchstens d Einserblöcken pro Zeile, und die Aufgabe ist, eine möglichst kleine Teilmenge der Spalten zu bestimmen, welche mindestens eine Eins aus jeder Zeile enthält.

d-Dimensionales Rectangle Stabbing ist unter Approximationsgesichtspunkten sehr gut studiert, die parametrisierte Komplexität blieb jedoch bislang unerforscht. Wir entwickeln eine Reduktion von einem Problem namens Multicolored Clique, welche für jeden Wert $d \geq 2$ die W[1]-Schwere des Problems d-Dimensionales Rectangle Stabbing bezüglich des Parameters „Anzahl der benötigten Geraden“ beweist. Dieses Ergebnis wird ergänzt durch Festparameteralgorithmen für verschiedene eingeschränkte Varianten des Problems (2-Dimensionales) Rectangle Stabbing.

Zusammengefasst, entwickeln wir einerseits zahlreiche neue Polynomzeitalgorithmen, Approximationsalgorithmen und Festparameteralgorithmen und erzielen andererseits NP-Schwere-, Nichtapproximierbarkeits- und W[1]-Schwereergeb-

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

nisse. Unser Hauptaugenmerk liegt auf einer systematischen und differenzierten Analyse der Komplexität der betrachteten Probleme und ihrer zahlreichen Varianten. Dies soll es zukünftigen Forschungen in Richtung praktikabler Algorithmen ermöglichen, sich auf diejenigen Problemvarianten zu konzentrieren, die wir als polynomzeitlösbar, mit konstantem Faktor approximierbar oder festparameterhandhabbar identifiziert haben.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Abstract

This thesis is about the computational complexity of several combinatorial problems closely related to the consecutive-ones property of binary matrices (0/1-matrices). Herein, a binary matrix has the *consecutive-ones property (C1P)* if there is a permutation of its columns that places the 1s consecutively in every row.

The C1P has a long history and plays an important role in many applications, ranging from computational biology to railway optimization. This thesis aims, on the one hand, to investigate how the C1P can be established when a given binary matrix initially does not have this property, and, on the other hand, to analyze how the C1P and "relaxed" variants of the C1P can help to tackle matrix problems that are hard to solve in general.

Typically, when analyzing computational problems, the first question to answer is whether the problem of interest can be solved in *polynomial time* with respect to the input size, which would be sufficiently fast for many applications. If, however, one can prove that the problem is NP-*hard*, this presumably implies that no polynomial-time algorithm can exist. For those problems that are NP-hard one can often provide positive results anyway—there are mainly two possibilities of doing so: On the one hand, one can try to develop an *approximation* algorithm for the problem. Such an approximation algorithm does not solve the problem exactly, but produces in polynomial time a solution that is only a constant factor away from the optimum. However, some problems are even NP-hard to approximate. On the other hand, one can examine the time needed for exactly solving the problem in a more fine-grained way. To this end, one considers, in addition to the size of the input, a "parameter" k and tries to confine the combinatorial explosion to this parameter. In this two-dimensional framework, a problem is called *fixed-parameter tractable* with respect to the chosen parameterization if any size-n instance with parameter k can be solved in $f(k) \cdot n^c$ time for a computable function f and a constant c. An algorithm running in this time is called a *fixed-parameter algorithm*. If a problem is fixed-parameter tractable, then even large problem instances can be solved quite fast provided that the value of the parameter is small. If, in contrast, one can prove that a problem is W[1]-*hard*, this presumably implies that the problem is not fixed-parameter tractable with respect to the considered parameterization.

Apart from giving an overview on the C1P and its connections to graph theory and algorithmics, the thesis examines four problem complexes.

Finding small forbidden submatrices. Here, one has given a binary matrix M without the C1P, and the task is to find a minimum-size submatrix of M that does not have the C1P. Such a submatrix is called a *forbidden submatrix* because it is a cause for the absence of the C1P in M. The problem of finding small forbidden submatrices arises when a matrix shall be modified to obtain the

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

C1P, but it is also of independent interest.

By exploiting a known result from graph theory, which characterizes the C1P by using so-called asteroidal triples, we show that we can find in polynomial time a forbidden submatrix of M whose number of rows and columns differs at most by an additive constant from the number of rows and columns of an optimal solution, that is, of a minimum-size forbidden submatrix. Moreover, at the cost of a slightly increased (but still polynomial) running time, we can find not only a forbidden submatrix of almost minimum size, but a forbidden submatrix consisting of a minimum number of rows and columns.

Obtaining the C1P by row or column deletions. The problems of deleting a minimum number of rows or columns to transform a given matrix into a matrix with the C1P are called Min-COS-R and Min-COS-C, respectively. The corresponding maximization problems, where one has to find a submatrix that has the C1P and consists of a maximum number of rows or columns, are called Max-COS-R and Max-COS-C, respectively. All four problems are known to be NP-hard even on very sparse matrices, containing only two 1-entries per row and at most three 1-entries per column.

Our main results for these problems are approximation and fixed-parameter algorithms for matrices that have only a bounded number Δ of 1-entries per row: Both Min-COS-R and Min-COS-C on such matrices can be approximated in polynomial time, where both the approximation factor and the degree of the polynomial in the running time depend on Δ. Moreover, Min-COS-R and Min-COS-C on such matrices are fixed-parameter tractable with respect to the combined parameters Δ and "solution size." All these results are based on a new structural result, which refines an already known "forbidden submatrix characterization" for binary matrices with the C1P.

Concerning matrices that have only two ones either per row or per column, we provide non-approximability and W[1]-hardness proofs for Max-COS-R and Max-COS-C, among various approximation and fixed-parameter tractability results for Min-COS-R, Min-COS-C, Max-COS-R, and Max-COS-C.

Covering problems on input matrices with the C1P. Here, we consider two variants of the subset selection problem Set Cover. Both of these two problems can be modelled as matrix problems and are NP-hard on general binary matrices; we investigate if they become polynomial-time solvable when some or all rows in the input matrix contain only one block of 1s each. The considered problems are called Minimum-Degree Hypergraph and Red-Blue Set Cover; in both cases the input consists of a matrix consisting of "red" and "blue" rows. Whereas Minimum-Degree Hypergraph asks for a subset of columns that contains at least one 1-entry from every blue row but only a minimum number of 1-entries from every red row, Red-Blue Set Cover asks for a subset of columns that contains at least one 1-entry from every blue row but

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

1-entries from a minimum number of red rows. Using integer linear programming and dynamic programming, respectively, we show that both Minimum-Degree Hypergraph and Red-Blue Set Cover are solvable in polynomial time on matrices with the C1P. If, however, only the submatrix consisting of the blue rows or only the submatrix consisting of the red rows has the C1P, then we can show that the resulting problems are NP-hard. By taking into account the number of red and blue rows and the number of 1-entries per row and per column, we explore a sharp border between the polynomial-time solvable and the NP-hard restrictions of the problems.

Rectangle stabbing in d dimensions. (2-Dimensional) Rectangle Stabbing is a geometric problem where the input consists of a set of axis-parallel rectangles and a set of axis-parallel lines, and the task is to select a minimum number of the given lines to intersect every rectangle at least once. The generalization of this problem to d dimensions is called d-Dimensional Rectangle Stabbing; here, hyperrectangles and hyperplanes are given instead of rectangles and lines. For any number $d \geq 2$ of dimensions, the problem can be interpreted as a special case of the problem Set Cover, and, therefore, as a column selection problem on specially structured binary matrices: Herein, the input consists of a binary matrix that has at most d blocks of 1s per row, and the task is to find a minimum-cardinality subset of columns that contains at least one 1-entry from each row.

d-Dimensional Rectangle Stabbing is well-studied from the approximation point of view; however, the parameterized complexity remained unexplored so far. We give nontrivial reductions from a problem called Multicolored Clique to show that d-Dimensional Rectangle Stabbing is W[1]-hard for the parameter "number of selected lines" for $d \geq 2$. This result is complemented by fixed-parameter algorithms for several restrictions of (2-Dimensional) Rectangle Stabbing.

In summary, we develop a number of new polynomial-time exact algorithms, polynomial-time approximation algorithms, and fixed-parameter algorithms on the one hand, and NP-hardness, non-approximability, and W[1]-hardness results on the other hand. Thereby, our main focus lies on providing a systematic and differentiated analysis of the computational complexity of the problems and their various restrictions. This should allow future research aiming towards practical algorithms to concentrate on those problem variants that we have identified as "tractable", that is, as polynomial-time solvable, constant-factor approximable, or fixed-parameter tractable.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Preface

This dissertation covers most of my research on the consecutive-ones property (C1P) of binary matrices and on the computational complexity of combinatorial problems related to the consecutive-ones property. I have done this research in the group of Prof. Rolf Niedermeier at the Friedrich-Schiller-Universität Jena, where I have been working as a scientific assistant since October 2004.

I am deeply grateful to Rolf Niedermeier, who supported and encouraged me throughout the time of my dissertation and who spent an immense amount of time and effort on suggesting research topics, proof-reading my results, and teaching me how to improve my writing style and presentation skills. Without having Rolf as supervisor and motivator, I would not have managed to write this work.

I also thank my colleagues Jiong Guo, Falk Hüffner, and Sebastian Wernicke for the enjoyable and productive cooperations—in particular, Jiong Guo and Falk Hüffner are co-authors of several of the publications where my name also appears in the author list. Furthermore, I am thankful to Nadja Betzler, Christian Komusiewicz, Hannes Moser, Johannes Uhlmann, and all the other people from Rolf Niedermeier's group for the pleasant and stimulating working atmosphere.

The co-authors I have not mentioned so far, but whom I also want to thank for the cooperations are Michael R. Fellows (University of Newcastle, Australia), Uwe Krüger (Universität Jena), Daniel Lokshtanov (University of Bergen, Norway), Frances A. Rosamond (University of Newcastle, Australia), Harald Sack (Universität Jena), Saket Saurabh (University of Bergen, Norway), Somnath Sikdar (The Institute of Mathematical Sciences, Chennai, India), Anke Truß (Universität Jena), and Yngve Villanger (University of Bergen, Norway). Mike Fellows and Frances Rosamond have been staying in Jena for almost one year, and Mike never got tired to share his ideas and a lot of anecdotes with us.

Finally, I owe thanks to the DAAD (Deutscher Akademischer Austauschdienst) and the DST (Department of Science and Technology, Government of India) for financing my stay at The Institute of Mathematical Sciences in Chennai, India, in October 2007 (DAAD-DST exchange program D/05/57666), and to the DFG (Deutsche Forschungsgemeinschaft), who sponsored three of my conference trips in the past four and a half years.

In this thesis, I present only results that concern the C1P and some combinatorial problems that are related to the C1P and to which I have made substantial contributions. Further research topics to which I have contributed and which are not part of this thesis are leaf roots [DGHN06, DGHN08] and tree roots [DGN05], feedback set problems [DGH$^+$10], dominating and covering problems [DLS09, DLSV08], and online databases for NP-hard problems [SKD06]. Moreover, I have worked on some book chapters and surveys [DHN08, Dom07, Dom08].

The thesis is structured into seven chapters. After a general introduction in Chapter 1, the C1P is introduced in Chapter 2; this chapter also provides an overview of the literature concerning the C1P. Chapters 3–6 consider four com-

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

binatorial problems: Finding a minimum-size submatrix that does not have the C1P (Chapter 3), deleting a minimum number of rows or columns to obtain the C1P (Chapter 4), and some variants of the Set Cover problem on input matrices that almost have the C1P (Chapters 5 and 6), where the variant considered in Chapter 6 can also be interpreted as a geometric stabbing problem. The findings are summarized in Chapter 7. Since I have collaborated with several co-authors, I will briefly describe my contributions to the results of every chapter.

Chapter 2 provides an overview over the literature concerning the C1P.[1] Moreover, I found and fixed a small error in Theorem 6.1 of [McC04].

Chapter 3 describes how in a given binary matrix without the C1P a small submatrix can be found that conflicts with the C1P. All results in this chapter were found by me; some of the results were presented at the *3rd Algorithms and Complexity in Durham (ACiD '07) Workshop* [DN07], together with the results from Section 4.5.2.[1]

Chapter 4 considers the problem of deleting a minimum number of rows or columns from a given matrix in order to obtain the C1P. Jiong Guo had the main idea for proving the hardness of the column deletion problem in Section 4.3, I extended the hardness proof to the row deletion problem. The algorithmic results for the maximization problems in Section 4.4 were found by me. Section 4.5 considers the minimization variant of the problem. The algorithmic framework presented in Section 4.5.1 was designed by Jiong Guo and me, the transformation of matrices with the circular-ones property into matrices with the C1P described in Section 4.5.2 was developed by me. All algorithms in Section 4.5 are based on Theorem 4.7. I came up with the proof of this theorem; this proof is presented in Section 4.7. The problem kernel for the column deletion problem in Section 4.6.1 was found by Jiong Guo, the adaption to the row deletion problem was done by me. The results in Section 4.6.2 were found by Jiong Guo and me. Most of the results from Sections 4.3, 4.5, 4.6, and 4.7 were presented at the *4th Annual Conference on Theory and Applications of Models of Computation (TAMC '07)* [DGN07], the results from Section 4.5.2 were presented at the *ACiD '07* [DN07].[1]

Chapter 5 deals with two variants of the problem Set Cover; our work on this topic was initiated by me. The greedy algorithm in Section 5.3.2 is due to Jiong Guo; the dynamic programming algorithm in Section 5.3.3 was designed by me. I thank an anonymous journal referee for pointing us to the negative cycle approach in Section 5.3.1. The hardness results in Section 5.4 were achieved by Jiong Guo, Sebastian Wernicke, Rolf Niedermeier and me in a number of discussions. The results of Chapter 5 were presented at the *10th Scandinavian Workshop on Algorithm Theory (SWAT '06)* and appeared in *Journal of Discrete Algorithms* [DGNW08].

Chapter 6 considers the problem d-Dimensional Rectangle Stabbing. I initiated our research on this topic. Except for Section 6.4, the work on this chapter was done by Somnath Sikdar and me, mostly during my stay at The In-

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

stitute of Mathematical Sciences in Chennai, India, in October 2007. All main results in Sections 6.3 and 6.5 were found by me, in each case after many fruitful discussions with Somnath Sikdar. Concerning the hardness proof for 3-Dimensional Rectangle Stabbing in Section 6.3, I thank Michael R. Fellows for explaining me the at that time unpublished "Multicolored Clique reduction technique", which we used in our proof. The results from Sections 6.3 and 6.5 were presented at the *2nd International Frontiers of Algorithmics Workshop (FAW '08)* [DS08]. For proving the hardness of 2-Dimensional Rectangle Stabbing in Section 6.4, I have been working together with Michael R. Fellows. The central idea for the reduction was brought up by him; this is, apart from its more complicated structure, one of the reasons for presenting this reduction separately from the hardness proof for 3-Dimensional Rectangle Stabbing in Section 6.3, which is a weaker result. The reduction found by Michael R. Fellows was quite involved; I significantly simplified the construction. The hardness result of Section 6.4 was published at the *3rd International Workshop on Algorithms and Computation (WALCOM '09)* [DFR09], together with some recent positive results for 2-Dimensional Rectangle Stabbing, which are not part of this thesis.

Jena, July 2008 / March 2009 *Michael Dom*

[1]The overview given in Chapter 2 has meanwhile appeared in *Bulletin of the European Association for Theoretical Computer Science*, 98:27–59, 2009; the results from Chapters 3 and 4 have appeared in *Journal of Computer and System Sciences*, 76(3–4):204–221.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Contents

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Chapter 1

Introduction

This thesis deals with combinatorial problems that are all closely related to the consecutive-ones property of binary matrices. Herein, the *consecutive-ones property (C1P)* means that the columns of a binary matrix can be ordered in such a way that the 1s appear consecutively in every row, that is, every row contains at most one block of 1s.

In this first chapter, we start with giving three short examples for the occurrence of the C1P in practical applications. In the remainder of the chapter, we introduce the notation used throughout the thesis and provide a short overview of the basic concepts of algorithmics and computational complexity theory.

1.1 Introductory Examples

This section presents, as a motivation and warm-up, three short examples that illustrate how the C1P can play a role in practical applications. The examples shall also demonstrate how problems from practical applications can be formulated in a compact and precise mathematical form that abstracts from all information that is unnecessary for solving the problem. Since our goal here is to give a rather intuitive understanding, we do not prove the correctness of the approaches described in the three examples.

Physical mapping of DNA. Our first example application has its background in computational biology, where the construction of physical maps for the human DNA was a central issue in the past years [ABH98, AM96, GGKS95, LH03, WR00]. A *physical map* is a map that describes the relative order of *markers* on a *chromosome*. A chromosome is basically a long sequence of DNA, and a marker is a short DNA sequence that appears only once on the chromosome and, therefore, is of special interest. To create a physical map, the chromosome is cut into shorter pieces, which are duplicated and called *clones*. Thereafter, one tests for each of the clones which of the markers appears on it. These tests, however, can only find out whether a marker appears on a clone, but it is not possible to

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

	A	*B*	*C*	*D*	*E*	*F*	*G*
1	0	**1**	0	0	0	**1**	0
2	0	**1**	**1**	**1**	0	0	**1**
3	**1**	**1**	**1**	**1**	0	0	0
4	0	**1**	**1**	**1**	0	0	**1**
5	0	0	**1**	0	**1**	0	0
6	**1**	0	0	**1**	**1**	**1**	0
7	0	**1**	0	**1**	0	**1**	**1**
8	**1**	**1**	0	0	0	**1**	0

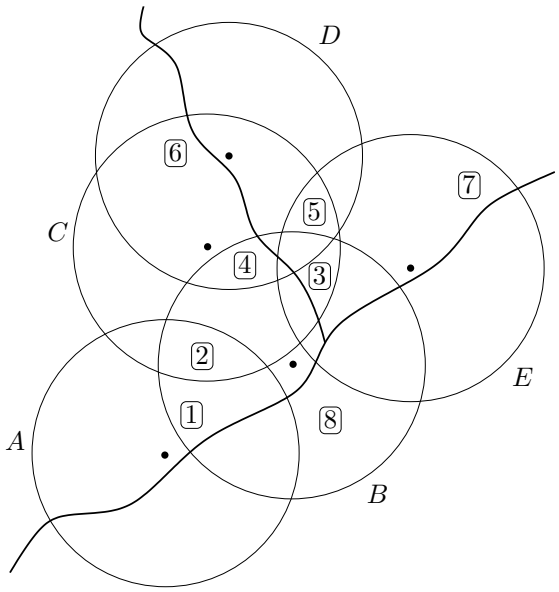
	A	*F*	*B*	*G*	*D*	*C*	*E*
1	0	**1**	**1**	0	0	0	0
2	0	0	**1**	**1**	**1**	**1**	0
~~3~~	~~**1**~~	~~0~~	~~**1**~~	~~0~~	~~**1**~~	~~**1**~~	~~0~~
4	0	0	**1**	**1**	**1**	**1**	0
5	0	0	0	0	0	**1**	**1**
~~6~~	~~**1**~~	~~**1**~~	~~0~~	~~0~~	~~**1**~~	~~0~~	~~**1**~~
7	0	**1**	**1**	**1**	**1**	0	0
8	**1**	**1**	**1**	0	0	0	0

Figure 1.1: An application from physical mapping.

determine the order of the markers on the clone. The result is a binary matrix as shown in the left part of Figure 1.1: Every row corresponds to a clone, and every column corresponds to a marker. If a marker appears on a clone, then the corresponding entry of the matrix is 1, otherwise it is 0. Now, the crucial observation for finding the correct order of the markers is that if two markers A and B appear on a clone x, but another marker C does not appear on x, then C cannot lie between A and B on the chromosome. Therefore, to figure out the order of the markers on the chromosome, all one has to do is to order the columns of the matrix in such a way that in every row the 1s appear consecutively. In concrete practical applications, however, the biochemical methods always produce errors such that it is often impossible to order the columns in the resulting matrices as described. One way to deal with these errors is to discard a smallest possible number of clones such that the remaining clones lead to a consistent order of the markers. On the level of binary matrices, this approach means that one has to delete a minimum number of rows such that in the resulting matrix the 1s can be placed consecutively by reordering the columns. This matrix problem is the subject of Chapter 4 of this thesis. The right part of Figure 1.1 shows that in our example we do not have to delete more than two rows.

Placing sender stations in cellular networks. Cellular networks are networks that consist of two types of participants: base stations and client stations. For example, cell phone towers and cell phones form a cellular network. The operator of a such a network can be confronted with the following problem (see also [KRW+05]): To guarantee the network coverage of an area with several settlements, new base stations have to be built. Taking into account the landscape, and, since the stations should be accessible by car, there exist only a certain number of locations that are suitable for planting new base stations. Once a base station is built, it has a certain transmission range. The left part of Figure 1.2 demonstrates the situation: there are eight settlements 1–8 and five suitable locations A–E; each location is drawn as a point together with a cycle denoting the

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.



	A	*B*	*C*	*D*	*E*
1	**1**	**1**	0	0	0
2	**1**	**1**	**1**	0	0
3	0	**1**	**1**	0	**1**
4	0	**1**	**1**	**1**	0
5	0	0	**1**	**1**	**1**
6	0	0	**1**	**1**	0
7	0	0	0	0	**1**
8	0	**1**	0	0	0
3	0	**1**	**1**	0	**1**
4	0	**1**	**1**	**1**	0

Figure 1.2: The problem of placing base stations in cellular networks.

transmission range of a base station that could be placed at this point. The task of the network operator now is to select a sufficient large number of the locations for building his new base stations there, such that an optimal network coverage of all settlements is obtained. Thereby, two constraints have to be regarded: First, every settlement should lie within the transmission range of at least one base station. Second, there are some client stations that are sensitive to interferences. In our example, let us assume that such client stations exist in the settlements 3 and 4. Therefore, each of the settlements 3 and 4 should lie within the transmission range of at most two base stations—receiving signals from three or more base stations would disturb the client stations there.

The right part of Figure 1.2 shows how to translate this problem into a matrix problem. The corresponding matrix consists of an upper part and a lower part. The upper part has one row for every settlement 1–8, and the lower part has one row for each of the settlements 3 and 4. Furthermore, the matrix has one column for every location *A*–*E*. If a settlement lies within the transmission range of a potential base station, then the corresponding entry of the matrix is 1, otherwise it is 0. The problem that now has to be solved on this matrix is: Find some columns that contain at least one 1 from every row of the upper part and at most two 1s from every row of the lower part. In our example, the columns *B*, *D*, and *E* would form a solution for this problem. One can easily verify that building base stations at the corresponding locations yields a network coverage for the settlements as desired.

Note that the matrix resulting from the base station problem typically has a very special structure (see also [MSW05, MW04, RS04]): it is “close” to having

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

the C1P—in our example, there are only two blocks of 1s in every row. This is due to the facts that the transmission ranges of the base stations are cycles and that the locations A–E of the base stations all are close to some street and, therefore, are arranged in a special way.

Sensor selection in multi-sensor fusion applications. Our third example is adapted from Koushanfar et al. [KSPS02] and deals with minimizing the number of sensors in a multi-sensor fusion application. In this application, one has to classify objects by using a set of sensors. Herein, classifying means to determine for every object that is detected to which of six given object types A–F the object belongs. Every object has two properties—we will call them size and wavelength here—, which can be expressed as a number each. By considering these two properties, every object can uniquely be assigned to one of the six object types. In particular, no two object types have the same combination of these two properties. See parts a) and c) of Figure 1.3 for two examples; the object types are displayed as points in a two-dimensional coordinate system where one coordinate stands for the size and the other for the wavelength of the corresponding object type. The object type C in part a) of Figure 1.3, for example, consists of objects of size 2.7 μm and a wavelength of 420 nm.

Now assume that there are a huge number of different sensors available; each of these sensors has a certain threshold value t and can either detect whether the size of an object is at least t, or whether the wavelength is at least t (in particular, a sensor cannot measure both size *and* wavelength). For several reasons (costs, simplicity,...), as few as possible sensors shall be bought and installed, such that with these few sensors it is possible to identify the type of every object that passes the sensors. For the object types displayed in part a) of Figure 1.3 one needs five sensors; part b) of Figure 1.3 shows one (of several) possibilities how these five sensors can be selected—every vertical or horizontal line in this illustration corresponds to one sensor. These five sensors indeed have the ability to classify every object: If, for example, the sensors report that an object has a size between 200 and 400 μm and a wavelength of at least 500 nm, then this object must be of type D. Part d) of Figure 1.3 shows that three sensors are sufficient for the object types shown in part c) of the figure.

Given a set of object types, how can one find a suitable set of as few as possible sensors? Parts b) and d) of Figure 1.3 illustrate a way how to interpret this task as a geometric problem: The problem of selecting sensors is equivalent to the problem of finding a minimum-size set of axis-parallel lines in the coordinate systems of parts a) and c) of Figure 1.3, such that every pair of points is divided by these lines. In other words, one has to find a minimum-size set of axis-parallel lines that divide the coordinate system into several areas, such that no two points lie within the same area. In order to solve the problem of dividing points with lines, we transform this problem into another geometric problem. To this end, for every pair of points in the coordinate system, we insert a rectangle such that

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

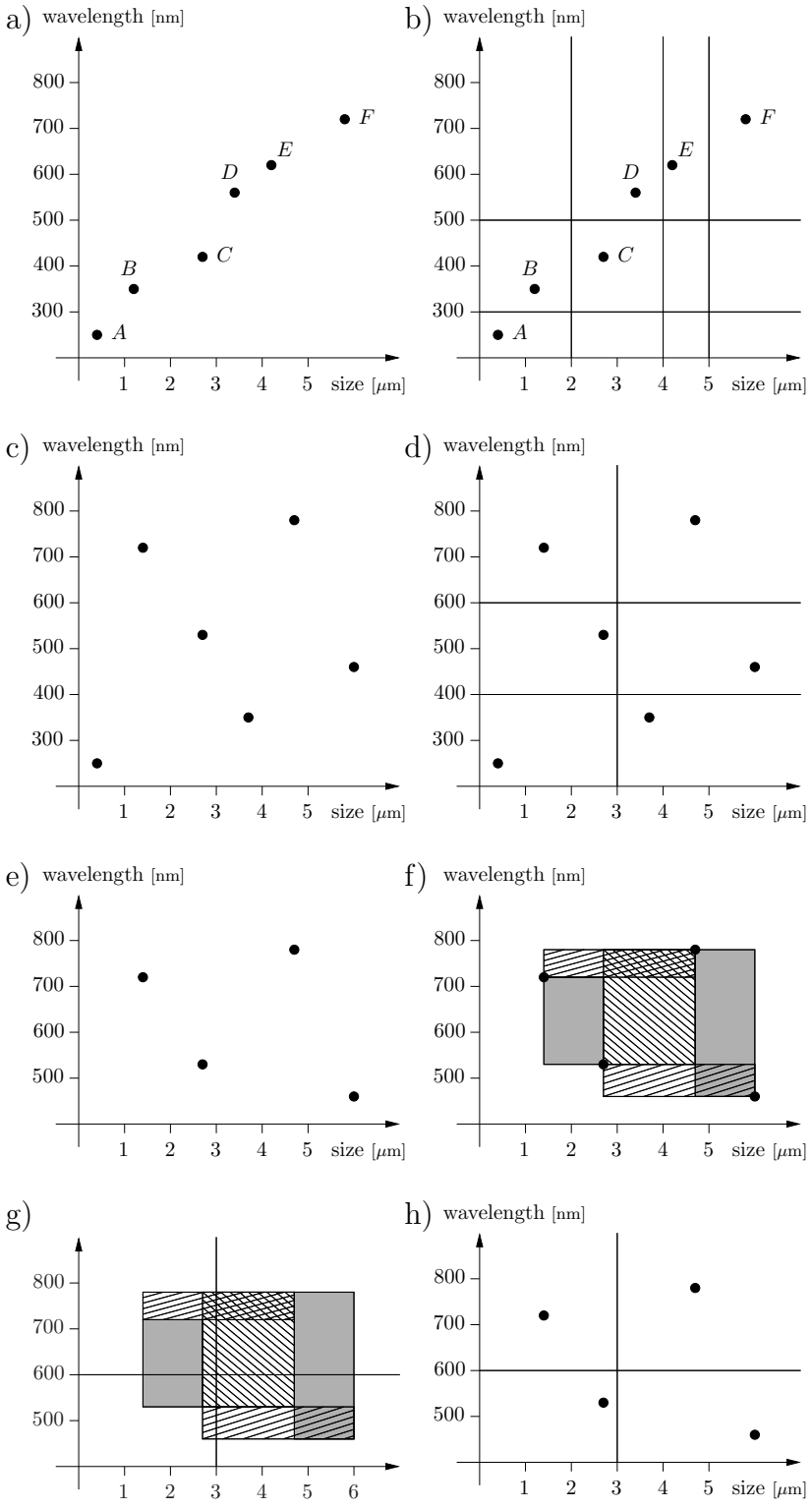


Figure 1.3: Selecting a minimum number of sensors.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

the two points lie on two opposite edges of the rectangle [CDKW05].

Part e) of Figure 1.3 shows an example which consists, for the ease of presentation, of only four object types. Part f) of Figure 1.3 shows how to insert the rectangles. Since rectangles that contain other rectangles can be omitted, there are five rectangles. (In the figure, two rectangles are grey-colored, two rectangles are drawn with a diagonal top right to bottom left hatching, and one is drawn with a diagonal top left to bottom right hatching.) The problem that has to be solved now is the following: Find a set of axis-parallel lines such that every rectangle is intersected by at least one of these lines. This problem is known as (2-Dimensional) Rectangle Stabbing, and it is identical to a column selection problem on matrices that have at most two blocks of 1s per row. Part g) of Figure 1.3 shows that all rectangles in the example can be intersected with only two lines, and part h) shows the corresponding solution for dividing the four given points in the coordinate system. Hence, for classifying objects, in this example two sensors would suffice: one sensor with a size threshold of 3 μm and one sensor with a wavelength threshold of 600 nm.

All problems occurring in these three application examples are subject of this thesis. The problem of obtaining the C1P by row or column deletions is addressed in Chapter 4. For solving this problem, we also have to identify those parts of a matrix that conflict with the C1P; in Chapter 3 we provide our results in this direction. Selecting columns from a matrix in order to hit at least one 1-entry from some of the rows but not too many 1-entries from the other rows is the subject of Chapter 5, and in Chapter 6, finally, we consider the (d-Dimensional) Rectangle Stabbing problem.

1.2 Basic Definitions

A set S *properly* contains a set S' if $S' \subseteq S$ and $S \setminus S' \neq \emptyset$; we also say that S' is a *proper subset* of S. A set S is called *minimal* (*maximal*) with respect to a property if no proper subset (no proper superset) of S also has this property. In contrast, a set is called *minimum* (*maximum*) with respect to a property if there exists no set of smaller (greater) cardinality that has the property.

As usual, we often write *iff* instead of "if and only if." With $\log(x)$ we denote the logarithm of x to the base 2. By $\mathbb{N}$, we refer to the set of positive integers. For two integers i, j with $j > 0$, the remainder of the division i by j is denoted by $i \bmod j$; for example, $17 \bmod 5 = 2$. We define $i \operatorname{\underline{mod}} j$ as

$$i \operatorname{\underline{mod}} j := ((i-1) \bmod j) + 1,$$

that is,

$$i \operatorname{\underline{mod}} j = \begin{cases} i \bmod j & \text{if } i \bmod j > 0 \\ j & \text{if } i \bmod j = 0 \end{cases}$$

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Moreover, for an integer $n > 0$ we define

$$\mathrm{pred}_n,\ \mathrm{succ}_n : \{1, \ldots, n\} \to \{1, \ldots, n\}$$

as the two functions given by

$$\mathrm{pred}_n(x) := (x-1) \underline{\bmod}\, n,\ \mathrm{succ}_n(x) := (x+1) \underline{\bmod}\, n,$$

that is,

$$\mathrm{pred}_n(x) = \begin{cases} x-1 & \text{if } x > 1 \\ n & \text{if } x = 1 \end{cases}$$

and

$$\mathrm{succ}_n(x) = \begin{cases} x+1 & \text{if } x < n \\ 1 & \text{if } x = n. \end{cases}$$

The *Big-O-Notation* allows to ignore constants when describing functions: Given two functions $f, g : \mathbb{N} \to \mathbb{R}$, then $f \in O(g)$ if $\exists c > 0\ \exists n_0\ \forall n > n_0 : |f(n)| \leq c \cdot |g(n)|$. One often writes $f = O(g)$ instead of $f \in O(g)$, and if $f \in O(g)$, one often writes $O(g(x))$ to denote the value $f(x)$.

The *O^*-Notation* is a notation similar to the Big-O-Notation; it was introduced by Woeginger [Woe03] for describing running times of algorithms and thereby omitting all polynomials in the input size. We use the following definition: Given two functions $f, g : \mathbb{N}^d \to \mathbb{R}$, then $f \in O^*(g)$ if there exists a polynomial $p : \mathbb{N}^d \to \mathbb{R}$ such that $\exists n_0\ \forall n_1, \ldots, n_d > n_0 : |f(n_1, \ldots, n_d)| \leq |p(n_1, \ldots, n_d) \cdot g(n_1, \ldots, n_d)|$.

For basic introductions to discrete mathematics and algorithmics, we refer to [Ros06, CLRS01].

Graphs. An *(undirected) graph* is a tuple (V, E), where V is a finite set and E is a set of size-two subsets from V. An element from V is called a *vertex*, and an element from E is called an *edge*. For a graph G, we denote with $V(G)$ the set of G's vertices and with $E(G)$ the set of G's edges. In a graph $G = (V, E)$, two vertices v and w are *adjacent* (or *connected by an edge*) if E contains the edge $\{v, w\}$; in this case v and w are *neighbors* of each other. The edge $\{v, w\}$ is *incident* to v and w, and the vertices v, w are the *endpoints* of $\{v, w\}$. The *degree* $\deg(v)$ of a vertex v denotes the number of its neighbors. The *(open) neighborhood* of a vertex v is the set of all neighbors of v and is denoted by $N(v)$. The *closed neighborhood* of v, denoted by $N[v]$, is defined as $N[v] := N(v) \cup \{v\}$. A subgraph of G is a graph $G' := (V', E')$ with $V' \subseteq V$ and $E' \subseteq E$. For $V' \subseteq V$, the subgraph of G that is *induced* by V' is the graph $G' = (V', E')$ with $E' = \{\{v, w\} \in E \mid v \in V' \wedge w \in V'\}$; this subgraph is denoted by $G[V']$. *Deleting* a vertex $v \in V$ from a graph $G = (V, E)$ means deleting v from V and deleting every edge from E where v is one of the endpoints.

A *path* is a graph $P = (V, E)$ with vertex set $V = \{v_1, \ldots, v_n\}$ and edge set $E = \{\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{n-2}, v_{n-1}\}, \{v_{n-1}, v_n\}\}$; the vertices v_1 and v_n are

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

called the *endpoints* of P. A *cycle* is a graph consisting of a path $v_1, \ldots, v_n$ and the additional edge $\{v_n, v_1\}$. Sometimes we describe a path or a cycle by giving only the sequence $v_1, \ldots, v_n$ of its vertices. The *length* of a path or cycle is the number of its edges, that is, the length of a path P equals the number of P's vertices minus 1 and the length of a cycle C equals the number of C's vertices. With P_n we denote a path with n vertices, and with C_n we denote a cycle with n vertices. A *chord* of a cycle is an edge e that is not part of the cycle but connects two vertices of the cycle. A *hole* is an induced cycle of length at least 5, that is, a cycle of length at least 5 where no chords exist. A graph is *chordal* if it contains no induced cycle (that is, no chordless cycle) of length greater than three. Two vertices v_i, v_j in a graph G are called *connected (by a path)* if G contains a path as subgraph whose endpoints are v_i and v_j. A graph is called a *connected graph* if all of its vertices are pairwisely connected by paths. A maximal connected subgraph of a graph G is called a *connected component* of G. A *tree* is a connected graph without cycles; a vertex of degree one in a tree is called a *leaf*. A *rooted* tree is a tree where one vertex is marked as the *root* of the tree. A *clique* is a complete graph, that is, a graph $G = (V, E)$ with $E = \{\{v, w\} \mid v, w \in V \ \wedge \ v \neq w\}$. With K_n we denote a clique with n vertices. An *independent set* is a set of vertices that are not connected by edges.

A graph $G = (V, E)$ is *bipartite* if V can be partitioned into two vertex sets V_1, V_2 such that every edge in E has one endpoint in V_1 and one endpoint in V_2. A bipartite graph $G = (V, E)$ together with a partition of V into V_1 and V_2 is often denoted by a triple (V_1, V_2, E). A hole in a bipartite graph is an induced cycle of length at least 6.

A *directed graph* is a tuple (V, E), where the edges from E are ordered pairs (instead of size-2 sets) of vertices from V. Directed paths and cycles are defined analogously to the undirected case, that is, a *directed path* is a directed graph $P = (V, E)$ with vertex set $V = \{v_1, \ldots, v_n\}$ and edge set $E = \{(v_1, v_2), \ldots, (v_{n-1}, v_n)\}$, and a *directed cycle* is a graph consisting of a path $v_1, \ldots, v_n$ and the additional edge (v_n, v_1).

For a general introduction to graph theory, we refer to [Die05, Wes01]; for more about directed graphs, see [BG02].

Matrices. A *matrix* is a rectangular table of numbers, which are called the *entries* of the matrix. An $m \times n$ *matrix* contains $m \cdot n$ entries, which are arranged in m rows and n columns. The entry in the ith row and jth column of a matrix M is denoted by $m_{i,j}$; moreover, we usually use r_i and c_j to denote the ith row and the jth column, respectively, of a matrix. One can also regard a matrix as a set of columns (or rows) together with an order on this set; the order of the columns (rows) is called the *column ordering* (*row ordering*) of the matrix. Two matrices M and M' are called *isomorphic* if M' is a permutation of the rows and columns of M. In particular, if two matrices consist of the same set of columns (rows) but only differ in their column orderings (row orderings), these matrices

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

are isomorphic (however, the reverse is not always true, since two isomorphic matrices may consist of differing rows *and* columns). We use the term *line* of a matrix M to denote a row or column of M.

A matrix M' is usually called a *submatrix* of a matrix M if one can select a subset of the rows and columns of M in such a way that deleting all but the selected rows and columns from M results in M'. We extend this notion and call M' also a submatrix of M if we can select a subset of the rows and columns of M in such a way that deleting all but the selected rows and columns results in a matrix that is *isomorphic* to M'. If one can find a submatrix M' of M in this way, we say that M *contains* M' as a submatrix and that M' is *induced* by the selected rows and columns. A matrix M is M'*-free* if M' is not a submatrix of M. Let r_i denote the ith row and let c_j denote the jth column of M, and let M' be the submatrix of M that results from deleting all rows except for $r_{i_1}, \ldots, r_{i_p}$ and all columns except for $c_{j_1}, \ldots, c_{j_q}$ from M. Then, a row r_i of M *belongs* to M', denoted by $r_i \in M'$, if $i \in \{i_1, \ldots, i_p\}$. Analogously, a column c_j of M belongs to M' if $j \in \{j_1, \ldots, j_q\}$. A submatrix of a matrix M is called a *proper* submatrix of M if not all rows or not all columns of M belong to M'.

A matrix whose entries are all from $\{0, 1\}$ is called a *binary matrix* or 0/1-*matrix*; a matrix whose entries are all from $\{0, 1, -1\}$ is called a $0/\pm 1$-matrix. A column of M that contains only 0-entries is called a 0-*column*. *Complementing* a line ℓ of a matrix means that all 1-entries of ℓ are replaced by 0s and all 0-entries are replaced by 1s.

A *square matrix* is an $m \times n$ matrix with $m = n$; the *main diagonal* of an $n \times n$ square matrix M denotes the entries $m_{1,1}, m_{2,2}, \ldots, m_{n,n}$. A *unit matrix* is a square matrix where the entries of the main diagonal are 1 and all other entries are 0. The *transpose* of an $m \times n$ matrix M, denoted by M^{T}, is the $n \times m$ matrix M' with $m'_{j,i} = m_{i,j}$. A *vector* $\vec{x}$ is an $m \times 1$ matrix, its entries are usually denoted with $x_1, \ldots, x_m$.

The *half adjacency matrix* of a bipartite graph $G = (V_1, V_2, E)$ with $V_1 = \{v_1, \ldots, v_{n_1}\}$ and $V_2 = \{w_1, \ldots, w_{n_2}\}$ is the $n_1 \times n_2$ binary matrix M with $m_{i,j} = 1$ iff $\{v_i, w_j\} \in E$. Every 0/1-matrix M can be interpreted as the half adjacency matrix of a bipartite graph; this graph is called the *representing graph* G_M of M. In other words, for every row and every column of a matrix M, there is a vertex in its representing graph G_M, and for every 1-entry $m_{i,j}$ in M, there is an edge in G_M connecting the vertices corresponding to the ith row and the jth column of M.

1.3 Computational Complexity Theory

The main chapters of this work deal with the question whether there are efficient algorithms for certain combinatorial problems. Herein, "efficient" means that the running time of an algorithm is a slowly growing function in the size of the input. This question, or, more generally, the analysis of the amount of required re-

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

sources (not only time but also, for example, memory space or bits of information exchanged between several processors) for solving problems is one of the main issues in computational complexity theory and theoretical computer science. When considering a problem (we will define later what exactly we mean with the term "problem"), hence, the task is typically to find either an efficient algorithm for the problem, or, contrariwise, a proof for the non-existence of such an algorithm. Unfortunately, in most cases where we are not able to find an efficient algorithm, there are no methods known how to prove that an efficient algorithm cannot exist. Therefore, we are usually satisfied with comparing such a difficult problem with other problems. To this end, we sort problems into complexity classes, which allows to say that a problem is "at least as difficult as many other problems that are already assumed to be difficult." There are many "hardness predicates" of this kind (for example, NP-hardness, APX-hardness, W[1]-hardness, ...), and the ways they are defined are very similar in most cases: First, define a class $\mathcal{C}$ that contains many problems that are already assumed to be "difficult" (which means that after a long period of research there is still no efficient algorithm known for them). Second, show that if there was an efficient algorithm for one of the "difficult" problems in $\mathcal{C}$, then for *all* problems in $\mathcal{C}$ there would exist efficient algorithms (this affirms the "difficulty" of each of the "difficult" problems). Now, a problem X can be called "$\mathcal{C}$-hard" if one can show that the existence of an efficient algorithm for X would imply the existence of efficient algorithms for *all* problems in $\mathcal{C}$.

In this section, we give a more formal description of the types of problems we are dealing with, and we introduce some of the most important complexity classes and notions of hardness. We restrict ourselves to the resource "time", that is, we only consider the time that is needed by an algorithm; on the one hand, because this is the resource that is studied most extensively in literature, and, on the other hand, because time seems to be the resource most relevant in practice.[1]

1.3.1 "Classical" Complexity Theory

Here, we introduce the main concepts of complexity theory as described by Papadimitriou [Pap94]. We start with considering decision problems and the corresponding complexity classes and will then turn over to other problem types.

Decision problems. The first type of problems to describe are *decision problems*. A decision problem has a (usually infinitive) set of possible inputs, which are called *(problem) instances* and consist of mathematical objects. For every given instance of a particular problem, a question is posed, which asks if the instance has a certain property and which can be answered with *yes* or *no*. This question

[1] If an algorithm needs only a limited amount of time for its computation, then, of course, the memory space it needs is also bounded because for every memory access a certain amount of time has to be spent.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

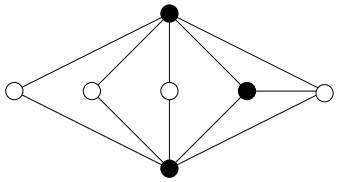


Figure 1.4: Example for VERTEX COVER. The black vertices form a vertex cover of size three: Every edge has at least one black endpoint.

is specific for each decision problem, which means that every decision problem is uniquely defined by the allowed inputs and the question that is asked for each of the inputs. Therefore, we consider a decision problem X as a pair (I_X, q_X) consisting of a (usually infinitive) set I_X of instances and a question q_X. A problem instance $x \in I_X$ is called a *yes-instance* of X if the answer to the question q_X is *yes* for x, and a *no-instance* otherwise.[2]

As an example for a decision problem, consider the problem VERTEX COVER, which is defined as follows and will be used as a running example throughout this section.

VERTEX COVER
Input: An undirected graph $G = (V, E)$ and a nonnegative integer k.
Question: Is there a subset $C \subseteq V$ of at most k vertices such that each edge in E has at least one endpoint in C?

In the case of VERTEX COVER, every problem instance consists of a pair $(G = (V, E), k)$; a vertex set $C \subseteq V$ with the property that every edge in E has at least one endpoint in C is called a *vertex cover* for the graph G (see Figure 1.4).

Running Times and the Turing Machine Model. Problems can be classified by considering the running times that are needed to solve them. To this end, an abstract computer model is used which is called *(deterministic) Turing machine* and whose computational power is identical to that of most relevant real-world computer programs—for example, each problem that can be solved by a Java program can also be solved within a "similar" running time by an adequate Turing machine and vice versa. The running time needed to solve a problem X is measured in terms of how many steps a Turing machine M has to perform to solve the problem. This number of steps is always given as a function $t_M(n)$ in the size n of the input. (For example, the running time of a Turing machine solving VERTEX COVER would be given as a function in the number of vertices

[2] An alternative point of view is the definition of a decision problem as a *language*: A language, in the sense of complexity theory, is a set of binary strings, the so-called *words*; instead of defining a decision problem X as a pair (I_X, q_X), one can define X by a language that contains as words all *yes*-instances of X, encoded as binary strings.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

and edges of the input graph.) More exactly, the function $t_M(n)$ always expresses the so-called *worst-case running time* of the Turing machine M, that is, $t_M(n)$ is the maximum running time needed by M, taken over all inputs of size n.

The Complexity Class P**.** In order to constitute a classifying tool, problems that can be solved within similar running times are grouped together and sorted into *complexity classes*, with P and NP being two of the most important of these classes. The class P contains all decision problems that can be solved within a *polynomial running time*. That is, for every problem $X \in$ P there is a Turing machine M_X whose running time $t_{M_X}(n)$ is a polynomial in n. For most practical applications such a polynomial running time means that the problem is solvable within a reasonable amount of time. However, there are a lot of problems of practical relevance for which no polynomial-time algorithms are known.

Nondeterminism. For defining the complexity class NP, a modified computer model called *nondeterministic Turing machine* is introduced: In contrast to deterministic Turing machines, a nondeterministic Turing machine has the freedom to *guess* in every step. By definition, a nondeterministic Turing machine solves a problem if for every *no*-instance it always answers correctly with *no*, and if for every *yes*-instance it answers correctly with *yes* provided that it has made the right guess in every step. One can imagine a nondeterministic Turing machine as a randomized machine that answers correctly for every *no*-instance and that answers correctly with a probability greater than zero for every *yes*-instance. Typically, a nondeterministic Turing machine solves a problem with the following two-phase approach: In the first phase, it guesses a "witness" that proves the correctness of the answer *yes* (for example, in the case of Vertex Cover it guesses a vertex set C of size k). In the second phase, it checks—without guessing—whether the witness is correct (in our example, it checks whether C is indeed a vertex cover) and answers according to the result of the check. A correct witness is usually called a *certificate* or *solution*.

The Complexity Class NP**.** Like in the case of deterministic Turing machines, the running time of a nondeterministic Turing machine M for a problem X is expressed as a function $t_M(n)$ in the input size n; it gives the maximum number of steps the Turing machine can need for solving a problem instance of size n. The class NP is defined as the class of all decision problems X that can be solved nondeterministically within a polynomial running time. That is, for every problem $X \in$ NP there is a nondeterministic Turing machine M_X whose running time $t_{M_X}(n)$ is a polynomial in n. Therefore, the class NP contains exactly those problems where each *yes*-instance has a certificate whose size is polynomial in the input size and whose correctness can be checked deterministically in polynomial time. (Note that the class P contains those problems of NP where a certificate of

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

each *yes*-instance can not only be checked, but also constructed deterministically in polynomial time.)

Polynomial-Time Reductions, NP-Hardness, and NP-Completeness. Due to the power of guessing, the class NP contains an enormous number of problems (although there exist problems that are even too hard to be solved nondeterministically in polynomial time), many of them of substantial practical relevance. In particular, all problems that belong to P are also contained in NP (which directly follows from the definitions). However, there are a lot of important problems in NP that seem not to be solvable deterministically in polynomial time: Replacing the powerful guessing by deterministic steps—for example, by trying several possibilities—often results in exponential running times (the so-called "combinatorial explosion"), which often makes these problems intractable in practice. In order to unite such "difficult" problems into a class of their own, and to produce evidence for their intractability by showing that either all or none of these problems are solvable deterministically in polynomial time, the concept of *reductions* is introduced: A problem X is *(polynomial-time) reducible* to a problem Y, denoted by $X \leq_{\mathrm{P}} Y$, if there is a function Φ that maps every problem instance x of X to a problem instance $y = \Phi(x)$ of Y such that y is a *yes*-instance of Y iff x is a *yes*-instance of X. Moreover, there must be a polynomial t_Φ such that the time for computing $\Phi(x)$—and, hence, also the size of $\Phi(x)$—does not exceed $t_\Phi(|x|)$. Intuitively speaking, the problem Y is "as least as hard" as the problem X, because a problem instance x of X can be solved in polynomial time by using any polynomial-time algorithm for the problem Y: First, compute the problem instance $y = \Phi(x)$ of Y, and then solve y using the algorithm for Y—the output of this algorithm is the answer for y as well as for x.[3]

The concept of reducing one problem to another can be illustrated by the very simple reduction of the problem Independent Set to Vertex Cover: The problem Independent Set asks, given a graph $G = (V, E)$ and a nonnegative integer k, whether G has an independent set $V' \subseteq V$ of at least k vertices. (An independent set is a set of vertices that are pairwise not connected by edges.) Independent Set can be reduced to Vertex Cover by mapping each problem instance (G, k) of Independent Set to a problem instance $(G, |V| - k)$ of Vertex Cover. If $(G, |V| - k)$ is a *yes*-instance of Vertex Cover, then G has a vertex cover C of size at most $|V| - k$ and, therefore, (G, k) is a *yes*-instance of Independent Set: The vertices not belonging to C are not connected by

[3]Polynomial-time reductions as described here are also called *many-one reductions* or *Karp reductions*. There are also other types of reductions (see [LLS75]), of which the most common is called *Turing reduction*: A problem X is called *Turing reducible* to a problem Y if there is a deterministic Turing machine M that can solve X in polynomial time provided that M has a built-in subroutine—called *oracle*—that solves Y in constant time. For solving X in polynomial time, M can construct polynomially many polynomial-size instances of Y and call the oracle on these instances. The class that contains all problems that are Turing reducible to problems in NP is called P^{NP}.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

edges and form an independent set of size at least k (to see this, consider the white vertices in Fig 1.4). If, however, the instance $(G, |V| - k)$ is a *no*-instance of Vertex Cover, then (G, k) is a *no*-instance of Independent Set, because if there was an independent set V' of size at least k, then the vertices in $V \setminus V'$ would form a vertex cover of size at most $|V| - k$.

The definition of polynomial-time reductions directly implies that if $X \leq_{\mathrm{P}} Y$ and $Y \in \mathrm{P}$, then also $X \in \mathrm{P}$ ("the class P is closed under polynomial-time reductions"). The other way round, if $X \leq_{\mathrm{P}} Y$ and if X is one of the "difficult" problems, it is unlikely that Y can be solved in polynomial time (because otherwise the reduction would constitute a polynomial-time algorithm for X). A problem Y is called NP-*hard* if every problem $X \in \mathrm{NP}$ can be reduced to Y. If, in addition, the problem Y itself belongs to NP, the problem Y is called NP-*complete*. For example, Vertex Cover is an NP-complete problem. Note that to show the NP-hardness of a problem Y it suffices to give a reduction from *one* NP-hard problem X to Y, because the composition of two polynomial-time reductions is again a polynomial-time reduction.

NP-complete problems are, by definition, the "most difficult" problems of the class NP, and no algorithms are known that solve these problems efficiently. In fact, it is very unlikely that a polynomial-time algorithm for any NP-hard problem can ever be found, because this would immediately imply that *all* problems in NP (in particular, all NP-complete problems) could be solved in polynomial time, meaning that NP = P. There are thousands of NP-complete problems, and they arise in all areas of life [GJ79, Pap97]; the question whether NP = P is one of the seven "Millennium Prize Problems" named by The Clay Mathematics Institute [Cla09].

Function Problems. So far, we have only considered decision problems. However, in practical applications one often does not only want to know whether a problem instance has a solution (that is, whether it is a *yes*-instance), but one is interested in finding such a solution. For example, in the case of Vertex Cover, the following problem definition could be more useful for many applications.

Vertex Cover II
Input: An undirected graph $G = (V, E)$ and a nonnegative integer k.
Task: Find a vertex cover C for G that consists of at most k vertices, or report that no such vertex cover exists.

Problems of this kind are called *function problems*. Since the definitions of P and NP do not apply to these problems, there are specific complexity classes for function problems: In particular, the class of problems where the task is to compute a certificate for an instance of a decision problem X from NP is called FNP (Vertex Cover II would be a typical representative for this class). If a problem in FNP can be solved deterministically in polynomial time, that is, a certificate for a given instance can be computed in polynomial time if existing, then it belongs to the class FP.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Many (see also [BG94]) function problems in FNP are not really "more difficult" than their corresponding decision versions in NP, a fact, which is called *self-reducibility*: A certificate for an instance x of a decision problem X from NP can often be constructed by calling polynomially many times an algorithm for X answering only with *yes* or *no*. For example, to construct a vertex cover of size k for a graph G, try every vertex $v \in V$ and ask whether there is a vertex cover of size $k-1$ for the graph G with v deleted. If for a vertex v the answer is *yes*, then put v into the solution to be constructed and continue by constructing a vertex cover of size $k-1$ for the graph G with v deleted.[4,5]

Of course, if a Turing machine M solves a function problem from FNP, then M can also be used to solve the corresponding decision problem from NP: just check whether M has returned a certificate or not. Function problems whose Turing machines can be used in this way to solve NP-hard decision problems are usually called NP-hard (although the term "NP-hard" was originally defined for decision problems only[6]) and we cannot hope to find polynomial-time algorithms for them.

Optimization Problems. The last sort of problems we will introduce are *optimization problems*. Here the task usually consists of finding a solution of minimum or maximum cost—a so-called *optimal solution*— from a set of possible solutions—the so-called *feasible* solutions—for a given problem instance. A problem is called a *minimization problem* if a solution of minimum cost is required, and a *maximization problem* if a solution of maximum cost is required. The description of what exactly a feasible solution is and how its cost has to be computed belongs to the definition of the problem. The optimization version of Vertex Cover, for example, is defined as follows.

[4]If G is a *yes*-instance then there is always a vertex v such that G with v deleted has a vertex cover of size $k-1$.

[5]The class containing all function problems that can be solved by calling polynomially many times an algorithm for a problem in NP is called $\mathrm{FP}^{\mathrm{NP}}$ (in analogy to P^{NP}); it contains, in particular, the problem Vertex Cover II.

[6]There is a specific notion of hardness for function problems, which directly corresponds to NP-hardness for decision problems and which is based on a special type of reduction: A function problem X is called (polynomial-time) reducible to a function problem Y if there is a polynomial-time computable function Φ that maps every instance x of X to an instance $y = \Phi(x)$ of Y, and a polynomial-time computable function that takes as input a solution for y and outputs a solution for x. A problem Y is called FNP-*hard* if every problem $X \in \mathrm{FNP}$ can be reduced to Y. This notion allows to define FNP-*complete* problems, that is, FNP-hard problems in FNP, as the most difficult problems in FNP. Using the term "NP-hard" for function problems neglects that a reduction for decision problems has to map *yes*-instances to *yes*-instances and *no*-instances to *no*-instances. Mapping *yes*-instances to *no*-instances and *no*-instances to *yes*-instances does not conform to the definition of a polynomial-time reduction for a decision problem, and the class NP, which is defined as the class of problems with polynomial-time checkable witnesses for *yes*-instances, is not closed under such reductions.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Vertex Cover III
Input: An undirected graph $G = (V, E)$.
Task: Find a minimum-size vertex cover C for G.

Here, the set of feasible solutions consists of all vertex covers for the input graph, and the cost of a solution is simply the number of its vertices. Formally, an optimization problem X consists of a tuple $(I_X, \mathrm{SOL}_X, \mathrm{cost}_X, \mathrm{type}_X)$ consisting of a set I_X of instances, a function SOL_X assigning to every instance $x \in I_X$ a set $\mathrm{SOL}_X(x)$ of feasible solutions, a function cost_X determining the cost of every solution in $\mathrm{SOL}_X(x)$ for all instances $x \in I_X$, and a string $\mathrm{type}_X \in \{\mathrm{MIN}, \mathrm{MAX}\}$ specifying whether X is a minimization problem or a maximization problem. We just mention in passing that optimization problems can also occur in other "flavors" than the one described so far: Instead of asking for the computation of an optimal solution, the task can be to compute only the cost of an optimal solution or to decide whether the cost of an optimal solution equals a given value. Here we will concentrate on the kind of optimization problems where one has to compute an optimal solution.

The complexity class that contains the optimization problems of our interest is NPO. This class contains all optimization problems $X = (I_X, \mathrm{SOL}_X, \mathrm{cost}_X, \mathrm{type}_X)$ where for every $x \in I_X$ the following three statements apply: The size of every solution in $\mathrm{SOL}_X(x)$ is polynomial in the size of x, one can verify in polynomial time that a solution indeed belongs to $\mathrm{SOL}_X(x)$, and the function cost_X is computable in polynomial time. The class PO contains those problems from NPO that can be solved deterministically in polynomial time, that is, those problems where an optimal solution for a given instance can be found in polynomial time.

The class NPO is closely connected to the class NP: The decision problem where, given an instance $x \in I_X$ of a minimization (maximization) problem $X \in \mathrm{NPO}$ and a number k, the task is to decide whether x has a feasible solution $s \in \mathrm{SOL}_X(x)$ whose cost is at most (at least) k, clearly belongs to NP and is called the "decision problem in NP that corresponds to X."[7] Many optimization problems $X \in \mathrm{NPO}$ can be solved by calling polynomially many times an algorithm for the corresponding decision problem in NP. In the case of Vertex Cover III, for example, one can first compute the size of a minimum vertex cover for the given graph by repeatedly calling an algorithm for Vertex Cover and setting k to $1, 2, 3, \ldots$ until it answers with *yes*. The vertex cover can then be constructed by using the approach described in the paragraph about function problems.[8] The other way round, a Turing machine M that solves Vertex Cover III can also be used to solve the NP-hard decision problem Vertex

[7] In contrast, the problem of deciding whether the *optimal* solution for a x has cost *exactly* k is not necessarily in NP; however, for $X \in \mathrm{NPO}$ it always belongs to a class called DP, which is a subset of P^{NP} [Pap94].

[8] Although both problems Vertex Cover II and Vertex Cover III can be solved by using polynomially times an algorithm for Vertex Cover (both Vertex Cover II and Vertex Cover III, as well as all problems in NPO, belong to $\mathrm{FP}^{\mathrm{NP}}$), there is a slight difference between the complexities of these two problems: While in the case of Vertex Cover II (and, by

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Cover: just check whether the vertex cover returned by M has a size of at most k. Such optimization problems are called NP-hard,[9] and, again, we cannot hope to find polynomial-time algorithms for them.

The fact that algorithms for NP-complete problems can be used to solve problems in FNP and in NPO and vice versa is also expressed in the following notion: Vertex Cover, Vertex Cover II, and Vertex Cover III, as well as all other NP-complete decision problems and all "NP-hard" problems in FNP and NPO are called *polynomially equivalent*, meaning that, on the one hand, they are not expected to be polynomial-time solvable, but, on the other hand, if one of these problems turns out to be polynomial-time solvable, then the other problems are polynomial-time solvable, too.[10]

1.3.2 Ways to Cope with NP-hard Problems

As mentioned in the previous section, it seems impossible to find exact algorithms that solve NP-hard problems efficiently, that is, in polynomial time. However, many problems of high practical relevance are NP-hard [GJ79, Pap97], and so there have been developed several ways to cope with these problems. The most common concepts to this end are

- heuristics,
- approximation algorithms, and
- fixed-parameter algorithms.

For the sake of completeness, we also mention the concept of

- randomized algorithms,

which at first sight also seems to be an appropriate tool to handle problems that are too difficult to be solved exactly within a reasonable amount of time. However, from the theoretical point of view, randomized algorithms could so far not obtain satisfying results for exactly solving NP-hard problems.

Roughly speaking, heuristics are algorithms that work well for many instances of a problem, but not for all. Approximation algorithms are algorithms for optimization problems that do not solve the problem exactly, but compute a solution

definition, all problems in FNP) the correctness of a solution can be checked deterministically in polynomial time, we do not have any polynomial-checkable "witness" for the correctness of a solution for Vertex Cover III, because it is not clear how to prove its minimality (see footnote 7 on page 16). In fact, the existence of such a witness would imply that the complexity classes NP and coNP (the class of problems having polynomial-time checkable witnesses for *no*-instances) coincide, which is considered unlikely.

[9] There are other notions of hardness (FNP-hardness, $\mathrm{FP}^{\mathrm{NP}}$-hardness) that would be more meaningful here, see footnote 6 on page 15.

[10] This polynomial equivalence also includes all problems that are "complete" for the classes coNP, DP, P^{NP}, and $\mathrm{FP}^{\mathrm{NP}}$.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

whose cost is never too far away from the optimum. Fixed-parameter algorithms are exact algorithms whose running time is measured not only in the input size, but also in some other property of the instance called the parameter, and whose running time is small for instances with small parameter values. Randomized algorithms are algorithms where either the output may be wrong with a certain probability or the running time is random.

Sections 1.3.3–1.3.6 will describe each of these four concepts. In the subsequent chapters of the thesis, we will then concentrate on approximation algorithms and fixed-parameter algorithms.

1.3.3 Heuristics

For many problems, there are algorithms that seem to be clever, but cannot guarantee to solve every problem instance correctly or within a predictable amount of time. Such algorithms are called *heuristics* (see also [GK03, MF04]). In the case of Vertex Cover III, for example, it seems to be a good idea to repeatedly take a vertex that covers a maximum number of edges, which leads to the following "greedy" heuristic:

```
1: C := ∅; // the vertex cover to be constructed
2: while G contains an edge: {
3:     let v be a vertex of maximum degree;
4:     C := C ∪ {v};
5:     delete v and all edges incident to v from G; }
6: return C;
```

However, for many instances of Vertex Cover III this algorithm does not output an optimal solution. Even worse, although for many instances the algorithm outputs a solution whose cost is not far from the optimum value, there are also instances for which not even this is the case: For every constant c, there is an instance of Vertex Cover III where the output of the algorithm consists of more than c times the number of vertices compared to an optimal solution.[11]

Typically, heuristics are easy to understand and to implement. Moreover, for many practical applications there exist heuristics that produce good results for most of the occurring problem instances, despite their bad worst-case behavior.

1.3.4 Approximation Algorithms

In the case of an NP-hard optimization problem one cannot hope to find a polynomial-time algorithm solving the problem exactly. However, for some of these problems there exist algorithms that run in polynomial time and, although

[11] More exactly, for every positive integer k there exists an instance of Vertex Cover III with $|V| = O(k \ln(k))$ such that the output of the algorithm consists of at least $k \cdot (\ln(k) - 1)$ vertices, but k vertices would be optimal (see [PS98]).

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

they do not find an optimal solution for every input instance, that always find at least an approximate solution. This means that even in the worst case the ratio between the cost of the solution produced by the algorithm and the cost of an optimal solution is bounded. Consider the following algorithm for Vertex Cover III.

1: $C := \emptyset$; // the vertex cover to be constructed
2: **while** G contains an edge: {
3: let $\{v, w\}$ be an arbitrary edge in G;
4: $C := C \cup \{v, w\}$;
5: delete v and w and all edges incident to v or w from G; }
6: **return** C;

This algorithm seems to be much less clever than the heuristic in the paragraph before (and for many instances it will produce worse results—that is, larger vertex covers—than the heuristic). However, for *any* instance of Vertex Cover III the output of the algorithm does not contain more than twice the number of vertices than an optimal solution, which means that in the worst-case this algorithm performs better than the heuristic. To see this, consider the set of all those edges that have been selected in line 3 of the algorithm: No two of these edges have a common endpoint, and, therefore, every optimal solution must contain at least one endpoint of each of these edges, whereas the algorithm takes both.

More exactly, for an algorithm A that approximately solves an optimization problem X we define the *approximation factor* of A for an instance x of X as

$$\rho_A(x) = \frac{\text{cost}_X(A(x))}{\text{opt}_X(x)},$$

where $A(x)$ is the solution found by the algorithm and $\text{opt}_X(x)$ is the cost of an optimal solution for x. The algorithm A is called a polynomial-time *factor-r approximation* algorithm (or, more general, a *constant-factor approximation*) if there exists a constant $r > 0$ such that for all $x \in I_X$ it holds that

$$\rho_A(x) \begin{cases} \leq r & \text{if } X \text{ is a minimization problem} \\ \geq r & \text{if } X \text{ is a maximization problem.} \end{cases}$$

The number r is then called the *approximation factor* of A. The algorithm above, for example, is a factor-2 approximation algorithm for Vertex Cover III. Similarly, if there is a function f such that for all $x \in I_X$ it holds that $\rho_A(x) \leq f(|x|)$ for minimization problems and $\rho_A(x) \geq f(|x|)$ for maximization problems, the algorithm A is called a *factor-f approximation*, and f is called the approximation factor of A.[12]

[12]For example, one can prove [Joh74, Lov75] that the heuristic for Vertex Cover III presented in Section 1.3.3 is a factor-$(\ln(d)+1)$ approximation algorithm, where d is the maximum degree of a vertex in the input graph (see also [Hoc97]).

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

While for some optimization problems not even constant-factor approximations are known, some other problems can be approximated arbitrarily closely. In such a case, a family of polynomial-time algorithms A_r, $r \in \mathbb{Q}$, for an optimization problem X is called a *polynomial-time approximation scheme (PTAS)* if for every given approximation factor r the algorithm A_r is a factor-r approximation algorithm for X.[13]

All optimization problems for which a constant-factor approximation exists form the class APX (which is basically equivalent [KMSV98] to the independently introduced class MAXSNP [PY91]), and all problems for which a PTAS exists, form the class PTAS.

Unfortunately, there are optimization problems for which no PTAS or even no constant-factor approximation could ever be found. In order to combine such hard to approximate problems into classes of their own, there have been defined notions of hardness (with respect to the approximability) for optimization problems. In analogy to the concept of NP-completeness for decision problems, these hardness predicates are defined by using reductions between problems. While in the case of decision problems a reduction from a problem X to a problem Y allows to solve X by using an algorithm for Y, a reduction as we need it here allows to approximate an optimization problem X by using an approximation algorithm for Y. If a problem X that is assumed to be hard to approximate can be reduced to a problem Y, then Y must also be hard to approximate. There have been defined several different variants of such so-called *approximation-preserving* reductions (see [AP06] for an overview), but the basic idea is always the same: An approximation-preserving reduction from an optimization problem X to an optimization problem Y consists of a polynomial-time computable function Φ_1 that maps every instance x of X to an instance $y = \Phi_1(x)$ of Y, and a polynomial-time computable function Φ_2 that takes as input a feasible solution for y and outputs a feasible solution for x. Moreover, if A is a constant-factor approximation for Y, then computing $y = \Phi_1(x)$, calling A on y, and computing $\Phi_2(A(y))$ leads to a constant-factor approximation for X.

A problem Y is called APX-*hard* (NPO-*hard*) if every problem $X \in$ APX ($X \in$ NPO) can be reduced to Y with an approximation-preserving reduction. If, in addition, the problem Y itself belongs to APX (NPO), the problem Y is called APX-*complete* (NPO-*complete*). Provided that P $\neq$ NP, it holds [ACG+99] that

$$\text{PO} \subsetneq \text{PTAS} \subsetneq \text{APX} \subsetneq \text{NPO},$$

and, consequently, under this assumption NPO-complete optimization problems are those problems in NPO where no approximation algorithms at all exist, and APX-complete optimization problems are those problems where constant-factor

[13] If the running times are not only polynomial in the input size, but also in $1/(|1-r|)$, the family of algorithms is called a *fully polynomial-time approximation scheme* (*FPTAS*).

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

approximations, but no PTAS exist.[14] To show that an optimization problem Y is APX-hard (NPO-hard), is suffices to give an approximation-preserving reduction from an APX-hard (NPO-hard) problem X to Y.

We close this section with pointing out that the existence of a constant-factor approximation algorithm for a minimization (maximization) problem does not imply the approximability for the corresponding "dual" maximization (minimization) problem: For example, Vertex Cover III can be approximated with a factor of 2, but the problem of finding a maximum-cardinality independent set (see Section 1.3.1), which is the "dual" of Vertex Cover III because the vertices not belonging to a maximum independent set form a minimum vertex cover, can probably not be approximated within a factor of $O(|V|^{(1-\epsilon)})$ for any $\epsilon > 0$ [Hås99].

For more details about approximation algorithms and approximation-related complexity we refer to [ACG+99, Pap94, JM08, Vaz01].

1.3.5 Fixed-Parameter Algorithms

When exact solutions are needed (in particular, in the case of decision problems), heuristics and approximations cannot be used. However, *parameterized complexity* analysis [DF99, FG06, Nie06] can help here. The main idea is to exploit the fact that the "classical" complexity, which measures the running time of an algorithm only with respect to the input size and which always considers the worst case, is somewhat coarse-grained: Vertex Cover, for example, can be solved "fast" if k is very small compared to the size of G—just try all $\binom{|V|}{k}$ possibilities of selecting k vertices from V. So it would be natural to measure the running time of an algorithm for Vertex Cover not only in the input size (which is dominated by the size of G), but also in the value of k. However, using the means of expression of "classical" complexity theory, one could only state that "the restricted version of Vertex Cover, where k is bounded from above by a constant, is in P"—a quite imprecise statement, because it says nothing about the relation between k and the running time needed to solve the problem. In contrast, parameterized complexity theory offers the mathematical framework, including complexity classes and reductions, for a fine-grained "multidimensional" running-time analysis.

In the framework of parameterized complexity, a parameterization of a decision problem X is a polynomial-time computable function κ that assigns to every problem instance a nonnegative integer; the pair (X, κ) is called a *parameterized problem*. The running time of an algorithm is then viewed as a function of two quantities: the size $|x|$ of the problem instance *and* the parameter $\kappa(x)$. A

[14] There are also classes (log-APX, poly-APX, exp-APX) that lie "between" APX and NPO: The class log-APX, for example, contains those minimization (maximization) problems that can be approximated with a factor of $O(\log(|x|))$ (a factor of $O(1/\log(|x|))$); the so-called log-APX-*complete* problems belong to log-APX, but have no constant-factor approximation unless P = NP.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

possible parameterization for Vertex Cover, for example, could be the function that maps every problem instance (G, k) to the number k, meaning that the parameter is the size of the desired vertex cover.

As mentioned above, Vertex Cover could be solved by trying all possibilities of selecting k vertices from V. The running time of this algorithm would be $\binom{|V|}{k} \cdot n^{O(1)} \leq |V|^k \cdot n^{O(1)}$. Hence, for every fixed k the problem Vertex Cover is solvable in polynomial time, where the degree of the polynomial depends on k. Parameterized problems of this kind are comprised in the complexity class XP: This class consists of all parameterized problems (X, κ) for which there exists a computable function that assigns to every positive integer k a polynomial p_k such that an instance x of X can be solved in $p_{\kappa(x)}(|x|)$ time. Clearly, if one can show that there is a constant k such that the problem X, restricted to instances with $\kappa(x) \leq k$, is NP-hard, then (X, κ) does not belong to XP unless P = NP.

For many practical applications where large problem instances occur, a running time of the form $|x|^{\kappa(x)}$ is too slow even for quite small parameters, because the degree of the polynomial depends on the parameter. A running time whose exponential part depends only on the parameter, but not on the input size would be much more desirable. Thus, a parameterized problem (X, κ) is called *fixed-parameter tractable (FPT)* (equivalently: "X is fixed-parameter tractable with respect to the parameter κ") if there exists an algorithm for X that runs in $f(\kappa(x)) \cdot |x|^{O(1)}$ time, where f is a computable (usually exponential) function only depending on $\kappa(x)$. We call such an algorithm an *FPT algorithm*, and a running time of the form $f(\kappa(x)) \cdot |x|^{O(1)}$ is called *FPT time*. The complexity class that contains all fixed-parameter tractable parameterized problems is denoted by FPT.

The problem Vertex Cover, for example, is fixed-parameter tractable with respect to the parameter $\kappa((G, k)) = k$ because it can be solved in $2^k \cdot n^{O(1)}$ time with the following recursive algorithm:

1: **if** G contains no edge: **return** *yes*;
2: **if** $k = 0$: **return** *no*;
3: let $\{v, w\}$ be an arbitrary edge in G;
4: test recursively if there is a size-$(k-1)$ vertex cover for G with v deleted;
5: if so, **return** *yes*; // G has a size-k vertex cover that contains v
6: test recursively if there is a size-$(k-1)$ vertex cover for G with w deleted;
7: if so, **return** *yes*; // G has a size-k vertex cover that contains w
8: **return** *no*;

Currently, the best known algorithm for Vertex Cover with parameter k runs in $O(1.2738^k + kn)$ time [CKX06], such that even large instances of Vertex Cover with "quite big" values of k can be solved exactly.

An important tool to prove fixed-parameter tractability on the one hand and to tackle problem instances in practice on the other hand is polynomial-time data preprocessing by so-called data reduction rules (see [GN07]). A *data reduction*

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

rule is a polynomial-time algorithm that takes as input a problem instance x and outputs an instance x' such that x' is a *yes*-instances iff x is a *yes*-instance; moreover, since the purpose of the data preprocessing is to decrease the size of the problem instance, the instance x' typically has $|x'| \leq |x|$ and $\kappa(x') \leq \kappa(x)$. An instance to which none of a given set of data reduction rules applies is called *reduced* with respect to these rules. In certain cases, the performance of a set of data reduction rules can be proven: If every problem instance x can be transformed into a reduced instance x' with $|x'| \leq f(|x|)$ and $\kappa(x') \leq g(\kappa(x))$ for some computable functions f, g, then the data reduction rules are called a *kernelization* and x' is called a *problem kernel*. That is, the size of a problem kernel x' is bounded from above by a function f depending only on $\kappa(x)$;[15] a kernelization, hence, is a polynomial-time data preprocessing with guaranteed worst-case performance. If there is a kernelization for a decidable parameterized problem (X, κ), then (X, κ) is clearly fixed-parameter tractable:[16] To solve X in $f(\kappa(x)) \cdot |x|^{O(1)}$ time, first apply the data reduction rules and then run a brute-force algorithm for X on the reduced instance.

As an example for a kernelization, consider the following data reduction rules for Vertex Cover:

Rule 1: If there is an isolated vertex, delete it.

Rule 2: If there is a vertex v with more than k neighbors, delete v and all edges incident to v from G, and decrease k by 1.

Rule 3: If neither Rule 1 nor Rule 2 can be applied and there are more than $k + k^2$ vertices or more than k^2 edges, report that there is no vertex cover of size at most k.

Clearly the first two rules are correct, that is, they never transform a *yes*-instance into a *no*-instance or vice versa: Isolated vertices can be deleted, because it never makes sense to take them into a vertex cover, and vertices with more than k neighbors have to be selected into any vertex cover of size at most k and, therefore, one can just delete them and decrease k. To see the correctness of Rule 3, consider an instance (G, k') that is reduced with respect to Rules 1 and 2: The reduced graph G' contains only vertices of degree at most k, and, hence, if there is a size-k vertex cover C for G', then G' can contain at most k^2 vertices outside C and at most k^2 edges. Rule 3 implies that any reduced instance consists of at most $k + k^2$ vertices and k^2 edges, meaning that Vertex Cover with parameter k admits a quadratic problem kernel.[17]

[15] If, in addition, $\kappa(x') \leq \kappa(x)$, then the problem kernel is called *proper* [AF06].

[16] The reverse direction of this statement also holds: If there is an algorithm A that solves a problem (X, κ) in $f(\kappa(x)) \cdot |x|^c$ time for some constant c, then the following "data reduction rule" yields a problem kernel of size $f(\kappa(x))$: If $f(\kappa(x)) \leq |x|$, then solve x in $f(\kappa(x)) \cdot |x|^c \leq |x|^{c+1}$ time by using A. Otherwise return x, whose size is at most $f(\kappa(x))$.

[17] There exist other reduction rules that even yield a linear number of vertices for Vertex Cover with parameter k [ACF+04, AFLS07, BG93, CFJ04, CKJ01, Fel03, Khu02, NT75].

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

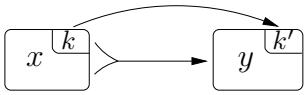


Figure 1.5: Parameterized reduction from (X, κ_1) to (Y, κ_2). An instance x with parameter $k := \kappa_1(x)$ is transformed within $g(k) \cdot |x|^{O(1)}$ time into an instance y with parameter $k' := \kappa_2(y)$. The parameter k' must only depend on k, but not on $|x|$. In contrast, the size of y is only bounded by the time needed for the reduction, that is, by $g(k) \cdot |x|^{O(1)}$.

As a second example for a problem with a kernel, consider the problem of separating points in the plane with axis-parallel lines—this problem was introduced in Section 1.1. Formulated as a decision problem, the question is whether one can separate all given points by inserting at most k axis-parallel lines. The argumentation that leads to the problem kernel is quite simple: With k lines, the plane can be divided into at most $\lfloor k/2 \rfloor \cdot \lceil k/2 \rceil$ areas. Hence, if there are more than $\lfloor k/2 \rfloor \cdot \lceil k/2 \rceil$ points, the answer must be *no*. The point separation problem, therefore, has a quadratic kernel with respect to the parameter $k =$ "maximum number of line insertions."

Unfortunately, there are also parameterized problems that seem not to be fixed-parameter tractable. In analogy to the concept of NP-hardness in the "classical" complexity theory, there is a hardness predicate for parameterized problems, which is based on reductions between parameterized problems. The idea is, in analogy to the definition of NP-hardness, to define a class that contains many of these "difficult" problems and to call a problem Y hard if it is "at least as hard" as all problems in this class, meaning that all problems in this class can be reduced to Y and thus be solved by any algorithm for Y. While in the case of decision problems in "classical" complexity theory a polynomial-time reduction from a problem X to a problem Y allows to solve X in polynomial time by using a (fictive) polynomial-time algorithm for Y, a reduction from a parameterized problem (X, κ_1) to a parameterized problem (Y, κ_2) as we need it here has to allow for solving a problem X in FPT time by using a (fictive) FPT algorithm for Y. Hence, a parameterized problem (X, κ_1) is *parameterized reducible* to a parameterized problem (Y, κ_2), denoted by $(X, \kappa_1) \leq_{\text{FPT}} (Y, \kappa_2)$, if there are two computable functions $f, g : \mathbb{N} \to \mathbb{N}$ and an algorithm Φ which transforms an instance x of X into an instance y of Y in $g(\kappa_1(x)) \cdot |x|^{O(1)}$ time such that $\kappa_2(y) = f(\kappa_1(x))$ and y is a *yes*-instance of Y iff x is a *yes*-instance of X. See Figure 1.5 for an illustration.

Note that the polynomial-time reduction from Vertex Cover to Independent Set described in Section 1.3.1 is *not* a parameterized reduction for the parameters "size of the desired vertex cover" and "size of the desired independent set", respectively: The size of the independent set does not only depend on the size of the vertex cover, but also on the number of all vertices in the input

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

graph of the Vertex Cover instance.

The complexity hierarchy that is used for defining the hardness of parameterized problems and that contains many "difficult" parameterized problems is the W-*hierarchy*, which is defined quite unintuitively by using Boolean circuits (see also [Ces03, CM08]) and consists of the classes $\mathrm{W}[1], \mathrm{W}[2], \ldots, \mathrm{W}[\mathrm{Sat}], \mathrm{W}[\mathrm{P}]$ interrelated as follows:

$$\mathrm{FPT} \subseteq \mathrm{W}[1] \subseteq \mathrm{W}[2] \subseteq \ldots \subseteq \mathrm{W}[\mathrm{Sat}] \subseteq \mathrm{W}[\mathrm{P}] \subseteq \mathrm{XP}\,.$$

There is strong evidence that all these subset inclusions are strict, which, in particular, presumably means that there are problems in W[1] that are not fixed-parameter tractable. Thus, a parameterized problem Y to which all problems in W[1] can be reduced with a parameterized reduction is called W[1]-*hard* and assumed not to be fixed-parameter tractable. A W[1]-hard problem that belongs to W[1] is called W[1]-*complete* and belongs to the "most difficult" problems in W[1]. For the other classes of the W-hierarchy, hardness and completeness are defined in complete analogy. For example, the problem Independent Set is W[1]-complete with respect to the parameter $\kappa((G,k)) = k$.

For the same reasons as in the case of NP-hardness and polynomial-time reductions, it suffices to give a parameterized reduction from *one* W[1]-hard parameterized problem X to a parameterized problem Y to show the W[1]-hardness of Y.

More details about FPT algorithms can be found in [Nie06]; for more details about parameterized complexity theory we refer to [DF99, FG06].

1.3.6 Randomized Algorithms

Randomized algorithms are algorithms whose outcome depends on chance. We distinguish two types of such algorithms: *Monte Carlo algorithms*, where the output may be wrong with a certain probability, and *Las Vegas algorithms*, which always produce correct results but where the running time is random. Consider, for example, the following Monte Carlo algorithm for Vertex Cover:

1: $s := 0$; // size of the vertex cover found
2: **while** G contains an edge: {
3: randomly select an edge e;
4: randomly select one of the endpoints of e; let v be this vertex;
5: delete v and all edges incident to v from G; $s := s + 1$; }
6: **if** $s \leq k$ **return** *yes*; **else return** *no*;

Clearly this algorithm runs in polynomial time. Moreover, if the input to this algorithm is a *no*-instance, the algorithm always answers correctly with *no*. If, however, the input is a *yes*-instance, then all we can say is that the probability for a correct answer is at least $(\frac{1}{2})^{|E|}$. Of course, such a behavior is unsatisfying

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

because for large *yes*-instances the probability for a correct answer tends towards zero. Instead, it would be desirable to have an algorithm where the probability for a correct answer is constant for all *yes*-instances. Problems that admit such algorithms are comprised in the complexity class RP: This class contains all decision problems for which there is a polynomial-time algorithm A and a constant c_A such that for every *no*-instance the answer of A is correct and for every *yes*-instance the probability for a correct answer is at least c_A.[18]

Clearly RP $\subseteq$ NP because, as mentioned in Section 1.3.1, every nondeterministic Turing machine can be seen as a randomized algorithm where for every *no*-instance the answer is correct and for every *yes*-instance the probability for a correct answer is greater than zero.[19] However, there are a lot of problems in NP (in particular, the NP-complete problems) that are not known to be in RP. Therefore, randomized complexity classes have been defined that make lower demands on the corresponding algorithms: The class BPP contains all decision problems for which there is a polynomial-time algorithm A and a constant $c_A > 1/2$ such that for every instance the probability for a correct answer is at least c_A. For membership in the even "more weak" class PP it suffices that for every *no*-instance the probability for a correct answer is at least $1/2$ and for every *yes*-instance the probability for a correct answer is greater than $1/2$.

There is also a complexity class for Las Vegas algorithms, which is defined in a very natural way: The class ZPP contains those decision problems for which there is an algorithm that always answers correctly and where for every problem instance the expectation value of the running time is polynomial in the size of the instance.

Clearly ZPP $\subseteq$ RP,[20] RP $\subseteq$ NP $\subseteq$ PP, and RP $\subseteq$ BPP $\subseteq$ PP. One could say that ZPP, RP and BPP contain those problems that can be solved satisfactorily by using randomized algorithms. Unfortunately, it seems that NP-hard problems belong to neither of these classes and, hence, that randomization is useful for tackling NP-hard problems only in combination with heuristics, approximation algorithms or fixed-parameter algorithms.

For more details about randomized algorithms and randomized complexity we refer to [Pap94, MR95, MU05].

[18] Analogously, a problem belongs to the class coRP if for every *yes*-instance the answer of A is correct and for every *no*-instance the probability for a correct answer is at least c_A.

[19] For the same reason, coRP $\subseteq$ coNP.

[20] Indeed, ZPP = RP $\cap$ coRP, see [Pap94].

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Chapter 2

The Consecutive-Ones Property

Chapter 2 provides an overview on the consecutive-ones property. The chapter describes connections to graph theory, algorithms for recognizing the C1P, characterizations of the C1P, and consequences of the C1P appearing in integer linear programs and instances of the problem Set Cover.

2.1 Basic Facts and Definitions

The consecutive-ones property of binary matrices appears in many practical applications, such as scheduling [BOR80, HL06, HM91, KS01, VW62], information retrieval [Kou77], railway optimization [MSW05, MW04, RS04], and computational biology [ABH98, ACE+08, AM96, COR98, GGKS95, LH03, WR00]—some examples have been presented in Section 1.1. Moreover, the C1P has close connections to graph theory (see Section 2.2) and plays an important role in the area of solving (integer) linear programs [HT02, HL06, OR00, OR03a, VW62] (see also Section 2.4). The formal definition of the C1P and some related concepts reads as follows.

Definition 2.1. 1. A *block of 1s* (*block of 0s*) in a row of a binary matrix M is a maximal set of consecutive 1-entries (0-entries) in this row.

2. A binary matrix has the *strong consecutive-ones property* (*strong C1P*) if in every row the 1s appear consecutively, that is, if every row contains at most one block of 1s.

3. A binary matrix has the *consecutive-ones property* (*C1P*) if its columns can be permuted in such a way that the resulting matrix has the strong C1P.

4. If an ordering for the columns of a binary matrix yields the strong C1P, it is called a *C1-ordering*.

See Figure 2.1 for examples of matrices with and without the C1P. The terms introduced in Definition 2.1 can be defined analogously for 1-entries appearing

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

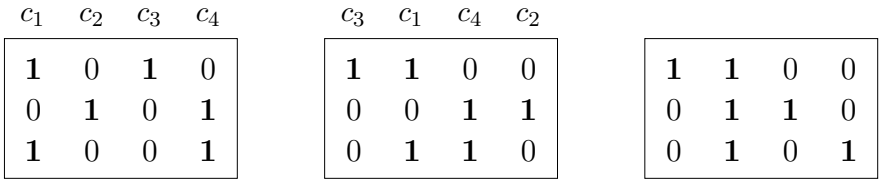


Figure 2.1: Example for the C1P: The matrix on the left has the C1P because by permuting its columns (labeled with c_1–c_4) one can obtain the matrix shown in the middle where the 1s in each row appear consecutively. The matrix on the right, in contrast, does not have the C1P [Tuc72].

consecutively in the columns of a matrix instead of the rows: If in every column of a binary matrix M the 1s appear consecutively, then M has the *strong C1P for columns*. If M's rows can be permuted such that the strong C1P for columns is obtained, then M has the *C1P for columns*. In order to avoid confusions between the C1P for columns and the C1P as defined in Definition 2.1, we sometimes write *strong C1P for rows* and *C1P for rows* to emphasize that the terms of Definition 2.1 are meant.

A property that is very similar to the C1P but less restrictive is called the *circular-ones property*: Here one imagines the matrix as wrapped around a vertical cylinder and demands that, possibly after some column permutations, in every row the 1s appear consecutively on the cylinder (which implies that the 0s also appear consecutively). In Chapter 4 we will use this property as an intermediate concept for dealing with the harder to achieve C1P. The formal definition reads as follows.

Definition 2.2. 1. A binary matrix has the *strong circular-ones property* (*strong Circ1P*) if in every row the 1s appear consecutively or the 0s appear consecutively (or both).

2. A binary matrix has the *circular-ones property* (*Circ1P*) if its columns can be permuted in such a way that the resulting matrix has the strong Circ1P.

3. If an ordering for the columns of a binary matrix yields the strong Circ1P, then it is called a *Circ1-ordering*.

See Figure 2.2 for an example. When imagining a matrix as wrapped around a vertical cylinder, it makes no sense to declare one of its columns as the "leftmost" or "rightmost" column. Therefore, we say that two column orderings are *shift-equivalent*[1] if one column ordering can be obtained from the other by repeatedly taking the column that is currently placed at the leftmost position and moving it

[1]The term "shift-equivalent" may have a different meaning in other publications and other contexts.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

A:

c_1	c_2	c_3	c_4
1	0	**1**	0
0	**1**	**1**	**1**
1	0	0	**1**

B:

c_1	c_3	c_2	c_4
1	**1**	0	0
0	**1**	**1**	**1**
1	0	0	**1**

C:

c_1	c_2	c_3	c_4
1	**1**	0	0
0	**1**	**1**	0
0	**1**	0	**1**

D:

c_1	c_2	c_3	c_4
1	**1**	0	0
0	**1**	**1**	0
1	0	**1**	0

Figure 2.2: Example for the Circ1P: The matrix A has the Circ1P because by permuting its columns (labeled with c_1–c_4) one can obtain the matrix B where in each row the 1s or the 0s appear consecutively. The matrices C and D in contrast, do not have the Circ1P [Tuc71, Tuc72].

from there to the rightmost position. As an example, for any matrix consisting of 13 columns, the column orderings

$$c_1, \ldots, c_{13} \tag{2.1}$$

and

$$c_7, c_8, \ldots, c_{12}, c_{13}, c_1, c_2, \ldots, c_5, c_6 \tag{2.2}$$

are shift-equivalent. The relation "shift-equivalent" is obviously an equivalence relation on the column orderings for a matrix; we call its equivalence classes *circular (column) orderings*. The column orderings displayed in (2.1) and (2.2), hence, represent the same circular column ordering. Intuitively, the the circular column ordering of a matrix M describes the order of M's columns when M is imagined as wrapped around a vertical cylinder; it defines for every column c of M a predecessor and a successor, but it does not declare any of M's columns as the "leftmost" or "rightmost" column.

Definition 2.3. If a circular column ordering of a binary matrix M is the equivalence class of at least one C1-ordering for M's columns, it is called a *C1-circular ordering*. If it is the equivalence class of at least one Circ1-ordering for M's columns, it is called a *Circ1-circular ordering*.

As an example, the circular column ordering $[c_1, c_4, c_2, c_3]$ is a C1-circular ordering for the columns of the matrix on the left of Figure 2.1: The circular ordering $[c_1, c_4, c_2, c_3]$ is identical to the circular ordering $[c_3, c_1, c_4, c_2]$, and the column ordering c_3, c_1, c_4, c_2 is a C1-ordering. Analogously, the circular column orderings $[c_3, c_2, c_4, c_1]$ and $[c_1, c_3, c_2, c_4]$ for the columns of matrix B in Figure 2.2 are identical and form a Circ1-circular ordering.

We use the term "shift-equivalent" not only for column orderings, but also for matrices: Two matrices are shift-equivalent if one can be obtained from the other by repeatedly taking the column that is currently placed at the leftmost position and moving it from there to the rightmost position.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

There exist several characterizations for matrices having the C1P (see Sections 2.2 and 2.3). Together with the following theorem due to Tucker [Tuc71], these characterizations can also be used to recognize matrices with the Circ1P. We will make use of the theorem in Chapter 4.

Theorem 2.1 ([Tuc71, Theorem 1])**.** *Form the matrix M' from a matrix M by complementing all rows with a 1 in the first column of M. Then, M has the Circ1P if and only if M' has the C1P.*

The following corollary is a direct consequence of Theorem 2.1.

Corollary 2.1. *Let M be an $m \times n$ matrix and let j be an arbitrary integer with $1 \leq j \leq n$. Form the matrix M' from M by complementing all rows with a 1 in the jth column of M. Then, M has the Circ1P if and only if M' has the C1P.*

To see the correctness of the corollary, one just has to apply Theorem 2.1 to the matrix that is shift-equivalent to M and whose leftmost column is the jth column of M. An example for Corollary 2.1 can be seen in Figure 2.2: Complementing in one of the matrices C and D all rows with a 1 in column c_4 yields the matrix D, which does not have the C1P (this can easily be seen by considering the columns c_1–c_3); hence, the matrices C and D do not have the Circ1P.

Corollary 2.1 implies the following conclusion.

Corollary 2.2. *Let M be a 0/1-matrix and M' be the matrix obtained by inserting a column that contains only 0s to M. Then the following statements are equivalent.*

1. *M' has the Circ1P.*
2. *M' has the C1P.*
3. *M has the C1P.*

2.2 Graph Classes and the C1P/Circ1P

A description of the C1P would be incomplete if the close relationship to some elementary graph classes was not mentioned: Matrices can be represented by graphs and vice versa, and, therefore, the C1P is related to certain properties of graphs. Many findings can be transferred from matrices to graphs or from graphs to matrices. For example, the task of recognizing matrices with the C1P can easily be reduced to recognizing so-called interval graphs. Also several characterizations for matrices with the C1P are formulated in terms of graph properties, or they are derived from characterizations for certain graph classes. In particular, in Chapter 4 we will use the connection between sparse matrices having the C1P on the one hand and certain kinds of graphs on the other hand to obtain hardness

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

proofs, algorithms and problem kernels for submatrix problems. Moreover, a characterization due to Tucker [Tuc72] for matrices with the C1P, which is a direct consequence of a characterization (also due to Tucker [Tuc72]) for so-called convex bipartite graphs, will be useful in Chapter 3 for finding submatrices that conflict with the C1P as well as in Chapter 4 for eliminating such submatrices.

We first give a short overview over some elementary graph classes that are closely related to the C1P or the Circ1P, and then describe Tucker's characterization for matrices with the C1P.

Graph classes closely related to the C1P or the Circ1P. Given a graph G, there are several "natural" ways to map G to a matrix that represents all information about G. The following definition describes the most common types of such matrices that represent graphs. (The half adjacency matrix of a bipartite graph was already introduced in Section 1.2. Nevertheless, in order to provide a list of all relevant matrix types that are related to graphs, Definition 2.4 contains the definition of the half adjacency matrix again.)

Definition 2.4. Let $G = (V, E_G)$ be a graph with $V = \{v_1, \ldots, v_n\}$ and $E_G = \{e_1, \ldots, e_m\}$, and let $H = (V_1, V_2, E_H)$ be a bipartite graph with $V_1 = \{u_1, \ldots, u_{n_1}\}$ and $V_2 = \{w_1, \ldots, w_{n_2}\}$.

1. The *adjacency matrix* of G is the symmetric $n \times n$ binary matrix M with $m_{i,j} = 1$ if and only if $\{v_i, v_j\} \in E_G$.

2. The *augmented adjacency matrix* of G is the matrix obtained from G's adjacency matrix by setting the entries of the main diagonal to 1.

3. The *edge-vertex incidence matrix* of G is the $m \times n$ binary matrix M with $m_{i,j} = 1$ if and only if v_j is an endpoint of e_i. The transpose of the edge-vertex incidence matrix is called the *vertex-edge incidence matrix* of G.

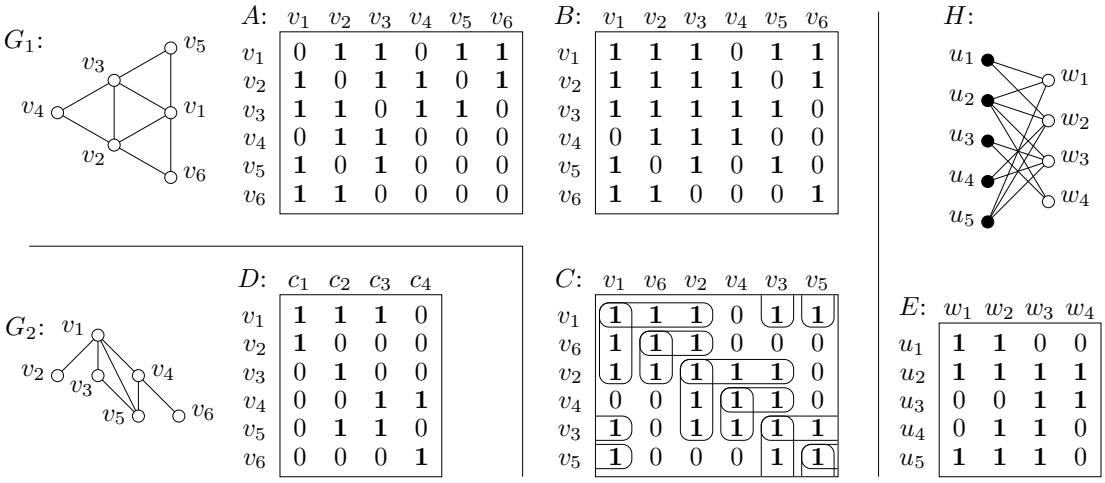
4. Let $c_1, \ldots, c_k$ be the maximal cliques of G. The *maximal clique matrix* (also called *vertex-clique incidence matrix*) of G is the $n \times k$ binary matrix M with $m_{i,j} = 1$ if and only if v_i belongs to c_j.

5. The *half adjacency matrix* of H is the $n_1 \times n_2$ binary matrix M with $m_{i,j} = 1$ if and only if $\{u_i, w_j\} \in E_H$.

Figures 2.3 and 2.6 show the matrix types introduced in Definition 2.4. Clearly, a matrix M is the half adjacency matrix of a bipartite graph H iff H is the representing graph of M.

Some elementary graph classes that are directly related to the properties C1P and Circ1P are defined as follows.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.



A:	v_1	v_2	v_3	v_4	v_5	v_6
v_1	0	**1**	**1**	0	**1**	**1**
v_2	**1**	0	**1**	**1**	0	**1**
v_3	**1**	**1**	0	**1**	**1**	0
v_4	0	**1**	**1**	0	0	0
v_5	**1**	0	**1**	0	0	0
v_6	**1**	**1**	0	0	0	0

B:	v_1	v_2	v_3	v_4	v_5	v_6
v_1	**1**	**1**	**1**	0	**1**	**1**
v_2	**1**	**1**	**1**	**1**	0	**1**
v_3	**1**	**1**	**1**	**1**	**1**	0
v_4	0	**1**	**1**	**1**	0	0
v_5	**1**	0	**1**	0	**1**	0
v_6	**1**	**1**	0	0	0	**1**

D:	c_1	c_2	c_3	c_4
v_1	**1**	**1**	**1**	0
v_2	**1**	0	0	0
v_3	0	**1**	0	0
v_4	0	0	**1**	**1**
v_5	0	**1**	**1**	0
v_6	0	0	0	**1**

C:	v_1	v_6	v_2	v_4	v_3	v_5
v_1	**1**	**1**	**1**	0	**1**	**1**
v_6	**1**	**1**	**1**	0	0	0
v_2	**1**	**1**	**1**	**1**	**1**	0
v_4	0	0	**1**	**1**	**1**	0
v_3	**1**	0	**1**	**1**	**1**	**1**
v_5	**1**	0	0	0	**1**	**1**

E:	w_1	w_2	w_3	w_4
u_1	**1**	**1**	0	0
u_2	**1**	**1**	**1**	**1**
u_3	0	0	**1**	**1**
u_4	0	**1**	**1**	0
u_5	**1**	**1**	**1**	0

Figure 2.3: Matrices defined in Definition 2.4. Matrix A is the adjacency matrix of the graph G_1, and Matrix B is the augmented adjacency matrix of G_1. Matrix C is obtained from B by permuting the rows and columns; the shapes enclosing its 1-entries illustrate the quasi Circ1P (see Definition 2.6), which will be used in Table 2.1. (Actually, the matrix C shows not only that B has the quasi Circ1P, but also that B has the Circ1P.) Matrix D is the maximal clique matrix of G_2, and Matrix E is the half adjacency matrix of the bipartite graph H. Matrix C shows that G_1 is a concave-round graph (see Definition 2.5) as well as a circular-arc graph (see Table 2.1), Matrix D shows that G_2 is an interval graph (see Table 2.1), and Matrix E shows that H is a convex bipartite graph (see Definition 2.5).

Definition 2.5. 1. A graph is *convex-round* if its adjacency matrix has the Circ1P, and it is *concave-round* if its augmented adjacency matrix has the Circ1P [BHY00].

2. A graph G is an *interval graph* if its vertices can be mapped to intervals on the real line such that two vertices are adjacent if and only if their corresponding intervals overlap [Ben59, Haj57]. If all intervals have the same length, then G is a *unit interval graph*; if no interval properly contains another interval, then G is a *proper interval graph*.

3. A graph G is a *circular-arc graph* if its vertices can be mapped to a set A of arcs on a circle such that two vertices are adjacent if and only if their corresponding arcs overlap. A circular-arc graph G is a *Helly circular-arc graph* if for every subset $A' \subseteq A$ it holds that $(\forall a_1, a_2 \in A' : a_1 \cap a_2 \neq \emptyset) \Rightarrow \bigcap_{a \in A'} a \neq \emptyset$.

4. A bipartite graph is *convex* if its half adjacency matrix has the C1P, it is *biconvex* if its half adjacency matrix has the C1P both for rows and

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

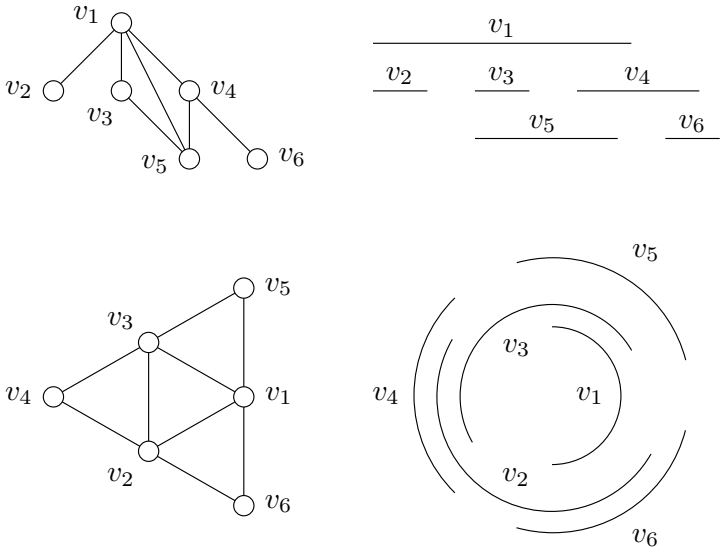


Figure 2.4: Top: An interval graph (which is not a proper or unit interval graph) and a set of intervals representing its vertices as described in Definition 2.5. Bottom: A circular-arc graph (which is not a Helly circular-arc graph) and a set of arcs representing its vertices as mentioned in Definition 2.5.

for columns, and it is *circular convex* if its half adjacency matrix has the Circ1P.

See Figures 2.3 and 2.4 for illustrations. Interval graphs and circular-arc graphs are known in graph theory for a long time; they are well-studied (alone for the recognition problem of these graphs there exists a number of results, see [BL76, COS98, FG65, HM99, HMPV00, Hsu92, KM89, KMMS06] for the recognition of interval graphs and [ES93, Hsu95, KN06, McC03, Tuc80] for the recognition of circular-arc graphs) and have applications in many fields [BLS99, Gol04]. One reason for the attention that these two graph classes attract is that many problems that are NP-complete on general graphs (for example, Independent Set) are polynomial-time solvable on interval graphs and circular-arc graphs and also on the other graph classes mentioned in Definition 2.5 (see [BLS99, Gol04, ISG08]). This important fact carries over to matrices with the C1P or the Circ1P, where many in general NP-hard matrix problems can be solved in polynomial time [CLRS01, ELR$^+$08, LBI$^+$01, NW88, VW62] (see also Section 5.3 and [MSW05, MW04, RS04]). We summarize the relationships between the graph classes of Definition 2.5 on the one hand and the C1P or Circ1P occurring in the matrices of Definition 2.4 on the other hand in Table 2.1. Note that, since proper interval graphs coincide with unit interval graphs [Rob69, Gar07], there is only one row for both classes. The property "quasi Circ1P" occurring in the table is defined as follows.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

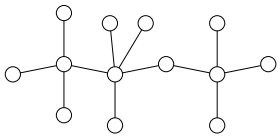


Figure 2.5: An example for a caterpillar.

Definition 2.6 ([Tuc71])**.** A symmetric matrix has the *quasi Circ1P* if (possibly after permuting the rows and columns without destroying the symmetry) for every 1-entry $m_{i,j}$ it holds that $m_{i,i} = m_{i,(i+1) \bmod n} = m_{i,(i+2) \bmod n} = \ldots = m_{i,(j-1) \bmod n} = m_{i,j} = 1$ or that $m_{j,j} = m_{(j+1) \bmod n,j} = m_{(j+2) \bmod n,j} = \ldots = m_{(i-1) \bmod n,j} = m_{i,j} = 1$.

The quasi Circ1P is illustrated in Figure 2.3. One can show that the Circ1P always implies the quasi Circ1P [Tuc71].

Table 2.1 contains only rather "prominent" graph classes; however, graphs whose vertex-edge incidence matrix or edge-vertex-edge incidence matrix has the C1P have also a very special structure.

Definition 2.7. A *caterpillar* is a tree in which every non-leaf vertex has at most two non-leaf neighbors.

See Figure 2.5 for an illustration.

The two characterizations given in the following theorem follow directly from the results of Tucker [Tuc72] described in the following paragraph and from the considerations of Hajiaghayi and Ganjali [HG02] and Tan and Zhang [TZ07].

Theorem 2.2. *1. A graph is a union of vertex-disjoint paths if and only if its edge-vertex incidence matrix has the C1P.*

2. A graph is a union of vertex-disjoint caterpillars if and only if its vertex-edge incidence matrix has the C1P.

See Figure 2.6 for an illustration.

Tucker's characterization of matrices having the C1P. Matrices with the C1P can be characterized by a set of forbidden submatrices: A matrix has the C1P iff it does not contain a matrix from this set as a submatrix. Such a characterization is very helpful when regarding matrix modification problems where one has to modify a matrix to achieve the C1P. This will be the subject of Chapter 4.

The characterization by Tucker [Tuc72] is based on a characterization of convex bipartite graphs in terms of so-called asteroidal triples defined as follows.

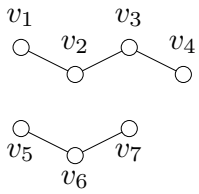
Definition 2.8. Let $G = (V, E)$ be a graph. Three vertices from V form an *asteroidal triple* if between any two of them there exists a path in G that does not contain any vertex from the closed neighborhood of the third vertex.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

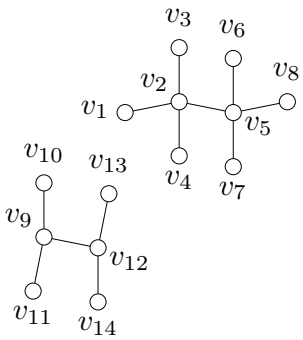
Table 2.1: Relationship between graph classes and matrix properties. The symbol "⇒" expresses that the membership in a graph class implies the matrix property for the matrix associated with the corresponding graph; the symbols "⇐" and "⇔" denote implications in the back direction and in both directions, respectively. The abbreviation "C1P r+c" stands for "C1P for rows *and* for columns." Of course, due to its symmetry an (augmented) adjacency matrix has the C1P or the Circ1P for rows iff it has the C1P or the Circ1P, respectively, for columns.

graph class	adjacency matrix	augmented adjacency matrix	half adjacency matrix	maximal clique matrix
convex-round	⇔ Circ1P (per def.)			
concave-round		⇔ Circ1P (per def.)		
⋂|				
circular-arc		⇔ quasi Circ1P [Tuc71] ⇐ Circ1P [Tuc71]		⇐ Circ1P
⋃|				
Helly circular-arc		⇒ quasi Circ1P ⇐ C1P		⇔ Circ1P [Gav74]
⋃|				
interval		⇒ quasi Circ1P ⇐ C1P		⇔ C1P [FG65]
⋃|				
proper / unit interval		⇔ C1P [Rob69]		⇔ C1P r+c [Fis85]
circular convex bipart.			⇔ Circ1P (per def.)	
⋃|				
convex bipart.			⇔ C1P (per def.)	
⋃|				
biconvex bipart.			⇔ C1P r+c (per def.)	

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.



	v_1	v_2	v_3	v_4	v_5	v_6	v_7
	1	**1**	0	0	0	0	0
	0	**1**	**1**	0	0	0	0
	0	0	**1**	**1**	0	0	0
	0	0	0	0	**1**	**1**	0
	0	0	0	0	0	**1**	**1**



	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}	v_{12}	v_{13}	v_{14}
v_1	**1**	0	0	0	0	0	0	0	0	0	0	0	0	0
v_2	**1**	**1**	**1**	**1**	0	0	0	0	0	0	0	0	0	0
v_3	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0
v_4	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0
v_5	0	0	0	**1**	**1**	**1**	**1**	0	0	0	0	0	0	0
v_6	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0
v_7	0	0	0	0	0	**1**	0	0	0	0	0	0	0	0
v_8	0	0	0	0	0	0	**1**	0	0	0	0	0	0	0
v_9	0	0	0	0	0	0	0	**1**	**1**	**1**	0	0	0	0
v_{10}	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0
v_{11}	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0
v_{12}	0	0	0	0	0	0	0	0	0	**1**	**1**	**1**	0	0
v_{13}	0	0	0	0	0	0	0	0	0	0	0	0	**1**	0
v_{14}	0	0	0	0	0	0	0	0	0	0	0	0	0	**1**

Figure 2.6: Top: A graph that is a union of vertex-disjoint paths, and its edge-vertex incidence matrix. Bottom: A graph that is a union of vertex-disjoint caterpillars, and its vertex-edge incidence matrix. Both matrices have the C1P.

For example, every cycle of length at least six contains several asteroidal triples. More examples for graphs containing asteroidal triples are shown in Figure 2.7.

In his characterization, Tucker does not use the term "convex bipartite graph"; however, convex bipartite graphs are identical to *"graphs with a V_2-consecutive[2] arrangement."* The following two theorems, hence, characterize convex bipartite graphs.

Theorem 2.3 ([Tuc72, Theorem 6])**.** *A bipartite graph $G = (V_1, V_2, E)$ has a V_2-consecutive[2] arrangement if and only if V_2 contains no asteroidal triple of G.*

Theorem 2.4 ([Tuc72, Theorem 7])**.** *In a bipartite graph $G = (V_1, V_2, E)$ the vertex set V_2 contains no asteroidal triple[2] if and only if G contains none of the graphs G_{I_k}, G_{II_k}, G_{III_k} (with $k \geq 1$), G_{IV}, and G_{V} as shown in Figure 2.7.*

[2]Tucker considers the C1P for columns, whereas we are interested in the C1P for rows. Hence, the roles of V_1 and V_2 are interchanged here compared to Tucker's publication [Tuc72].

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

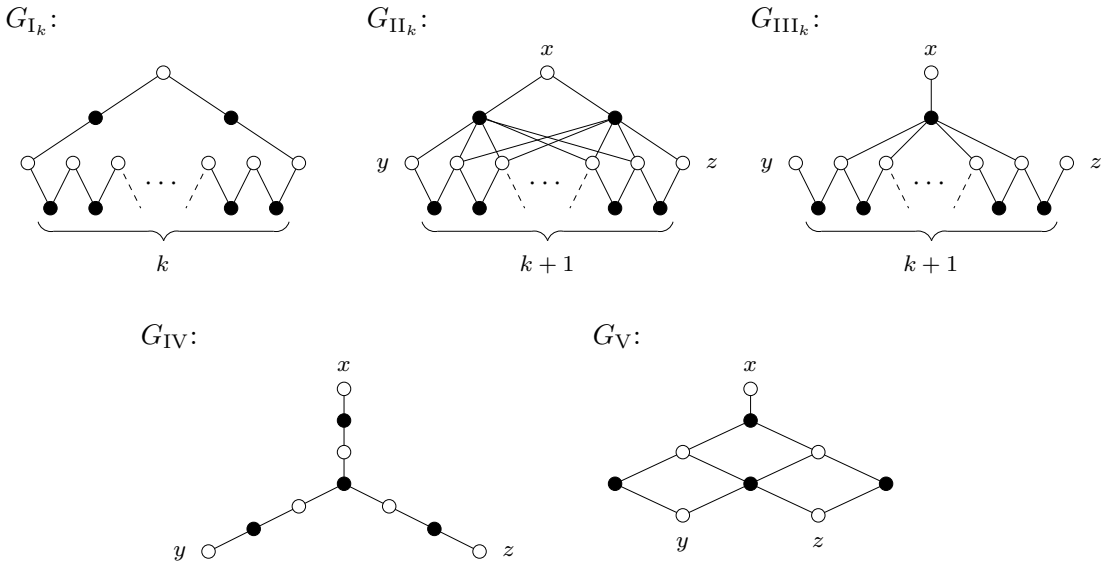


Figure 2.7: Forbidden induced subgraphs due to Tucker [Tuc72]: The vertex set V_2 of a bipartite graph $G = (V_1, V_2, E)$ contains an asteroidal triple iff G contains one of the displayed graphs as an induced subgraph, where white vertices correspond to vertices in V_2. The numbers k and $k+1$ refer to the number of black vertices in the lower parts of the first three graphs. In the case of the graph $G_{\mathrm{I}_k} \in T$, every triple of white vertices is an asteroidal triple. In all other cases, there is exactly one asteroidal triple consisting of white vertices; this triple is denoted by x, y, z.

A characterization that is very similar to the one given in Theorem 2.3 is also known for interval graphs: A graph is an interval graph iff it is chordal and contains no asteroidal triple [LB62].

The following theorem, which finally characterizes matrices with the C1P, is a direct consequence of Theorems 2.3 and 2.4.

Theorem 2.5 ([Tuc72, Theorem 9])**.** *A matrix M has the C1P if and only if it contains none of the matrices M_{I_k}, M_{II_k}, M_{III_k} (with $k \geq 1$), M_{IV}, and M_{V} as shown in Figure 2.8 as a submatrix.*[3]

The matrices in Figure 2.8 are obtained from the graphs given in Figure 2.7 by considering the half adjacency matrices of these graphs, that is, the graphs given in Figure 2.7 are the representing graphs of the matrices given in Figure 2.8.

We will use the symbol T throughout the whole thesis to denote the set of matrices M_{I_k}, M_{II_k}, M_{III_k} (with $k \geq 1$), M_{IV}, and M_{V} given in Theorem 2.5. The set T is called a set of *forbidden submatrices* for the C1P.

[3]The roles of rows and columns are interchanged here compared to Tucker's publication [Tuc72].

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

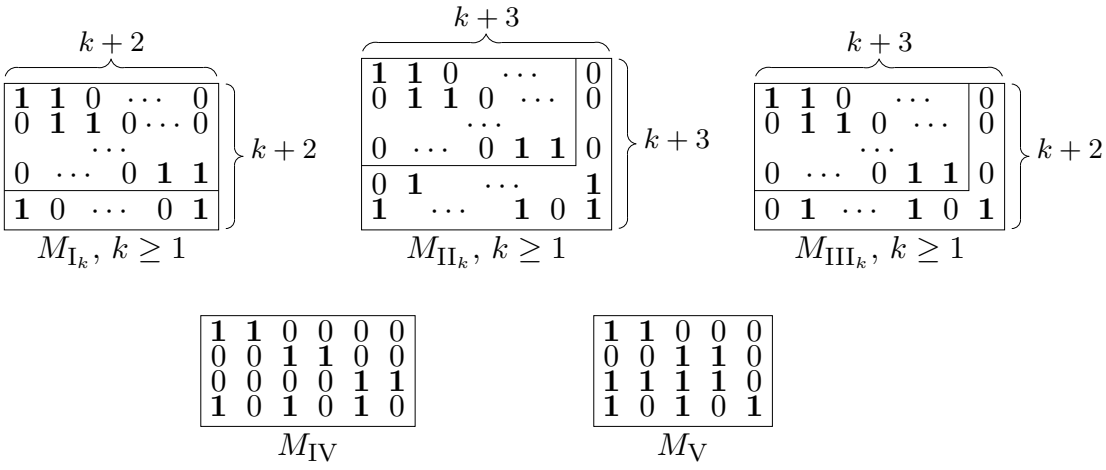


Figure 2.8: The forbidden submatrices for the C1P due to Tucker [Tuc72], given in Theorem 2.5.

2.3 Recognizing the C1P

Here we demonstrate some of the most prominent algorithms known for recognizing matrices with the C1P. Hand in hand with these algorithms, several characterizations for matrices having the C1P have been developed; we will also present some of these characterizations. Moreover, we fix a small error in Theorem 6.1 of [McC04]. As we will see (Theorem 2.7), every algorithm that recognizes interval graphs can also be used to recognize matrices with the C1P. However, here we mention only those results that explicitly deal with matrices and the C1P, because first transforming a matrix into a graph and then testing whether this graph is an interval graph does not automatically yield an efficient (that is, linear-time) algorithm for recognizing the C1P. Note that some of the presented results concern the C1P for rows and some the C1P for columns; clearly, due to the symmetry of these two properties, all algorithms can be used for recognizing both of these properties.

The following definition introduces some terms needed in this section.

Definition 2.9. Two columns c_1, c_2 of a 0/1-matrix *overlap* if there exist three rows r_1, r_2, r_3 such that row r_1 contains a 1 in both columns c_1 and c_2, row r_2 contains a 1 in c_1 but not in c_2, and row r_3 contains a 1 in c_2 but not in c_1.

A column c_1 *contains* a column c_2 if for every row r it holds that if c_2 has a 1 in row r then c_1 also has a 1 in row r. If a column is not contained in any other row, it is *maximal*.[4]

All terms can be defined analogously for rows.

[4]Note that this definition of a maximal column does not allow the existence of two identical maximal columns.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Using overlapping columns. The first polynomial-time algorithm to recognize matrices having the C1P (for columns) was presented by Fulkerson and Gross [FG65]. Their idea is to decompose the input matrix M into disjoint column sets in such a way that the whole matrix has the C1P for columns iff each matrix induced by one of the column sets has the C1P for columns. The partitioning of M's columns into different column sets is performed by defining an *overlap graph* $\mathcal{G}(M)$: Every vertex of this graph corresponds to a column of M, and two vertices are connected iff their corresponding columns overlap. Every connected component of this graph defines one column set of the partition of M needed by the algorithm. Now, for the rows of every submatrix of M that is induced by one of the column sets of the partition, a C1-ordering can easily be found, if existing, by considering one column after the other in a certain order and re-arranging the rows if necessary. If, however, the considered submatrix does not have the C1P for columns, this can easily be detected because in this case one always encounters a column whose 1-entries cannot be placed consecutively without destroying the C1P for columns in the already considered columns. In the last phase of the algorithm, the row orderings computed for the submatrices are combined, while possibly re-arranging some of the rows, by using a directed graph called the *component graph* $\mathcal{D}(M)$, which defines a partial order on the connected components of $\mathcal{G}(M)$. The whole matrix M has the C1P for columns iff all of the submatrices have the C1P for columns [FG65, Theorem 4.1]. The whole procedure takes polynomial time. A more recent linear-time algorithm by Hsu [Hsu02] for recognizing the C1P is based on very similar ideas.

PQ-Trees. Booth and Lueker were the first to present a linear-time algorithm for recognizing matrices with the C1P for columns [BL76]. Linear time means a running time that is linear in the number of rows plus the number of columns plus the number of 1-entries of the given matrix. Booth and Lueker introduced so-called *PQ-Trees*, which are not only useful for recognizing matrices with the C1P, but also, for example, for recognizing matrices with the Circ1P and for recognizing interval graphs and planar graphs. In the context of recognizing matrices with the C1P for columns, a PQ-tree is an ordered rooted tree that represents a C1-ordering for the rows of a matrix M. To this end, the inner nodes of the PQ-tree are labeled as P-nodes and Q-nodes, and the leaves one-to-one correspond to the rows of the underlying matrix M. Ordering the rows of M in the same way as their corresponding leaves in the PQ-tree yields the strong C1P for columns. In addition, any PQ-tree for M implicitly represents *all* possible C1-orderings for M's rows, because by applying a series of certain node reordering operations, every possible PQ-tree for the set of M's rows can be transformed into any other possible PQ-tree for this row set. Therefore, PQ-trees have the following properties.

1. If T is a PQ-tree for a matrix M, then the sequence of T's leaves from left to right describes a C1-ordering for M's rows.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

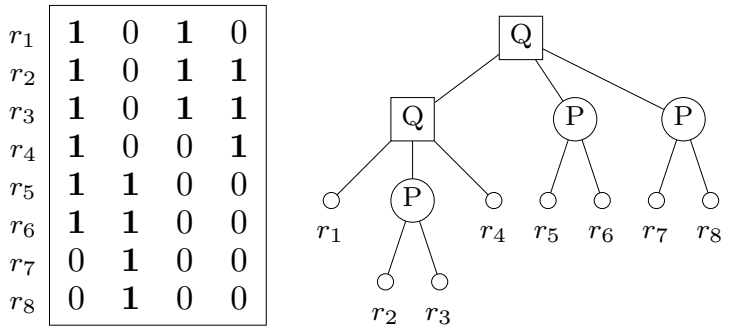


Figure 2.9: A matrix M and the PQ-tree representing the row ordering of M.

2. Each C1-ordering for M's rows one-to-one corresponds to a PQ-tree.

3. The set of PQ-trees for the rows of a matrix is closed under the following two operations:

   - Arbitrarily reordering the children of a P-node.
   - Putting the children of a Q-node in reverse order.

   In particular, none of these two operations destroys property 1.

See Figure 2.9 for an illustration.

In order to either construct a PQ-tree for a given matrix M or decide that M does not have the C1P for columns, the algorithm of Booth and Lueker starts with a tree (the so-called *universal PQ-tree*) consisting of one P-node forming the root and one leaf node for every row of M. The algorithm considers the columns of M one after the other, and in every step it either reports that M does not have the C1P for columns, or it modifies the tree, by using a complicated case distinction, in such a way that the resulting tree is a PQ-tree for the matrix that is induced by the columns considered so far.

Variations of PQ-Trees. Several variations of PQ-trees have been proposed since their first appearance. Korte and Möhring [KM89] introduced *MPQ-trees* ("modified PQ-trees"), where the inner nodes contain some additional information, which results in a simpler construction of these trees. Meidanis et al. [MPT98] defined *PQR-trees*, which are a generalization of PQ-trees in the following sense: For every matrix M that has the C1P for columns, the set of PQR-trees for M's rows is identical with the set of PQ-trees for M's rows. However, in contrast to PQ-trees, PQR-trees are also defined for matrices that do not have the C1P for columns; in this case they contain, in addition to P-nodes and Q-nodes, inner nodes labeled as R-nodes, which can be useful for identifying why the matrix does not have the C1P. A similar approach was used by McConnell [McC04]. He introduced *generalized PQ-trees* in order to determine

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

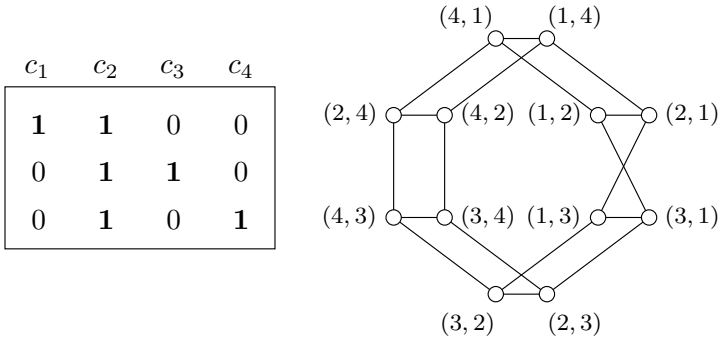


Figure 2.10: The matrix M_{III_1} and its incompability graph [McC04]. The graph contains several odd cycles of length 7.

if a matrix has the C1P for rows and, if it does not, to generate a "certificate" therefor: Such a certificate is a small (compared to the size of the input matrix) proof that can be verified by a "fast and uncomplicated" polynomial-time algorithm (for more details about such certificates see [KMMS06]). The certificate produced by the algorithm of McConnell [McC04] for an $m \times n$ input matrix M consists of an odd cycle of length at most $n + 3$ in the so-called *incompatibility graph* of M. This graph is defined as the graph $G = (V, E)$ with

$$\begin{aligned} V =& \{(j_1, j_2) \mid 1 \leq j_1 \leq n \ \wedge \ 1 \leq j_2 \leq n \ \wedge \ j_1 \neq j_2\} \\ E =& \{\{(j_1, j_2), (j_2, j_1)\} \mid 1 \leq j_1 < j_2 \leq n\} \cup \\ & \{\{(j_1, j_2), (j_2, j_3)\} \mid j_1 \neq j_3 \wedge \\ & \exists i \in \{1, \ldots, m\} : (m_{i,j_1} = m_{i,j_3} = 1 \ \wedge \ m_{i,j_2} = 0)\}. \end{aligned}$$

Intuitively speaking, the incompability graph contains two vertices for every pair of columns of M: one vertex for every possible relative ordering of the two columns. If two vertices (j_1, j_2) and (j_3, j_4) in the incompatibility graph are connected by an edge, then the corresponding two orderings conflict in the sense that there is no C1-ordering for M's columns that places the column j_1 to the left of j_2 *and* the column j_3 to the left of j_4. See Figure 2.10 for an example. The connection between the incompability graph of a matrix and the C1P is specified in the following theorem. The original formulation of this theorem is due to McConnell [McC04] and contains a minor error concerning the cycle length. We will describe this error as well as a sketch how to fix it at the end of this section.

Theorem 2.6 ([McC04, Theorem 6.1])**.** *An $m \times n$ matrix M has the C1P iff its incompatibility graph is bipartite. If M does not have the C1P, then its incompatibility graph has an odd cycle of length at most $n + 3$.*

See Figure 2.10. In order to use an odd cycle as mentioned in Theorem 2.6 as a certificate for the absence of the C1P, one encodes the cycle as a list of at most $n + 3$ pairs (e_i, d_i), where e_i is the ith edge in the cycle and d_i, which

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

can either be null or an integer, describes the "reason" for the existence of e_i: if e_i is an edge of the kind $\{(j_1, j_2), (j_2, j_1)\}$, then d_i is null; if e_i is an edge of the kind $\{(j_1, j_2), (j_2, j_3)\}$ with $j_1 \neq j_3$, then d_i contains the index of a row that contains a 1 in the columns j_1 and j_3 and a 0 in column j_2. Clearly, the correctness of this certificate can be checked in $O(n)$ time.

PC-Trees. A remarkable simplification for building PQ-trees was exhibited by Hsu and McConnell [HM03], who introduced *PC-trees*. These trees can be seen as unrooted PQ-trees that represent Circ1-circular orderings for the columns or rows of a matrix instead of C1-orderings. Instead of Q-nodes, PC-trees contain C-nodes; the order of the leaves of a PC-tree describes a Circ1-circular ordering for the columns of the underlying matrix. PC-trees have the following properties.

1. If T is a PC-tree for a matrix M, then any sequence obtained by considering T's leaves in clockwise or counter-clockwise order describes a Circ1-circular ordering for M's columns.

2. Each Circ1-circular ordering for M's columns one-to-one corresponds to a PC-tree.

3. The set of PC-trees for the columns of a matrix is closed under the following two operations:

   - Arbitrarily reordering the neighbors of a P-node.
   - First rooting T at a neighbor of a C-node v, then "flipping" the subtree whose root is v, and finally un-rooting the tree. Herein, "flipping" a subtree means putting the children of every node of the subtree in reverse order.

   In particular, none of these two operations destroys property 1.

See Figure 2.11 for an illustration.

Like in the case of PQ-trees, there is a linear-time algorithm that either constructs a PC-tree for a given matrix M or decides that M does not have the Circ1P [HM03]. Similarly to the algorithm of Booth and Lueker [BL76], this algorithm starts with a tree consisting of one P-node which has one leaf neighbor for every column of M. The algorithm considers the rows of M one after the other, and in every step it either reports that M does not have the Circ1P, or it modifies the tree. These modifications, however, are much simpler than those proposed by Booth and Lueker for updating a PQ-tree.

Due to Corollary 2.2, the PC-tree algorithm can not only be used to decide whether a matrix has the Circ1P, but also to decide whether it has the C1P: Just add a column that contains only 0s to the given matrix. Rooting the PC-tree for the resulting matrix at the neighbor of the leaf node that corresponds to the newly inserted column and then deleting this leaf node yields a PQ-tree for the original matrix.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	c_{10}
1	**1**	0	0	0	0	0	0	0	0
0	**1**	**1**	0	0	0	0	0	0	0
1	0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
0	0	0	**1**	**1**	0	0	0	0	0
0	0	0	0	**1**	**1**	0	0	0	0
0	0	0	0	0	0	**1**	**1**	0	0
0	0	0	0	0	0	0	**1**	**1**	0
0	0	0	0	0	0	0	0	0	**1**

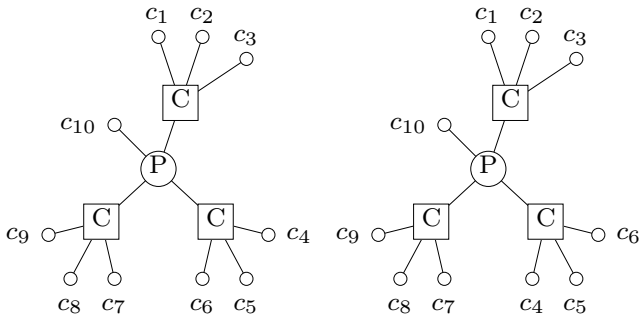


Figure 2.11: PC-trees for a matrix. Left: a matrix M. Middle: the PC-tree representing the circular column ordering of M. Right: The PC-tree obtained from the PC-tree in the middle by "flipping" the subtree rooted at the bottom-right C-node.

Further recognition algorithms. A simple linear-time algorithm without using any variant of PQ-trees was presented by Habib et al. [HMPV00]: They use a so-called "Lex-BFS ordering" of the vertices of a graph to decide in linear time whether the graph is an interval graph. Habib et al. also prove Theorem 2.7 below, which implies that any algorithm recognizing interval graphs can also be used for recognizing matrices with the C1P. Habib et al. show that the recognition of matrices with the C1P in this way is possible in linear time.

For describing how an algorithm recognizing interval graphs can be used to decide whether a given matrix M has the C1P, let $G_{\mathrm{ra}}(M)$ denote the graph that has one vertex for every row of M and where two vertices are adjacent iff their corresponding rows in M have a 1 in a common column; we call this graph the *row adjacency graph* of M. Note that several matrices can have identical row adjacency graphs; moreover, if a matrix M is the maximal clique matrix of a graph G, then G is the row adjacency graph of M. The latter observation leads to the following finding.

Theorem 2.7 ([HMPV00, Theorem 2])**.** *For a* 0/1*-matrix* M *the following statements are equivalent:*

1. *The row adjacency graph* $G_{ra}(M)$ *is an interval graph and* M *is its maximal clique matrix.*

2. *The columns of* M *are maximal and* M *has the C1P for rows.*

By appending a size-$n \times n$ unit matrix I to a matrix M, a matrix

$$\tilde{M} = \begin{pmatrix} M \\ I \end{pmatrix}$$

can be constructed, which obviously has the following properties.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

1. $\tilde{M}$ has the C1P iff M has the C1P,
2. every column of $\tilde{M}$ is maximal, and
3. $G_{\mathrm{ra}}(\tilde{M})$ is an interval graph and $\tilde{M}$ is its maximal clique matrix iff M has the C1P for rows (this follows from Theorem 2.7).

Testing whether M has the C1P now obviously reduces to checking whether $G_{\mathrm{ra}}(\tilde{M})$ is an interval graph and $\tilde{M}$ is its maximal clique matrix. However, note that using Theorem 2.7 in combination with a linear-time algorithm for recognizing interval graphs does not automatically yield a *linear-time* algorithm for recognizing matrices with the C1P. Nevertheless, Habib et al. give an algorithm using Theorem 2.7 for recognizing matrices with the C1P in linear time.

Finally, we like to point out that recognizing matrices that have "almost the C1P" is much more difficult than recognizing matrices with the C1P: While matrices with the C1P can be recognized in linear time, the problem of deciding whether the columns of a matrix M (not having the C1P) can be permuted in such a way that the overall number of blocks of 1s is at most k is NP-complete [GJ79, Had02] (see also [HL08]); deciding whether M's columns can be permuted such that the number of blocks of 1s in each row is at most c is NP-complete for every constant $c \geq 2$ [AM96, FGS96, GGKS95, WLZ07] (see also [BGRS04, WR00]). However, there are algorithms for recognizing matrices that are "close" to having the C1P in some other sense [Hsu97, LH03].

An Error in Theorem 6.1 of McConnell [McC04]

The original version [McC04, Theorem 6.1] of Theorem 2.6 contains an error concerning the cycle length. In this subsection, we describe the error and prove the correct cycle length.

The original theorem is formulated in terms of subset families instead of matrices. From this point of view, a matrix is represented by a *domain* V, which is a set and contains one element for every column of M, and a set family $\mathcal{F}$, which represents the rows of M and contains one subset of V for every row of M: a row i of M is represented by a set in $\mathcal{F}$ that contains those elements of V that correspond to columns of M with a 1 in row i, see Figure 2.12. Clearly, all terms defined for matrices, like the C1P or incompability graphs, can be directly applied to such set families. In particular, a *consecutive ordering* for V orders the elements of V in such a way that the members of every set $Y \in \mathcal{F}$ appear consecutively in this ordering. The original version of Theorem 2.6 reads as follows.

[McC04], Theorem 6.1. *Let $\mathcal{F}$ be an arbitrary set family on domain V. Then $\mathcal{F}$ has the consecutive-ones property if and only if its incompatibility graph is bipartite, and if it does not have the consecutive-ones property, the incompatibility graph has an odd cycle of length at most $n + 2$ (where $n = |V|$).*

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

c_1	c_2	c_3	c_4
1	0	**1**	0
0	**1**	0	**1**
1	0	**1**	**1**

$V = \{c_1, c_2, c_3, c_4\}$
$\mathcal{F} = \{\{c_1, c_3\}, \{c_2, c_4\}, \{c_1, c_3, c_4\}\}$

Figure 2.12: A matrix can be represented by a set family $\mathcal{F}$ on a ground set ("domain") V whose elements correspond to the columns of the matrix. The shown set family has the C1P, because there is an ordering (for example c_3, c_1, c_4, c_2) for V's elements such that the members of every set from $\mathcal{F}$ appear consecutively.

A counterexample for this theorem is shown in Figure 2.10: The matrix M_{III_1} does not have the C1P; however, the shortest odd cycle in its incompability graph does not have a length of $n+2$ as claimed by the theorem, but a length of $n+3$. Indeed, replacing "$n+2$" by "$n+3$" in Theorem 6.1 of [McC04] makes the statement correct.

We will shortly explain the reason for the error and then sketch how to prove the correct Theorem 2.6 along the lines of McConnell's proof for Theorem 6.1 of [McC04].

The error in the proof for Theorem 6.1 of [McC04]. The proof of McConnell for Theorem 6.1 uses the following approach. By using a generalized PQ-tree, the set family $\mathcal{F}$ is partitioned into sub-families in such a way that if $\mathcal{F}$ does not have the C1P, then there is at least one such sub-family $\mathcal{Q}$ that also does not have the C1P. Now, an algorithm is proposed that considers the sets in $\mathcal{Q}$ one after the other in a specific order and updates in each step some data structures, which finally allow to prove an odd cycle of the claimed length in the incompability graph.

During the execution of the algorithm, let $\mathcal{F}' \subseteq \mathcal{Q}$ denote all sets considered so far. In every step, the algorithm updates an ordered list $\mathcal{P}$ of pairwise disjoint subsets $X_1, \ldots, X_k$ of V such that $X_1 \cup \ldots \cup X_k = \bigcup \mathcal{F}'$. Herein, the following invariants are maintained: First, every set X_i in $\mathcal{P}$ forms a *block* for $\mathcal{F}'$, which means that X_i is a maximal subset of $\bigcup \mathcal{F}'$ with the property that

$$\forall a, b \in X_i \; \forall Y \in \mathcal{F}' : (a \in Y \Leftrightarrow b \in Y).$$

See Figure 2.13 for an illustration of the blocks for a set family. Second, the ordering of the sets X_i in $\mathcal{P}$ is consistent with a consecutive ordering for $\bigcup \mathcal{F}'$. This means that when the elements of $\bigcup \mathcal{F}'$ are ordered in such a way that for every $1 \leq i < j \leq k$ the elements of X_i appear before the elements of X_j, one obtains a consecutive ordering for $\bigcup \mathcal{F}'$ (see Figure 2.13). The order in which the algorithm considers the sets of $\mathcal{Q}$ guarantees that $X_1, \ldots, X_k$ and $X_k, \ldots, X_1$

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

$$
\begin{array}{lccccccccccc}
Y_1 = & \{ & c_1 & c_2 & c_3 & c_4 & c_5 & \} & & & & \\
Y_2 = & & & & & \{ & c_4 & c_5 & c_6 & c_7 & c_8 & c_9 & c_{10} & \} \\
Y_3 = & & & & & & & & & & \{ & c_8 & c_9 & c_{10} & c_{11} & c_{12} & \} \\
 & \underbrace{\qquad} & & & & \underbrace{\qquad} & & & \underbrace{\qquad} & & \underbrace{\qquad} & & & & \underbrace{\qquad} & & \\
 & X_1 & & & & X_2 & & & X_3 & & X_4 & & & & X_5 & &
\end{array}
$$

Figure 2.13: A sequence $\mathcal{P} = (X_1, \ldots, X_5)$ of five blocks for a set family $\mathcal{F}' = \{Y_1, Y_2, Y_3\}$ with $\bigcup \mathcal{F}' = \{c_1, \ldots, c_{12}\}$. Such a sequence is computed by the algorithm used in the proof for Theorem 6.1 of [McC04]. The shown order $c_1, \ldots, c_{12}$ of the elements is a consecutive ordering for $\bigcup \mathcal{F}'$. The sequence $\mathcal{P}$ and its reverse are the only orderings of the blocks that are consistent with a consecutive ordering for $\bigcup \mathcal{F}'$.

are the only two block orderings that fulfill this invariant—this is crucial for the argumentation used in the proof.

If $\mathcal{Q}$ does not have the C1P, then the algorithm must clearly perform a step in which a set $Z \in \mathcal{Q}$ is selected such that the set family $\mathcal{F}'$, which contains the sets considered in the previous steps, has the C1P, but $\mathcal{F}' \cup \{Z\}$ does not. This implies that it is impossible to find a sequence $\mathcal{P}$ of blocks as mentioned above for $\mathcal{F}' \cup \{Z\}$. Following the argumentation in [McC04], this can only be the case if there are three elements a, b, c in $\bigcup \mathcal{F}'$ and three indices $1 \leq p < q < r \leq k$ such that $a \in X_p, b \in X_q, c \in X_r$, and $a \in Z, b \notin Z, c \in Z$. As a consequence, an odd cycle of the claimed length $n + 2$ in the incompability graph can be found.

However, there can also be another situation, which has not been considered in [McC04]: It is as well impossible to find a sequence $\mathcal{P}$ of blocks for $\mathcal{F}' \cup \{Z\}$ if there are three elements a, b, c in $\bigcup \mathcal{F}'$, an index $1 < p < k$ and an element $z \in V \setminus \bigcup \mathcal{F}'$ such that $a \in X_1, b \in X_p, c \in X_k$, and $a \notin Z, b \in Z, c \notin Z, z \in Z$. In this case, an odd cycle of length $n + 2$ in the incompability graph cannot be guaranteed.

We mention in passing that there is also a fault in Corollary 6.1 of [McC04]: The path length claimed in the corollary is $k - 1$, whereas the correct length is k. The proof of Corollary 6.1, however, is correct (that is, it proves the correct length k of the path).

Proof for Theorem 2.6. To prove Theorem 2.6, it suffices to show the following lemma because the case that is not covered by this lemma is proven correctly in [McC04].

Lemma 2.1. *Assume that $\mathcal{F}' \cup \{Z\}$ does not have the C1P and that there are three elements a, b, c in $\bigcup \mathcal{F}'$, an index $1 < p < k$ and one element $z \in V \setminus \bigcup \mathcal{F}'$ such that $a \in X_1, b \in X_p, c \in X_k$, and $a \notin Z, b \in Z, c \notin Z, z \in Z$. Then there is an odd cycle of length at most $n + 3$ in the incompability graph of $\mathcal{F}' \cup \{Z\}$.*

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Proof. Due to Theorem 2.5, it suffices to prove that the lemma is true if $\mathcal{F}' \cup \{Z\}$ represents one of the matrices M_{I_k}, M_{II_k}, M_{III_k} (with $k \geq 1$), M_{IV}, and M_{V} (see Figure 2.8).

For the matrices M_{I_k} and M_{II_k} with $k \geq 1$ we do not have to prove anything, because, given the situation described in the lemma, the set family $\mathcal{F}' \cup \{Z\}$ cannot represent one of these matrices. The reason is that every column of M_{I_k} and M_{II_k} contains at least two 1s. Hence, when the set Z is considered by the algorithm, the set $V \setminus \bigcup \mathcal{F}'$ is already empty, and the element z cannot exist. (To see this, note that $\mathcal{F}'$ represents all but one rows of the matrix and $V \setminus \bigcup \mathcal{F}'$ represents those columns that contain only 0s in the rows corresponding to $\mathcal{F}'$.)

In the case of M_{IV} or M_{V} one can easily verify that the incompability graph of these matrices contains an odd cycle of length at most $n + 3$: The incompability graph of M_{IV} contains a C_9, the incompability graph of M_{V} contains a C_5. It remains to show that the incompability graph of the matrix M_{III_k}, $k \geq 1$, contains an odd cycle of length at most $n + 3$. We distinguish the two cases where n is even and n is odd.

If n is even, then one can easily verify that there is a path

$$(n, n-1), (n-2, n), (n, n-3), \ldots, (2, n), (n, 1)$$

of length $n-2$ from $(n, n-1)$ to $(n, 1)$ in the incompability graph because of the rows $m-1, m-2, \ldots, 1$. In addition, the incompability graph contains

- the edge $\{(n, 1), (1, n-2)\}$ because of row m,
- the edge $\{(1, n-2), (n-1, 1)\}$ because of row $m-1$,
- the edge $\{(n-1, 1), (2, n-1)\}$ because of row 1,
- the edge $\{(2, n-1), (n-1, n)\}$ because of row m, and
- the edge $\{(n-1, n), (n, n-1)\}$.

The path from $(n, n-1)$ to $(n, 1)$ together with these edges forms an odd cycle of length $n + 3$.

If n is odd, then there is a path

$$(n, n-1), (n-2, n), (n, n-3), \ldots, (n, 2), (1, n)$$

of length $n-2$ from $(n, n-1)$ to $(1, n)$ in the incompability graph because of the rows $m-1, m-2, \ldots, 1$. In addition, the incompability graph contains

- the edge $\{(1, n), (n-2, 1)\}$ because of row m,
- the edge $\{(n-2, 1), (1, n-1)\}$ because of row $m-1$,
- the edge $\{(1, n-1), (n-1, 2)\}$ because of row 1, and
- the edge $\{(n-1, 2), (n, n-1)\}$ because of row m,

such that there is an odd cycle of length $n + 2$. □

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

2.4 Integer Linear Programming on Coefficient Matrices having the C1P/Circ1P

In this section, we consider the connection between matrices with the C1P or Circ1P and the hardness of solving integer linear programs (ILPs). Integer linear programs (see [Sch86]) are inequation/function systems that can be used to model a huge number of decision and optimization problems (see [KW97, Sie96]) and for which many solving algorithms and implementations exist (for example, GLPK (see [GLP08]) or CPLEX®).

While solving ILPs in general is NP-hard (see Section 2.4.1), we are here interested in the solvability of ILPs in the special case where the matrix consisting of the coefficients of the inequations has the C1P—such ILPs occur, for example, in biological applications [ACE+08] and in Section 5.3.1—or the Circ1P. We first consider some classes of polynomial-time solvable ILPs and show that ILPs whose coefficient matrices have the C1P belong to these classes. Then we describe some more specific algorithms for ILPs with coefficient matrices having the C1P or the Circ1P.

The algorithms that we consider do not only work for ILPs with 0/1-coefficient matrices, but also for ILPs whose coefficient matrices consist of entries from $\{0, 1, -1\}$. Therefore, we extend the definition of the C1P, which was defined only for 0/1-matrices so far, to $0/\pm 1$-matrices as follows: A $0/\pm 1$-matrix has the C1P if every row contains only entries either from $\{0, 1\}$ or from $\{0, -1\}$ and the columns can be permuted such that in every row the non-zero entries appear consecutively. Analogously, a $0/\pm 1$-matrix has the Circ1P if every row contains only entries either from $\{0, 1\}$ or from $\{0, -1\}$ and the columns can be permuted such that in every row the non-zero entries appear consecutively when the matrix is wrapped around a vertical cylinder.

We will only give a very basic introduction to (integer) linear programming and refer to Schrijver [Sch86] for more details wherever no other references are given (see also [MG06, PS98] for introductions).

2.4.1 (Integer) Linear Programming Basics

A *linear program (LP)* is an instance of the following optimization problem.

Linear Programming
Input: A set $x_1, \ldots, x_n$ of variables, a set of m linear inequations

$$a_{i,1}x_1 + \ldots + a_{i,n}x_n \leq b_i,\ 1 \leq i \leq m,$$

and a linear *objective function*

$$c_1x_1 + \ldots + c_nx_n$$

with $a_{i,j}, b_j, c_j \in \mathbb{Q}$ for $1 \leq i \leq m, 1 \leq j \leq n$.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Task: Assign values to $x_1, \ldots, x_n$ such that all given inequations are satisfied and that the value of the objective function is maximized.[5]

An assignment of values to the variables that satisfies the given inequations is called a *(feasible) solution* for the LP. Equivalently, the problem can formulated in a more compact form as follows.

Input: An $m \times n$ matrix $A = (a_{i,j})$, an n-entry column vector $\vec{b}$ and an m-entry row vector $\vec{c}^{\,\mathrm{T}}$ with all entries in $A, \vec{b}, \vec{c}^{\,\mathrm{T}}$ from $\mathbb{Q}$.
Task: Find an m-entry column vector $\vec{x}$ that satisfies $A\vec{x} \leq \vec{b}$ and maximizes $\vec{c}^{\,\mathrm{T}}\vec{x}$.

A vector containing only integers is called *integral*. The variant of Linear Programming where only integral solutions are allowed is called Integer Linear Programming; its instances are called *integer linear programs (ILPs)*. There are also decision problems corresponding to the optimization problems Linear Programming and Integer Linear Programming. The instances of these decision problems contain no objective function (no vector c); the task is to decide if there is any feasible solution.

Linear Programming can be solved in polynomial time. In particular, there is an algorithm for solving LPs that needs $O((n^3/\ln n)L)$ arithmetic operations [Ans99] (see also [PW00] for an overview over efficient algorithms for solving LPs), where L is the total bit number of the input. In contrast, it is easy to see that Integer Linear Programming is NP-hard. For example, Vertex Cover III (see Section 1.3.1) can easily be reduced to Integer Linear Programming by transforming an instance $G = (\{v_1, \ldots, v_n\}, E)$ of the latter problem into an ILP

$$\begin{array}{lll} \text{Maximize } n - \sum_{j=1}^{n} x_j & & \\ \text{subject to} & & \\ -x_{j_1} - x_{j_2} \;\leq\; -1, & & \forall \{v_{j_1}, v_{j_2}\} \in E \\ x_j \;\in\; \{0,1\} & & \forall j \in \{1, \ldots, n\}. \end{array}$$

Actually, the decision version of Integer Linear Programming is NP-complete [BT76, GS78, KM78, HU79].

An LP (ILP) is called *feasible* if it admits a feasible solution, and *unfeasible* otherwise. Even if an LP (ILP) is feasible, it is possible that there is no optimal solution: the optimum value of the objective function may be unbounded, meaning that for every feasible solution there exists another feasible solution yielding

[5] This is the standard form for LPs. To express an inequation of the form $a_{i,1}x_1 + \ldots + a_{i,n}x_n \geq b_i$, multiply it with -1; to express an equation $a_{i,1}x_1 + \ldots + a_{i,n}x_n = b_i$, replace it by the two inequations $a_{i,1}x_1 + \ldots + a_{i,n}x_n \leq b_i$ and $-a_{i,1}x_1 - \ldots - a_{i,n}x_n \leq -b_i$. Analogously, a minimization problem can be turned into a maximization problem by multiplying the objective function with -1.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

a higher value of the objective function. Given an LP on n variables, one can interpret its solution space as an n-dimensional Eukledian space; every inequation of the LP defines a half-space that contains all value-to-variable assignments satisfying this inequation. The intersection of all the half-spaces defined by the inequations of an LP is called the *polyhedron* defined by the LP. If the polyhedron defined by an LP is *integral*, which means that each of its corners corresponds to an integral solution, then the ILP defined by the inequations and the objective function of the LP can be solved in polynomial time by solving the LP. The reason is that for any feasible LP with bounded optimum value there is an optimal solution that corresponds to a corner of the polyhedron defined by the LP, and this corner can be found in polynomial time.

Given an LP

$$\begin{array}{l} \text{Maximize } \vec{c}^{\,\mathrm{T}}\vec{x} \\ \text{subject to} \\ A\vec{x} \quad \leq \quad \vec{b}, \end{array}$$

then the problem

$$\begin{array}{l} \text{Minimize } \vec{z}^{\,\mathrm{T}}\vec{b} \\ \text{subject to} \\ \vec{z}^{\,\mathrm{T}}A \quad = \quad \vec{c}^{\,\mathrm{T}} \\ \vec{z}^{\,\mathrm{T}} \quad \geq \quad \vec{0}^{\,\mathrm{T}} \end{array}$$

is called the *dual (problem)* for the given LP[6]—here the task is to find an optimal row vector $\vec{z}^{\,\mathrm{T}}$. The dual problem is feasible and its optimum value is bounded iff the original LP is feasible and has a bounded optimum value; if this is the case then it holds that

$$\max\{\vec{c}^{\,\mathrm{T}}\vec{x} \mid A\vec{x} \leq \vec{b}\} = \min\{\vec{z}^{\,\mathrm{T}}\vec{b} \mid \vec{z}^{\,\mathrm{T}}A = \vec{c}^{\,\mathrm{T}} \ \wedge \ \vec{z}^{\,\mathrm{T}} \geq \vec{0}^{\,\mathrm{T}}\}.$$

For ILPs, there is no such equation, and it only holds that $\max\{\vec{c}^{\,\mathrm{T}}\vec{x} \mid A\vec{x} \leq \vec{b} \wedge \vec{x} \text{ is integral}\} \leq \min\{\vec{z}^{\,\mathrm{T}}\vec{b} \mid \vec{z}^{\,\mathrm{T}}A = \vec{c}^{\,\mathrm{T}} \wedge \vec{z}^{\,\mathrm{T}} \geq \vec{0}^{\,\mathrm{T}} \wedge \vec{z}^{\,\mathrm{T}} \text{ is integral}\}$.

[6]The dual of an LP of the (non-standard) form

$$\begin{array}{l} \text{Maximize } c_1x_1 + \ldots + c_nx_n \\ \text{subject to} \\ \begin{array}{rcll} a_{i,1}x_1 + \ldots + a_{i,n}x_n & = & b_i, & 1 \leq i \leq p, \\ a_{i,1}x_1 + \ldots + a_{i,n}x_n & \leq & b_i, & p+1 \leq i \leq m, \\ x_j & \geq & 0 & 1 \leq j \leq q \end{array} \end{array}$$

with $0 \leq p \leq m$ and $0 \leq q \leq n$ can be defined as

$$\begin{array}{l} \text{Minimize } b_1z_1 + \ldots + b_mz_m \\ \text{subject to} \\ \begin{array}{rcll} a_{1,j}z_j + \ldots + a_{m,j}z_m & \geq & c_j, & 1 \leq j \leq q, \\ a_{1,j}z_j + \ldots + a_{m,j}z_m & = & c_j, & q+1 \leq j \leq n, \\ z_i & \geq & 0 & p+1 \leq i \leq m. \end{array} \end{array}$$

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

2.4.2 Balanced and Totally Unimodular Matrices

The problem Integer Linear Programming is NP-hard. However, there are special cases that can be solved in polynomial time. Here, we consider the two special cases where the coefficient matrix A of the ILP is "balanced" or "totally unimodular" and describe how these properties lead to polynomial-time solvability. We will see that matrices with the C1P have both of these properties. For a more detailed description of the matrix classes mentioned here see [BLS99, CCV06, Gol04, Sch86].

Balanced Matrices. Originally, balanced matrices have been defined by Berge [Ber70] as 0/1-matrices that do not contain a square submatrix of odd order with exactly two 1s per row and per column—the order of a matrix denotes the number of its entries. Truemper [Tru78] extended this definition to $0/\pm 1$-matrices in the following way.

Definition 2.10 ([Tru78])**.** A matrix with entries $0, 1, -1$ is *balanced* if for each submatrix B of A it holds that if B has exactly two non-zero entries per row and per column, then the sum of the entries of B is a multiple of four.

Berge also gave an alternative, useful characterization for balanced 0/1-matrices based on a "bicolorability" property of the columns of such a matrix [Ber70]. This characterization was extended by Conforti and Cornuéjols [CC95] to balanced $0/\pm 1$-matrices.

Theorem 2.8 ([CC95])**.** *An $m \times n$ matrix A with entries $0, 1, -1$ is balanced if and only if each collection of columns from A can be partitioned into two column sets C_1 and C_2 such that in each row r_i containing more than one non-zero entry*

- *there are two non-zero entries m_{i,j_1}, m_{i,j_2} such that the columns j_1 and j_2 either both belong to C_1 or both belong to C_2 and $m_{i,j_1} = -m_{i,j_2}$, or*
- *there are two non-zero entries m_{i,j_1}, m_{i,j_2} such that the column j_1 belongs to C_1 and the column j_2 belongs to C_2 and $m_{i,j_1} = m_{i,j_2}$.*

The following theorem shows that ILPs with balanced coefficient matrices are polynomial-time solvable for certain vectors $\vec{b}$; the theorem was originally formulated for balanced 0/1-matrices only [Ber72, FHO74] and was extended to balanced $0/\pm 1$-matrices by Conforti and Cornuéjols [CC95].

Theorem 2.9 ([CC95])**.** *Let A be a $0/\pm 1$-matrix, and for a submatrix B of A, let $\vec{n}(B)$ denote the column vector whose ith entry equals the number of -1-entries in the ith row of B. Then the following statements are equivalent:*

1. *The matrix A is balanced.*

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

2. *For each $m \times n$ submatrix B of A, the polyhedron defined by*

$$B\vec{x} \leq 1^m - \vec{n}(B)$$
$$0^n \leq \vec{x} \leq 1^n$$

is integral.[7]

3. *For each $m \times n$ submatrix B of A, the polyhedron defined by*

$$B\vec{x} = 1^m - \vec{n}(B)$$
$$0^n \leq \vec{x} \leq 1^n$$

is integral.

4. *For each $m \times n$ submatrix B of A, the polyhedron defined by*

$$B\vec{x} \geq 1^m - \vec{n}(B)$$
$$0^n \leq \vec{x} \leq 1^n$$

is integral.[8]

In particular, Theorem 2.9 implies that for any balanced matrix A and any vector $\vec{c}^{\,\mathrm{T}}$, the ILP

$$\begin{array}{lll} \text{Maximize } \vec{c}^{\,\mathrm{T}}\vec{x} & & \\ \text{subject to} & & \\ A\vec{x} & \leq & \vec{b} \\ x_j & \in & \{0,1\} \quad \text{for every entry } x_j \text{ of } \vec{x} \end{array}$$

can be solved in polynomial time if $\vec{b} = 1^m - \vec{n}(A)$. Moreover, if $\vec{c}^{\,\mathrm{T}}$ is integral, then the optimum value of the ILP, if bounded, is an integer value.

Totally Unimodular Matrices. In the case of balanced coefficient matrices A, the polyhedron defined by $A\vec{x} \leq \vec{b}$ is integral only for certain vectors $\vec{b}$. However, there also exist matrices with the property that the polyhedron is integral for *every* integral vector $\vec{b}$. As we will see, these matrices coincide with so-called totally unimodular matrices, which are defined as follows.

Definition 2.11. A $0/\pm 1$-matrix is *totally unimodular* if every square submatrix has determinant 0, 1, or -1.

Similar to Theorem 2.8 for balanced matrices, there is a useful "bicolorability" characterization for totally unimodular matrices.

[7] A $0/\pm 1$-submatrix B that has the property mentioned in statement 2 is called *perfect*.
[8] A $0/\pm 1$-submatrix B that has the property mentioned in statement 4 is called *ideal*.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Theorem 2.10 ([Gho62]). *An $m \times n$ matrix A with entries $0, 1, -1$ is totally unimodular if and only if each collection of columns from A can be partitioned into two column sets such that in each row the sum of the entries of the first set and the sum of the entries of the second set differ by at most 1.*

Note that Theorems 2.10 and 2.8 immediately imply that every totally unimodular matrix is balanced.

The following theorem shows the polynomial-time solvability of ILPs with totally unimodular coefficient matrices.

Theorem 2.11 ([HK56]). *Let A be an $m \times n$ integral matrix. Then the polyhedron defined by*

$$\begin{aligned} A\vec{x} &\leq \vec{b} \\ \vec{x} &\geq \vec{0} \end{aligned}$$

is integral for every integral vector $\vec{b} \in \mathbb{Z}^m$ if and only if A is totally unimodular.

From Theorem 2.11 it follows that for every totally unimodular matrix A, every integral vector $\vec{b}$ and every vector $\vec{c}^{\,\mathrm{T}}$ the ILP

$$\begin{array}{lll} \text{Maximize } \vec{c}^{\,\mathrm{T}}\vec{x} & & \\ \text{subject to} & & \\ A\vec{x} & \leq & \vec{b} \\ \vec{x} & \geq & \vec{0} \\ \vec{x} & \text{is integral} & \end{array} \tag{2.3}$$

can be solved with $O((n^3/\ln n)L)$ arithmetic operations [Ans99], where L is the total bit number needed for encoding the ILP.

2.4.3 ILPs with Coefficient Matrices having the C1P

The first method to solve ILPs whose coefficient matrices have the C1P is to use the fact that any matrix A having the C1P clearly fulfills the conditions of Theorem 2.10 and, hence, is totally unimodular. To see this, consider an arbitrary collection of columns from A and order them according to the C1P. Partitioning the columns by putting every second column, starting with the first, into one column set and every remaining column into the other column set leads to a partitioning as required in Theorem 2.10 (see also [NW88, page 544]). Therefore, if a matrix A has the C1P, then for every integral vector $\vec{b}$ and every vector $\vec{c}^{\,\mathrm{T}}$ the ILP shown in (2.3) can be solved in polynomial time due to Theorem 2.11: just omit the constraint "$\vec{x}$ is integral" and solve the resulting LP, which has always an integral solution.

Using Theorem 2.11 to solve ILPs with coefficient matrices having the C1P exploits only the fact that such coefficient matrices are totally unimodular. However, it is known that an ILP whose coefficient matrix has the C1P can be solved

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

even faster by transforming it into an edge-weighted graph and solving a shortest-path problem or a minimum-cost flow problem on this graph, depending on whether the decision version or the optimization version of Integer Linear Programming is considered (see [VW62] and [AMO93, pages 304–306] for the transformation of the optimization version into the flow problem and [AMO93, pages 310–315] for the connection between minimum-cost flow problems and shortest-path problems; see also [NW88, pages 546–550]). The running time obtained in this way is $O(mn)$ for the decision version and $O(m^2 \log(n) + mn \log(n)^2)$ for the optimization version; this approach is, therefore, much faster than using Theorem 2.11, where $O((n^3/\ln n)L)$ operations [Ans99] are needed—L is the size of the ILP and, hence, lower-bounded by mn.

We start with showing how to solve the decision version of Integer Linear Programming for coefficient matrices with the C1P by reducing it to a shortest path problem. We assume that the coefficient matrix A of the given ILP has m rows and that the rows of A and the entries of $\vec{b}$ are sorted in such a way that the first $m' \leq m$ of these rows contain only entries from $\{0, 1\}$ and the remaining $m - m'$ rows contain only entries from $\{0, -1\}$. Moreover, we assume that A has the strong C1P, which is not a restriction since a C1-ordering for A's columns can be found in linear time (see Section 2.3). With $\mathrm{lx}(i)$ and $\mathrm{rx}(i)$ we denote the column index of the first and the last, respectively, non-zero entry in the ith row of A. Hence, an instance of the problem to be solved consists of an inequation system as follows.

$$\begin{array}{rcll} x_{\mathrm{lx}(i)} + x_{\mathrm{lx}(i)+1} + \ldots + x_{\mathrm{rx}(i)} & \leq & b_i & \forall i \in \{1, \ldots, m'\} \\ -x_{\mathrm{lx}(i)} - x_{\mathrm{lx}(i)+1} - \ldots - x_{\mathrm{rx}(i)} & \leq & b_i & \forall i \in \{m'+1, \ldots, m\} \\ x_j & \in & \mathbb{Z} & \forall j \in \{1, \ldots, n\} \end{array} \tag{2.4}$$

To transform the inequation system into a graph, we first drop the constraint of integrality and replace the n variables $x_1, \ldots, x_n$ by $n + 1$ variables $y_0, \ldots, y_n$ such that $x_j = y_j - y_{j-1}$ for all $j \in \{1, \ldots, n\}$. This yields the following inequation system.

$$\begin{array}{rcll} -y_{\mathrm{lx}(i)-1} + y_{\mathrm{rx}(i)} & \leq & b_i & \forall i \in \{1, \ldots, m'\} \\ y_{\mathrm{lx}(i)-1} - y_{\mathrm{rx}(i)} & \leq & b_i & \forall i \in \{m'+1, \ldots, m\} \end{array} \tag{2.5}$$

In the resulting coefficient matrix, each row contains exactly one 1 and one -1; hence, each row can be interpreted as a directed edge in a graph G whose vertices correspond to the variables $y_0, \ldots, y_n$. More precisely, let $G = (V, E)$ be the directed edge-weighted graph with

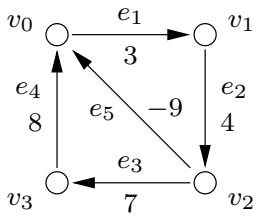
$$\begin{aligned} V = \{v_j \mid & \text{ the inequation system (2.5) contains a variable } y_j\}, \\ E = \{(v_{j_1}, v_{j_2}) \mid & \text{ the inequation system (2.5) contains an inequation} \\ & \text{whose left side is } -y_{j_1} + y_{j_2}\}, \end{aligned}$$

where every edge $e \in E$ has a weight that is equal to the right side of the inequation corresponding to e in the inequation system (2.5), see Figure 2.14.

Now consider the following statement known as *Farkas' Lemma* (see [Sch86]).

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

$$
\begin{array}{rrrrll}
-y_0 & +y_1 & & & \leq 3 & \text{(interpreted as edge } e_1\text{)} \\
 & -y_1 & +y_2 & & \leq 4 & \text{(interpreted as edge } e_2\text{)} \\
 & & -y_2 & +y_3 & \leq 7 & \text{(interpreted as edge } e_3\text{)} \\
y_0 & & & -y_3 & \leq 8 & \text{(interpreted as edge } e_4\text{)} \\
y_0 & & -y_2 & & \leq -9 & \text{(interpreted as edge } e_5\text{)}
\end{array}
$$



$$
\begin{array}{rrrrrll}
-z_1 & & & +z_4 & +z_5 & = 0 & \text{(interpreted as flow conservation at } v_0\text{)} \\
z_1 & -z_2 & & & & = 0 & \text{(interpreted as flow conservation at } v_1\text{)} \\
 & z_2 & -z_3 & & -z_5 & = 0 & \text{(interpreted as flow conservation at } v_2\text{)} \\
 & & z_3 & -z_4 & & = 0 & \text{(interpreted as flow conservation at } v_3\text{)} \\
3z_1 & +4z_2 & +7z_3 & +8z_4 & -9z_5 & < 0 & \\
 & & & z_1, \ldots, z_5 & & \geq 0 &
\end{array}
$$

Figure 2.14: Solving an ILP whose coefficient matrix has the C1P. Top: An example for the inequation system (2.5) obtained from an ILP with the C1P. Every row can be interpreted as an edge in a directed, edge-weighted graph G. This graph is displayed in the middle. Bottom: This inequation system is *not* feasible iff the the inequation system displayed at the top of the figure is feasible (Farkas' Lemma). The inequation system at the bottom can be interpreted as a flow problem.

Lemma 2.2. *Let A be an $m \times n$ matrix with entries from $\mathbb{R}$, and let $\vec{b} \in \mathbb{R}^m$ be a vector. Then the inequation system $A\vec{y} \leq \vec{b}$ has a solution $\vec{y} \in \mathbb{R}^n$ if and only if the inequation system $\vec{z}^{\,T}A = (0^n)^T, \vec{z}^{\,T}\vec{b} < 0, \vec{z} \geq 0^m$ has* no *solution $\vec{z} \in \mathbb{R}^m$.*

Applying Farkas' Lemma to the inequation system (2.5), the lemma says that the inequation system is feasible iff G contains no *negative cycle*, that is, no directed cycle in which the sum of the edge weights is negative. To see this, observe that by interpreting the edge weights b_i as "per-flow-costs", the inequation system $\vec{z}^{\,\mathrm{T}}A = (0^n)^{\mathrm{T}}, \vec{z}^{\,\mathrm{T}}\vec{b} < 0, \vec{z} \geq 0^m$ in Farkas' Lemma can be interpreted as the following "negative-cost" flow problem on the graph G: Find a flow function $f : E \to \mathbb{R}$ such that

- the flow $f(e_i)$ along every directed edge $e_i \in E$ is nonnegative (expressed by the constraint $\vec{z} \geq 0^m$),

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

- for every vertex the sum of the ingoing and the outgoing flow is 0 (expressed by the constraint $\vec{z}^{\mathrm{T}}A = (0^n)^{\mathrm{T}}$), and
- the sum $\sum_{i=1}^{m} b_i f(e_i)$ of all costs arising from sending flow along the edges is negative (expressed by the constraint $\vec{z}^{\mathrm{T}}\vec{b} < 0$).

See Figure 2.14. If a flow f has these three properties, then setting $z_i = f(e_i)$ clearly yields a feasible solution for the LP. It is easy to see that a flow with negative cost can only exist if the graph G contains a negative cycle: decompose any negative cost flow, if existing, into a set of cyclic flows (that is, a set of possibly overlapping directed cycles with each edge in every cycle carrying exactly one unit of flow); at least one of these flows must go along a cycle in which the sum of the edge weights is negative. By using the Bellmann-Ford-Moore-Algorithm (see [CLRS01]), it can be decided in $O(|V| \cdot |E|)$ time whether G contains a negative cycle. Hence, the decision version of Integer Linear Programming with C1P can be decided in $O(n \cdot m)$ time.

If G contains no negative cycle and a solution for the inequation system (2.5) shall be constructed (that is, the values of the y_j shall be computed), then just select an arbitrary $k \in \{0, \ldots, n\}$ and set y_k to 0. For every $j \in \{0, \ldots, n\} \setminus \{k\}$ for which there exists no directed path from v_k to v_j in G, add an edge (v_k, v_j) of weight $|E| \cdot \max\{-b_i \mid i \in \{1, \ldots, m\} \wedge b_i < 0\}$. Note that this operation does not create any negative cycles; note also that in the resulting graph G' every vertex is reachable from v_k on a directed path. Now, for every $j \in \{0, \ldots, n\} \setminus \{k\}$, set y_j to the length of the shortest path in G' from v_k to v_j. Since G' contains no negative cycle, these shortest paths are all well-defined. It is easy to see that this solution satisfies all inequations of the inequation system (2.5): Consider an arbitrary inequation from (2.5), and let $-y_{j_1} + y_{j_2}$ be its left side. Then the edge (v_{j_1}, v_{j_2}) ensures that the distance from v_k to v_{j_2} is upper-bounded by the distance from v_k to v_{j_1} plus the right side of the inequation, and, thus, the inequation is satisfied by the the described values of y_{j_1} and y_{j_2}. The shortest paths can be computed by the Bellmann-Ford-Moore-Algorithm in $O(|V| \cdot |E|) = O(n \cdot m)$ time. A solution for the original ILP (2.4) can be computed by setting $x_j = y_j - y_{j-1}$ for all $j \in \{1, \ldots, n\}$.

Now we turn our attention to the optimization version of the problem, that is, instances of the form

$$
\begin{array}{l}
\text{Maximize } c_1 x_1 + c_2 x_2 + \ldots + c_n x_n \\
\text{subject to} \\
\begin{array}{rcll}
x_{\mathrm{lx}(i)} + x_{\mathrm{lx}(i)+1} + \ldots + x_{\mathrm{rx}(i)} & \leq & b_i & \forall i \in \{1, \ldots, m'\} \\
-x_{\mathrm{lx}(i)} - x_{\mathrm{lx}(i)+1} - \ldots - x_{\mathrm{rx}(i)} & \leq & b_i & \forall i \in \{m'+1, \ldots, m\} \\
x_j & \in & \mathbb{Z} & \forall j \in \{1, \ldots, n\}
\end{array}
\end{array}
\tag{2.6}
$$

Here we have to use a minimum-cost flow algorithm instead of a shortest path algorithm. Again, we start with omitting the integrality constraint and replacing

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

the variables $x_1, \ldots, x_n$ by variables $y_0, \ldots, y_n$, yielding the following LP.

$$\begin{array}{l} \text{Maximize } -c_1y_0 + (c_1 - c_2)y_1 + \ldots + (c_{n-1} - c_n)y_{n-1} + c_ny_n \\ \text{subject to} \\ \begin{array}{rcll} -y_{\mathrm{lx}(i)-1} + y_{\mathrm{rx}(i)} & \leq & b_i & \forall i \in \{1, \ldots, m'\} \\ y_{\mathrm{lx}(i)-1} - y_{\mathrm{rx}(i)} & \leq & b_i & \forall i \in \{m'+1, \ldots, m\} \end{array} \end{array} \tag{2.7}$$

Let $A' = (a'_{i,j})$ be the coefficient matrix of the inequation system in (2.7) with exactly one 1 and one -1 in each of its rows. Now consider the dual problem displayed below, where an m-entry vector $\vec{z}^{\,\mathrm{T}}$ has to be found.

$$\begin{array}{l} \text{Minimize } b_1z_1 + b_2z_2 + \ldots + b_mz_m \\ \text{subject to} \\ \begin{array}{rcll} a'_{1,0}z_1 + a'_{2,0}z_2 + \ldots + a'_{m,0}z_m & = & -c_1 & \\ a'_{1,j}z_1 + a'_{2,j}z_2 + \ldots + a'_{m,j}z_m & = & c_j - c_{j+1} & \forall j \in \{1, \ldots, n-1\} \\ a'_{1,n}z_1 + a'_{2,n}z_2 + \ldots + a'_{m,n}z_m & = & c_n & \\ z_i & \geq & 0 & \forall i \in \{1, \ldots, m\} \end{array} \end{array} \tag{2.8}$$

Note that the inequation system $\vec{z}^{\,\mathrm{T}}A = (0^n)^{\mathrm{T}}, \vec{z}^{\,\mathrm{T}}\vec{b} < 0, \vec{z} \geq 0^m$ mentioned in Farkas' Lemma (Lemma 2.2) can be seen as a special case of the inequation system in (2.8): just set all c_j to 0 in the latter inequation system. The LP (2.8) can now be interpreted as a minimum-cost flow problem on a directed, vertex-weighted and edge-weighted graph $G = (V, E)$ with $n + 1$ vertices $v_0, \ldots, v_n$ and m edges, where G is constructed in analogy to the graph used for solving the decision problem: the ith column on the left side of the inequation system in (2.8) corresponds to a directed edge e_i, which is labeled with a "per-flow-cost" b_i and leads from v_{j_1} to v_{j_2}, where j_1 and j_2 are the two indices that satisfy $a'_{i,j_1} = -1$ and $a'_{i,j_2} = 1$. In addition, every vertex v_j has a "flow demand" that is equal to the right side of the jth inequation in the LP (2.8). The flow problem that has to be solved on G is: Find a flow function $f : E \to \mathbb{R}$ such that

- the flow $f(e_i)$ along every directed edge $e_i \in E$ is nonnegative,
- for every vertex v_j the difference between the sum of the incoming flows and the sum of the outgoing flows equals the flow demand of v_j, that is,

$$\sum_{(v_i,v_j)\in E} f((v_i, v_j)) - \sum_{(v_j,v_i)\in E} f((v_j, v_i)) = \begin{cases} -c_1 & \text{if } j = 0 \\ c_n & \text{if } j = n \\ c_j - c_{j+1} & \text{otherwise,} \end{cases}$$

  and

- the sum $\sum_{i=1}^{m} b_i f(e_i)$ of all costs arising from sending flow along the edges is minimized.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

This flow problem is the optimization version of the flow problem that we have constructed for the inequation system in Farkas' Lemma; however, here we cannot reduce the flow problem to the problem of computing shortest paths.

Computing a minimum-cost flow f as desired and setting $z_i = f(e_i)$ clearly yields an optimal solution for the LP (2.8). Such a minimum-cost flow can be found in $O(m^2 \log(n) + mn \log(n)^2)$ time [AMO93]. Moreover, if the vector $\vec{c}^{\,\mathrm{T}}$ is integral, one can always find an integral minimum-cost flow, and if, in addition, the vector $\vec{b}$ is integral, then the integral optimal solution $\vec{z}^{\,\mathrm{T}}$ for the LP (2.8) corresponds to an integral optimal solution for the LP (2.7) and, hence, to an optimal solution for the ILP (2.6).

2.4.4 ILPs with Coefficient Matrices having the Circ1P

Not all matrices that have the Circ1P are totally unimodular. For example, all matrices M_{I_k} (see Figure 2.8) with even k are totally unimodular, while all matrices M_{I_k} with odd k are not (this can easily be seen by using the characterization of Theorem 2.10). Nevertheless, every ILP whose coefficient matrix has the Circ1P can be solved in polynomial time by solving a series of ILPs that all have the C1P [BOR80] (see also [AMO93, page 346–347] and [HT02]).

To solve a given ILP

$$\begin{array}{l} \text{Maximize } c_1x_1 + c_2x_2 + \ldots + c_nx_n \\ \text{subject to} \\ \begin{array}{rcll} a_{1,1}x_1 + a_{1,2}x_2 + \ldots + a_{1,n}x_n & \leq & b_i & \forall i \in \{1, \ldots, m\} \\ x_j & \in & \mathbb{Z} & \forall j \in \{1, \ldots, n\} \end{array} \end{array} \tag{2.9}$$

whose $0/\pm 1$-coefficient matrix $A = (a_{i,j})$ has the Circ1P, define L_k, $k \in \mathbb{Z}$, as the ILP that results from appending the constraint

$$x_1 + x_2 + \ldots + x_n \;=\; k \tag{2.10}$$

to the the ILP (2.9). It is obvious that the ILP (2.9) is feasible iff there is a value k such that L_k is feasible. Moreover, if the ILP (2.9) is feasible, then there is a k such that the optimal solution for L_k is an optimal solution for the ILP (2.9): just set k to the sum of the x_i in an optimal solution for the ILP (2.9). Now, any ILP L_k can be transformed into an ILP having the C1P: Add the equation (2.10) to every inequation of L_k whose coefficients are from $\{0, -1\}$ and in which the non-zero coefficients do not appear consecutively, and subtract the equation (2.10) from every inequation of L_k whose coefficients are from $\{0, 1\}$ and in which the non-zero coefficients do not appear consecutively. The resulting ILP is equivalent to the ILP L_k (that is, every feasible solution of the latter ILP is a feasible solution for L_k and vice versa) and has the C1P—therefore, it can be solved with the approach described in Section 2.4.3. The optimum value of k can be determined by a binary search [AMO93, BOR80], such that the number of ILPs that have to be solved is linear in the size of the given ILP (2.9).

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

2.4.5 ILPs with Coefficient Matrices having the C1P/Circ1P for Columns

The methods described in Sections 2.4.3 and 2.4.4 can also be applied to ILPs whose coefficient matrices have the C1P (Circ1P) for columns instead of the C1P (Circ1P) for rows. To this end, note that if the coefficient matrix of an LP has the C1P (Circ1P) for columns, then the coefficient matrix of its dual has the C1P (Circ1P) for rows. Therefore, an ILP with the C1P (Circ1P) for columns can be solved by applying the method of Section 2.4.3 (Section 2.4.4) to its dual. More precisely, since the dual of a maximization LP is a minimization LP and contains equations instead of inequations, first conform it to the standard form (as defined in Section 2.4.1) by multiplying the objective function with -1 and replacing each equation by two inequations, and then solve it.

2.5 Set Cover on Input Matrices having the C1P/Circ1P

The C1P has attracted interest not least because it often makes hard problems easy. Our first example substantiating this statement was Integer Linear Programming in Section 2.4. As a second example, we consider the problem Set Cover.

Set Cover is a very "general" NP-complete problem; that is, a huge number of natural NP-complete problems, as for example Vertex Cover and Independent Set, can be reduced to Set Cover by very simple reductions. Formulated as a matrix problem, Set Cover is defined as follows.

Set Cover
Input: A binary matrix M and a positive integer k.
Question: Is there a set C' of at most k columns of M such that the submatrix M' of M that is induced by these columns has at least one 1 in every row?

The reader may be familiar with Set Cover as a subset selection problem; however, the equivalence of our definition and the more common definition of Set Cover as a subset problem can easily be seen by identifying columns with subsets and rows with elements to be covered.

Due to its generality, Set Cover has practical applications in almost all disciplines (see [CLRS01, CP93, CTF00]); unfortunately, Set Cover is not only NP-hard, but it allows only for a logarithmic-factor polynomial-time approximation [Fei98]. Moreover, Set Cover is W[2]-complete (that is, parameterized intractable) with respect to the parameter k = "solution size" [DF99]. In the *weighted version* of Set Cover, each column of the given matrix M has a positive integral weight and one asks for a column set C' of *weight* at most k.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Input: A binary matrix M with the strong C1P.
Output: A minimum-cardinality subset C' of M's columns that contains at least one 1 from each row of M.

1: $C' := \emptyset$;
2: **while** M has at least one row: {
3: $r :=$ a row from M with minimum $\mathrm{rx}(r)$;
4: $C' := C' \cup \{c_{\mathrm{rx}(r)}\}$;
5: delete all rows from M that have a 1 in column $c_{\mathrm{rx}(r)}$; }
6: **return** C';

Figure 2.15: Greedy algorithm for solving Set Cover. We denote the columns of M with $c_1, \ldots, c_n$, ordered from left to right. For a row r of M, we denote with $\mathrm{rx}(r)$ the index of the rightmost column having a 1 in row r.

Set Cover with the C1P/Circ1P. Whereas Set Cover in general is NP-complete, the problem becomes polynomial-time solvable when the input matrix M has the C1P or the Circ1P. To solve such a restricted instance of Set Cover, one can formulate the problem in a straightforward way as an ILP that has one variable for each column of M and whose coefficient matrix is M. As described in Section 2.4, this ILP is polynomial-time solvable. Moreover, there is a well-known greedy algorithm for Set Cover on input matrices with the C1P: First order the columns of M such that in each row the 1s appear consecutively (this takes linear time, see Section 2.3), and then proceed from left to right as shown in the pseudocode in Figure 2.15. The correctness of the algorithm in Figure 2.15 is easy to see: The row r selected in line 3 in each execution of the main loop does not contain a 1 in a column that belongs to the set C' constructed so far—otherwise, r would have been deleted from M already. Therefore, one has to add a column to C' that contains a 1 in row r. Since, due to the selection of r, no row in M contains a block of 1s whose rightmost column has an index smaller than $\mathrm{rx}(r)$, it is always optimal to choose $c_{\mathrm{rx}(r)}$.

If the input matrix M has the C1P or the Circ1P, even the weighted version of Set Cover can be solved in polynomial time: one can either use the ILP approach or, in case of input matrices having the C1P, use a simple dynamic programming algorithm. The following theorem summarizes these known results.

Theorem 2.12. Weighted Set Cover *can be solved in polynomial time if the input matrix has the C1P or the Circ1P (for rows or for columns).*

A further approach for tackling Set Cover is to use the following well-known polynomial-time data reduction rules, whose correctness is obvious.

Dominated row: If M contains two rows r_{i_1}, r_{i_2} such that for each column c_j it holds that $m_{i_1,j} = 1$ implies $m_{i_2,j} = 1$, then remove row r_{i_2} from M.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Dominated column: If M contains two columns c_{j_1}, c_{j_2} such that for each row r_i it holds that $m_{i,j_1} = 1$ implies $m_{i,j_2} = 1$, then remove column c_{j_1} from M.

Useless column: If M contains a column c_j without any 1-entry, then remove column c_j from M.

Unique column: If M contains a row r_i that contains exactly one 1-entry $m_{i,j}$, then remove r_i and all rows $r_{i'}$ with $m_{i',j} = 1$ from M, remove column c_j from M, and decrease k by one.

Yes **instance:** If $k \geq 0$ and M has no rows, then answer "M is a *yes*-instance."

No **instance:** If M contains a row without any 1-entry, or if $k < 0$, or if $k = 0$ and M contains at least one row, then answer "M is a *no*-instance."

For Set Cover without restrictions, these rules can be used in a preprocessing step in order to decrease the size of a problem instance before solving it. However, since there are Set Cover instances to which none of the rules applies, it is not possible to give any guarantee on the size of the problem instance resulting from the preprocessing step. If, however, the input matrix M has the C1P (for rows or for columns), then the instance can be solved by iteratively applying the rules, that is, eventually one of the rules will output "M is a *yes*-instance" or "M is a *no*-instance."

Set Cover with "almost C1P". Motivated by problems arising from railway optimization, Mecke and Wagner [MW04], Ruf and Schöbel [RS04], and Mecke et al. [MSW05] consider Weighted Set Cover on input matrices that have "almost 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 1s by 0s [MW04], that the average number of blocks of 1s per row is much smaller than the number of columns of the matrix [RS04], or that the maximum number of blocks of 1s per row is small [MSW05]. Apart from heuristics performing well in practice [MW04, RS04], the following results have been obtained.

Theorem 2.13 ([MW04, MSW05]). *1.* Set Cover *is* NP*-complete even if the input matrix M can be split into two submatrices M_1, M_2 such that $M = (M1 \mid M_2)$ and both M_1 and M_2 have the strong C1P.*

2. Weighted Set Cover *restricted to input matrices with at most d blocks of 1s per row can be approximated in polynomial time with a factor d.*

3. Weighted Set Cover *can be solved in $2^{\ell} \cdot \mathrm{poly}(m, n)$ time with ℓ denoting the maximum distance between the topmost and the bottommost 1 in any column of M.*

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Note that statement 1 follows also from an earlier result by Gaur et al. [GIK02]. Chapters 5 and 6 also deal with variants of Set Cover where the input matrices are "close" to the C1P in some sense: Chapter 5 considers a generalization of Set Cover on input matrices in which some, but not all rows contain only one block of 1s. In Chapter 6, we analyze the parameterized complexity of Set Cover on matrices having a constant number $d \geq 2$ of blocks of 1s per row.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Chapter 3

Finding Forbidden Submatrices

This chapter deals with the problem of finding in a given binary matrix M a submatrix of small size that does not have the C1P. Each such submatrix is a cause for the absence of the C1P in M; we give several algorithms for detecting these trouble spots.

3.1 Introduction and Overview

From Section 2.2, we know that matrices having the C1P can be characterized by a set T of forbidden submatrices (see Theorem 2.5 and Figure 2.8): A matrix has the C1P iff it does not contain a matrix from T as a submatrix. In this chapter, we show how to find, given a binary matrix M, a submatrix of M that belongs to the set T and has a minimum number of rows, columns, rows and columns, or entries. Note that neither the known polynomial-time algorithms for deciding whether a given matrix has the C1P (see Section 2.3) nor the known algorithms for finding an asteroidal triple in a graph (see [Köh04]) output a minimum-size submatrix from T or a minimum-size induced subgraph containing an asteroidal triple.

One motivation for finding small forbidden submatrices is the following NP-hard problem: Given a matrix M and an integer k, delete at most k rows or columns such that the resulting matrix has the C1P. The problem is called Min-COS-R or Min-COS-C, depending on if we are allowed to delete rows or columns. We will consider both problems in detail in Chapter 4, where we use the characterization given in Theorem 2.5 for developing polynomial-time approximation algorithms and fixed-parameter algorithms for Min-COS-R and Min-COS-C. All algorithms presented in Chapter 4 for these problems iteratively search and destroy in the input matrix every submatrix that is isomorphic to one of the forbidden submatrices from the set T: In the approximation scenario *all* rows or columns belonging to every found forbidden submatrix are deleted, whereas in the fixed-parameter setting a search tree algorithm branches recursively into several subcases—deleting in each case *one* of the rows or columns of

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

the found forbidden submatrix. The approximation factor of the approximation algorithm as well as the running time of the fixed-parameter-algorithm directly depend on the maximum number of rows or columns of a forbidden submatrix found during the execution of the algorithms: If x denotes this number of rows or columns, then, in the first case, the approximation factor is x and, in the second case, the number of nodes in the search tree is $O(x^k)$. Moreover, the running times of both algorithms obviously depend also on the time needed for finding a forbidden submatrix. Therefore, efficiently detecting small forbidden submatrices from T is a crucial issue concerning the performance of these algorithms.

To state the running times of the algorithms more precisely, we give the running times as functions not only depending on the size of the input matrix, but also on the maximum number of 1s occurring in a row of the input matrix; this number will be denoted with Δ. In the case of sparse matrices as occurring in many applications [AM96, DER89, KS96, TZ07, WR00], the number Δ can be much smaller than the number of columns of the input matrix. We denote matrices that contain at most Δ 1s per row (but arbitrary many 1s per column) as $(*, \Delta)$-*matrices* (see also Chapter 4).

The remainder of the chapter is structured as follows. In Section 3.2, we present an algorithm that outputs, given a binary matrix M that contains a submatrix from T with m' rows and n' columns, a submatrix of M that belongs to T and consists of at most $m' + 5$ rows and $n' + 3$ columns. In Section 3.3, we give several algorithms that find a forbidden submatrix from T of minimum size, at the cost of an increased running time compared to the algorithm presented in Section 3.2.

3.2 Approximating the Minimum-Size Forbidden Submatrix

The approach used here is to exploit the following characterization via asteroidal triples (see Definition 2.8), which is a direct consequence of Theorem 2.3 due to Tucker [Tuc72].

Corollary 3.1. *A matrix M has the C1P if and only if its representing bipartite graph G_M does not contain an asteroidal triple whose three vertices correspond to columns of M.*

Using Corollary 3.1, a forbidden submatrix from T in a given matrix M can be found as follows: For every vertex triple u, v, w in G_M corresponding to columns of M, determine the sum of the lengths of three shortest paths connecting u with v, u with w, and v with w, respectively, each time avoiding the closed neighborhood of the third vertex. If all three paths exist, then the vertices u, v, w form an asteroidal triple in G_M. Select a triple u, v, w where the sum is minimum compared to all other triples, and return the rows and columns of M that correspond to the vertices of the three shortest paths computed for this triple. The

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Input: A binary matrix M.
Output: A submatrix M' from T occurring in M.

1: construct M's representing graph $G_M = (R, C, E)$; // R corresponds to rows
// and C to columns
2: **for each** vertex $u \in C$: {
3: $G^u := G[(R \cup C) \setminus N[u]]$;
4: **for each** vertex $v \in C \setminus \{u\}$: {
5: compute the lengths of all shortest paths in G^u that start in v; }}
6: choose $u, v, w \in C$ such that $|P^u_G(v, w)| + |P^v_G(u, w)| + |P^w_G(u, v)|$ is minimum;
7: $V' := P^u_G(v, w) \cup P^v_G(u, w) \cup P^w_G(u, v)$;
8: $M' :=$ the submatrix of M whose rows and columns correspond to V';
9: **while** M' contains a row r such that M' without r does not have the C1P or a column c such that M' without c does not have the C1P: {
10: delete r or c, respectively, from M'; }
11: **return** M';

Figure 3.1: Algorithm for finding forbidden submatrices.

returned submatrix must contain a submatrix from T because the corresponding vertices in G_M induce a subgraph that contains an asteroidal triple. However, this procedure does not always return a submatrix of minimum size, because the sum of the lengths of the three paths computed for a triple u, v, w is not always the number of vertices in the union of the three paths—some vertices may be part of more than one path. More specifically, assume that the algorithm selects a triple u, v, w where the sum of the shortest paths is minimum and where the shortest paths are vertex-disjoint except for u, v, w. Then there can be another triple u', v', w' with the same sum but whose three shortest paths have several vertices in common. Selecting this triple would lead to a submatrix of smaller size. In what follows, we will analyze the size of the returned matrix and show that it contains at most five more rows and three more columns than a forbidden submatrix with minimum number of rows or minimum number of columns.

For a graph $G = (V, E)$ and an asteroidal triple $u, v, w \in V$ of G, we denote with $P^u_G(v, w)$ the vertex set of a shortest path in $G[V \setminus N[u]]$ between v and w (including v and w). Figure 3.1 contains the pseudocode of the algorithm behind the above approach. The following proposition gives an upper bound on the numbers of rows and columns of the submatrix returned by the algorithm.

Proposition 3.1. *Let M be a $(*, \Delta)$-matrix of size $m \times n$ that contains a forbidden $m' \times n'$ submatrix M' from the set T shown in Figure 2.8. Then the algorithm in Figure 3.1 returns in $O(\Delta mn^2 + n^3)$ time a submatrix of M that belongs to T and has at most*

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

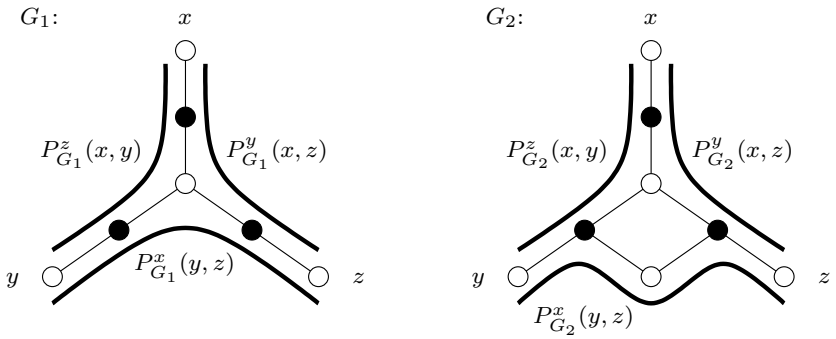


Figure 3.2: Example showing that the matrix constructed in line 8 of the algorithm in Figure 3.1 is not necessarily minimal: Both graphs G_1 and G_2 contain exactly one asteroidal triple x, y, z consisting of white vertices, which correspond to columns. The graph G_2 is not minimal, because it contains G_1 as an induced subgraph. However, the number of vertices in the three paths that are displayed with bold lines between x, y, and z are the same: $|P^x_{G_1}(y,z)|+|P^y_{G_1}(x,z)|+|P^z_{G_1}(x,y)| = |P^x_{G_2}(y,z)| + |P^y_{G_2}(x,z)| + |P^z_{G_2}(x,y)| = 15$.

m' *rows and* n' *columns*	*if* $M' = M_{\mathrm{I}_k}$,
m' *rows and* n' *columns*	*if* $M' = M_{\mathrm{II}_k}$,
$m'+3$ *rows and* $n'+2$ *columns*	*if* $M' = M_{\mathrm{III}_k}$,
$m'+5$ *rows and* $n'+3$ *columns*	*if* $M' = M_{\mathrm{IV}}$, *and*
$m'+1$ *rows and* n' *columns*	*if* $M' = M_{\mathrm{V}}$.

Proof. As discussed above, the returned matrix M' clearly contains a submatrix from T. Furthermore, the lines 9 and 10 of the pseudocode in Figure 3.1 ensure that M' is minimal in the sense that the representing graph of no proper submatrix of M' contains an asteroidal triple. Since the latter property also holds for every matrix in T (see [Tuc72, proof of Theorem 7]), the matrix M' must be one of the matrices from T. (Note that the lines 9 and 10 cannot be omitted, because the matrix M' constructed in line 8 must not necessarily be minimal, see Figure 3.2.)

Next, we prove the claimed row and column numbers of the returned matrix M'. Since M' does not have the C1P, the representing graph $G_{M'}$ of M' contains an asteroidal triple x, y, z corresponding to three columns of M' (Corollary 3.1). If $M' = M_{\mathrm{I}_k}$, then every triple of vertices corresponding to columns of M' is an asteroidal triple in $G_{M'}$. To see this, consider the graph type G_{I_k} in Figure 3.3 showing the representing graphs of the forbidden submatrices from T: For every triple of white vertices, there is a path between any two of the vertices of the triple that avoids the closed neighborhood of the third. If $M' \neq M_{\mathrm{I}_k}$, then there is exactly one asteroidal triple in $G_{M'}$. This can be seen by considering the graph types G_{II_k}–G_{V} in Figure 3.3: The white vertices x, y, z form an asteroidal triple; any other triple of white vertices contains two vertices that are not connected by a path avoiding the closed neighborhood of the third.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

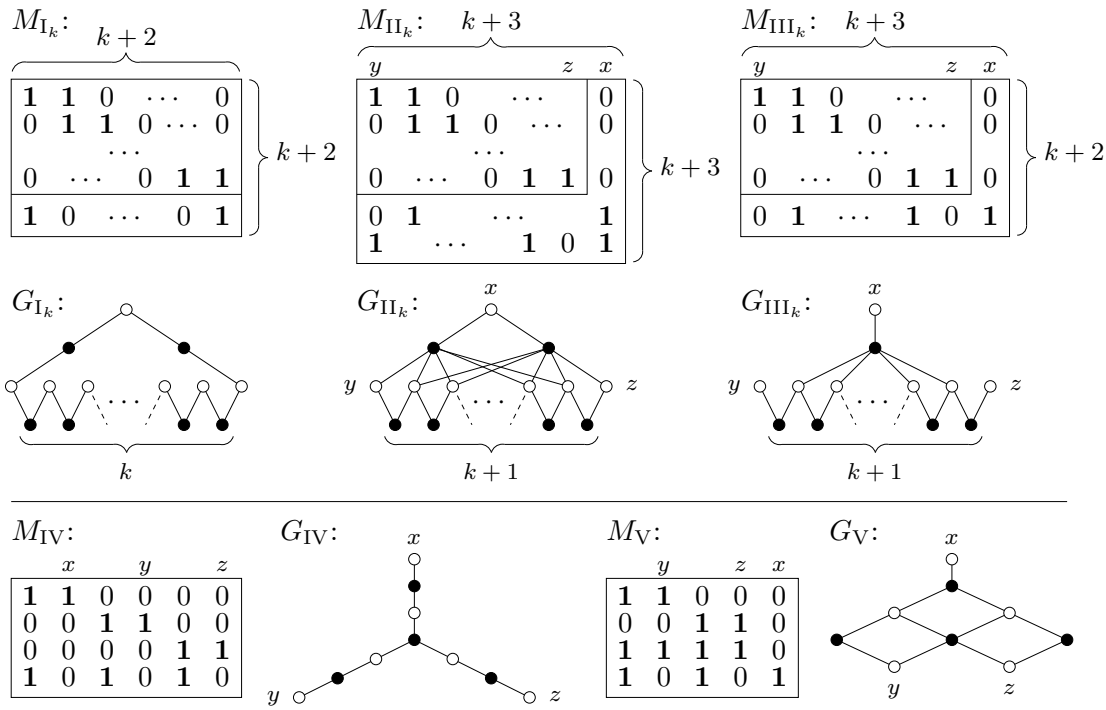


Figure 3.3: Representing graphs of the forbidden submatrices from T due to Tucker [Tuc72]. Black vertices correspond to rows, white vertices correspond to columns. The numbers k and $k+1$ refer to the number of black vertices in the lower parts of the graphs G_{I_k}–G_{III_k}. In the case of the graph G_{I_k}, every triple of white vertices is an asteroidal triple. In all other cases, there is exactly one asteroidal triple consisting of white vertices; this triple is denoted with x, y, z.

Let $p_{xyz} := |P^x_{G_{M'}}(y,z)| + |P^y_{G_{M'}}(x,z)| + |P^z_{G_{M'}}(x,y)|$. By considering the asteroidal triples in the graph types G_{I_k}–G_{V} (Figure 3.3) one can verify that

$$\begin{aligned}
p_{xyz} &= 2k+7 && \text{if } M' = M_{\mathrm{I}_k},\\
p_{xyz} &= 2k+9 && \text{if } M' = M_{\mathrm{II}_k},\\
p_{xyz} &= 2k+13 && \text{if } M' = M_{\mathrm{III}_k},\\
p_{xyz} &= 21 && \text{if } M' = M_{\mathrm{IV}}, \text{ and}\\
p_{xyz} &= 13 && \text{if } M' = M_{\mathrm{V}}.
\end{aligned}$$

For example, if $M' = M_{\mathrm{III}_k}$, then $|P^x_{G_{M'}}(y,z)| = 2k+3$, $|P^y_{G_{M'}}(x,z)| = 5$, and $|P^z_{G_{M'}}(x,y)| = 5$, and, hence, $p_{xyz} = (2k+3)+5+5 = 2k+13$.

Let $u, v, w \in C$ be the vertices chosen in line 6 of the algorithm, and let $p_{uvw} := |P^u_G(v,w)| + |P^v_G(u,w)| + |P^w_G(u,v)|$. Clearly, $p_{uvw} \leq p_{xyz}$ because u, v, w are selected such that $|P^u_G(v,w)| + |P^v_G(u,w)| + |P^w_G(u,v)|$ is minimized. The returned submatrix consists of at most $(p_{uvw}-3)/2 \leq (p_{xyz}-3)/2$ rows because each of the vertices u, v, w is counted twice in p_{uvw} and because every second vertex in each of the vertex sets $P^u_G(v,w), P^v_G(u,w), P^w_G(u,v)$ corresponds to a column

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

in M. It follows that the row number of the submatrix returned by the algorithm is upper-bounded by

$$\begin{array}{ll}
((2k+7)-3)/2 = k+2 = m' & \text{if } M' = M_{\mathrm{I}_k} \text{ (where } m' = k+2), \\
((2k+9)-3)/2 = k+3 = m' & \text{if } M' = M_{\mathrm{II}_k} \text{ (where } m' = k+3), \\
((2k+13)-3)/2 = k+5 = m'+3 & \text{if } M' = M_{\mathrm{III}_k} \text{ (where } m' = k+2), \\
(21-3)/2 = 9 = m'+5 & \text{if } M' = M_{\mathrm{IV}} \text{ (where } m' = 4), \text{ and} \\
(13-3)/2 = 5 = m'+1 & \text{if } M' = M_{\mathrm{V}} \text{ (where } m' = 4).
\end{array}$$

With a completely analogous argumentation it follows that the column number of the submatrix returned by the algorithm is upper-bounded by

$$\begin{array}{ll}
((2k+7)-3)/2 = k+2 = n' & \text{if } M' = M_{\mathrm{I}_k} \text{ (where } n' = k+2), \\
((2k+9)-3)/2 = k+3 = n' & \text{if } M' = M_{\mathrm{II}_k} \text{ (where } n' = k+3), \\
((2k+13)-3)/2 = k+5 = n'+2 & \text{if } M' = M_{\mathrm{III}_k} \text{ (where } n' = k+3), \\
(21-3)/2 = 9 = n'+3 & \text{if } M' = M_{\mathrm{IV}} \text{ (where } n' = 6), \text{ and} \\
(13-3)/2 = 5 = n' & \text{if } M' = M_{\mathrm{V}} \text{ (where } n' = 5).
\end{array}$$

To see the claimed running time, note that lines 2–5 can be executed in $O(n^2 \cdot (n + \Delta m))$ time by using breadth-first search (running on a graph $G = (V, E)$ in $O(|V| + |E|)$ time [CLRS01]) in line 5: the number of vertices in C is n, and the input graph G^u for the breadth first search has $m + n$ vertices and at most Δm edges. For considering all triples u, v, w in line 6, the algorithm needs $O(n^3)$ time. The test in line 9 can be executed in linear time (see Section 2.3), that is, in $O(m' + n' + \Delta m')$ time, and, hence, the time needed for lines 9–10 is dominated by the time needed for lines 1–8. □

3.3 Exact Algorithms for Finding Minimum-Size Forbidden Submatrices

In the previous section, we have seen how to find a forbidden submatrix whose number of rows and columns is close to be minimum. Now we turn our attention to the problem of finding a forbidden submatrix from T whose number of rows, columns, rows and columns, or entries is actually minimum, at the cost of an increased running time compared to the procedure presented in the previous section.

We first present in Section 3.3.1 two rather straightforward brute-force approaches, which are able to find any submatrix in a given matrix. These algorithms are efficient for finding submatrices of the types M_{IV} and M_{V} and for finding any submatrix from T in $(*, \Delta)$-matrices where Δ is small compared to the number of columns. In Section 3.3.2 we show how to find submatrices of the types M_{I_k}, M_{II_k}, and M_{III_k} in polynomial time even in matrices with a large number of 1s per row. Finally, in Section 3.3.2 we combine the algorithms of Sections 3.3.1 and 3.3.2.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Input: An $m \times n$ matrix M, an $m' \times n'$ matrix M' without duplicate columns.
Output: A submatrix of M that is isomorphic to M'.

1: **for each** m'-tuple of rows from M: {
2: **for each** column c' of M': {
3: check whether c' appears at least once in the submatrix of M that is induced by the rows selected in line 1; }
4: **if** for every column c' of M' the check was successful: {
5: **return** the corresponding rows and columns of M; }}
6: **return** $\emptyset$;

Figure 3.4: Algorithm for finding a given submatrix. The algorithm could easily be modified such that it also works for submatrices M' containing duplicate columns: in the case of a successful check in line 3, memorize which column in the submatrix of M was identical to c'; ignore this column during the subsequent checks.

3.3.1 Finding Forbidden Submatrices with Brute Force

By using a straightforward brute-force approach, one can find a size-$m' \times n'$ submatrix M' in an $m \times n$ matrix M in $O(m^{m'} \cdot n^{n'} \cdot m'n')$ time—just try every possibility of selecting an m'-tuple of rows and an n'-tuple of columns from M and compare the entries in the selected rows and columns with the entries in the desired submatrix. However, one can do faster, and we will demonstrate two ways how to do so.

First, instead of trying every combination of m' rows and n' columns of M, it suffices to try every combination of m' rows *or* n' columns: Assume without loss of generality that $m' \leq n'$. Then try every possibility of selecting an m'-tuple of rows from M. For each of the selected tuples, one can check in $O(nm'n')$ time whether every column of M' appears at least once in the submatrix of M that is induced by the selected rows: For every column of M', consider every column of the submatrix induced by the selected rows and check if the entries in this column are identical to the entries in the current column of M'. The pseudocode for finding a submatrix with this method is displayed in Figure 3.4; the proposition below follows.

Proposition 3.2. *Let M be a matrix of size $m \times n$ and M' be a matrix of size $m' \times n'$. Then the algorithm in Figure 3.4 returns in $O(m^{m'} \cdot nm'n')$ time a submatrix in M that is isomorphic to M'.*

Applying Proposition 3.2 to the matrix types M_{IV} and M_{V}, which are the matrix types from T that are, due to their small row number, most suitable for the algorithm of Figure 3.4, yields the following corollary.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Input: An $m \times n$ binary matrix M, an $m' \times n'$ binary matrix M' without duplicate rows.
Output: A submatrix of M that is isomorphic to M'.

1: $n'_1 :=$ maximum number of 1s occurring in a row of M';
2: $n'_0 := n' - n'_1$;
3: **for** every row r of M: {
4: **for each** n'_1-tuple of columns that have a 1 in row r and **each** n'_0-tuple of columns that have a 0 in row r: {
5: **for each** row r' of M': {
6: check whether r' appears at least once in the submatrix of M that is induced by the columns selected in line 4; }
7: **if** for every row r' of M' the check was successful: {
8: **return** the corresponding rows and columns of M; }}}
9: **return** $\emptyset$;

Figure 3.5: Algorithm for finding a given submatrix. The algorithm could easily be modified such that it also works for submatrices M' containing duplicate rows: in the case of a successful check in line 6, memorize which row in the submatrix of M was identical to r'; ignore this row during the subsequent checks.

Corollary 3.2. *Let M be a binary matrix of size $m \times n$. A submatrix in M that is isomorphic to M_{IV} (or to M_{V}) can be found in $O(m^4 n)$ time.*

Our second brute-force approach is suitable if M' contains a row with only few 0s, whereas M contains only a few number of 1s in every row, that is, M is a $(*, \Delta)$-matrix with small Δ. In such a case, let $r'_{\max}$ be the row of M' that contains the maximum number of 1s, let n'_1 be the number of 1s in $r'_{\max}$, and let n'_0 be the number of 0s in $r'_{\max}$. For example, if $M' = M_{\mathrm{III}_k}$, then the row $r'_{\max}$ is the $(k+2)$nd row, and we have $n'_1 = k+1$ and $n'_0 = 2$. Since there are at most Δ 1s in every row of a $(*, \Delta)$-matrix M, the number of the possibilities to select n'_1 columns from M that all contain a 1 in a specific row is bounded by $O(\Delta^{n'_1})$. Therefore, the idea for searching a forbidden submatrix in M is to iterate over all rows r_i of M and test whether M contains M' as a submatrix in such a way that r_i forms the row $r'_{\max}$ of M'; this test can be performed by considering every n'_1-tuple of columns from M having a 1 in row r_i in combination with every n'_0-tuple of columns from M having a 0 in row r_i. For each of these combinations, check in $O(m' \cdot n' \cdot m)$ time whether every row of the matrix M' appears at least once in the submatrix induced by the selected columns. The pseudocode of this algorithm is displayed in Figure 3.5, and we get the following proposition.

Proposition 3.3. *Let M be a $(*, \Delta)$-matrix of size $m \times n$ and M' be a binary matrix of size $m' \times n'$ that contains a row where the number of 1s is n'_1 and the number of 0s is n'_0. Then the algorithm in Figure 3.5 returns in $O(\Delta^{n'_1} \cdot n^{n'_0} \cdot$*

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

$m^2m'n'$) time a submatrix in M that is isomorphic to M'.

Applying Proposition 3.3 to those submatrices from T that are most suitable for the algorithm of Figure 3.5 yields the following corollary.

Corollary 3.3. *Let M be a binary $(*, \Delta)$-matrix of size $m \times n$ and M' be a binary matrix from T. A submatrix in M that is isomorphic to M' can be found in*

$$\begin{array}{ll} O(\Delta^{\Delta+2} \cdot m^2 n) \text{ time} & \text{if } M' = M_{\mathrm{II}_k},\ 1 \leq k \leq \Delta - 2, \\ O(\Delta^{\Delta+2} \cdot m^2 n^2) \text{ time} & \text{if } M' = M_{\mathrm{III}_k},\ 1 \leq k \leq \Delta - 1, \\ O(\Delta^3 m^2 n^3) \text{ time} & \text{if } M' = M_{\mathrm{IV}}, \text{ and} \\ O(\Delta^4 m^2 n) \text{ time} & \text{if } M' = M_{\mathrm{V}}. \end{array}$$

3.3.2 Finding M_{I_k}, M_{II_k}, or M_{III_k} via Induced Paths and Holes

In Section 3.3.1, we have considered exponential-time algorithms that can find any given submatrix M' within a given matrix M. In contrast, we will now consider the problem where we have given a matrix M and one of the matrix types M_{I_k}, M_{II_k}, and M_{III_k}, and the task is to find a minimum-size submatrix of this type in M. We present polynomial-time algorithms for each of these three matrix types. All three algorithms are based on the observation that the representing graphs of the matrices M_{I_k}, M_{II_k}, and M_{III_k} contain, after certain modifications, either a hole or an induced path, and that the rows and columns in M corresponding to this hole or path induce an M_{I_k}, M_{II_k}, or M_{III_k}, respectively. Since finding a minimum-size hole or induced path can be done in polynomial time, we can use this observation to obtain polynomial-time algorithms for finding minimum-size submatrices of the types M_{I_k}, M_{II_k}, and M_{III_k}.

For finding an induced M_{I_k}, we use the following observation (see Figure 3.3).

Observation 3.1. *The representing graph of an M_{I_k} is a chordless cycle of length $2k + 4$.*

Observation 3.1 immediately implies that finding a minimum-size submatrix of the type M_{I_k} reduces to finding a minimum-length hole in a bipartite graph. The latter task can be solved in polynomial time, as demonstrated by the pseudocode in Figure 3.6. The idea of the algorithm shown in Figure 3.6 is to try all 4-tuples $(r_{i_1}, c_{j_1}, r_{i_2}, c_{j_2})$ of vertices (line 2) and search for the shortest hole on which these four vertices appear consecutively (lines 3–8). To find such a hole, the two vertices c_{j_1} and r_{i_2} are deleted together with their neighbors except for r_{i_1} and c_{j_2} (line 4), and in the remaining graph a shortest path from c_{j_2} to r_{i_1} is sought (line 6). Since a shortest path in an unweighted graph $G = (V, E)$ can be found in $O(|V| + |E|)$ time with a breadth first search [CLRS01], we get the following result.

Proposition 3.4. *Let M be a $(*, \Delta)$-matrix M of size $m \times n$. Then a minimum-size submatrix of the type M_{I_k} in M can be found in $O(\Delta^3 m^3 + \Delta^2 m^2 n)$ time.*

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Input: A bipartite graph $G = (R, C, E)$.
Output: A subset $H \subseteq R \cup C$ inducing a minimum-length hole in G.

1: $H := \emptyset$;
2: **for each** 4-tuple $(r_{i_1}, c_{j_1}, r_{i_2}, c_{j_2})$ with $r_{i_1} \in R$, $c_{j_1} \in N(r_{i_1})$, $r_{i_2} \in N(c_{j_1})$, $c_{j_2} \in N(r_{i_2})$: {
3: **if** $r_{i_1}, c_{j_1}, r_{i_2}, c_{j_2}$ is an induced P_4 in G: {
4: $G' := G[(R \cup C) \setminus ((N[c_{j_1}] \cup N[r_{i_2}]) \setminus \{r_{i_1}, c_{j_2}\})]$;
5: **if** c_{j_2} and r_{i_1} are connected by a path in G': {
6: $P :=$ the vertices of the shortest path in G' from c_{j_2} to r_{i_1};
7: **if** $|\{r_{i_1}, c_{j_1}, r_{i_2}, c_{j_2}\} \cup P| < |H|$: {
8: $H := \{r_{i_1}, c_{j_1}, r_{i_2}, c_{j_2}\} \cup P$; }}}}
9: **return** H;

Figure 3.6: Algorithm for finding a minimum-length hole. The algorithm can easily be modified such that, if a vertex $r \in R$ is given in the input, it finds a minimum-size hole containing r: just set $r_{i_1} := r$ in line 2 and iterate over all 3-tuples $(c_{j_1}, r_{i_2}, c_{j_2})$ instead of all 4-tuples $(r_{i_1}, c_{j_1}, r_{i_2}, c_{j_2})$. This modification is useful for finding submatrices of the type M_{III_k}, whereas we use the algorithm as shown in the figure for finding submatrices of the type M_{I_k}.

Finding an induced M_{II_k} of minimum size is more complicated; however, we can use a similar approach as for searching submatrices of the type M_{I_k}. The main observation in this direction is that the upper left part of an M_{II_k} is identical to the upper part of an M_{I_k}, that is, the submatrix of an M_{II_k} that is induced by the first $k+1$ rows and $k+2$ columns is identical to the submatrix of an M_{I_k} that is induced by the first $k+1$ rows and all $k+2$ columns (the difference between an M_{II_k} and an M_{I_k} lies in the rightmost column and the two bottommost rows of the M_{II_k}, see Figure 3.3). We can formulate this finding more precisely as follows.

Observation 3.2. *Let M be the $(k+3) \times (k+3)$ matrix that results from complementing the $(k+2)$-nd and $(k+3)$-rd rows of an M_{II_k}. Then the representing graph of M consists of an isolated vertex, corresponding to the $(k+3)$-rd column of M, and an induced path with $2k+5$ vertices whose first vertex corresponds to the $(k+2)$-nd row of M and whose last vertex corresponds to the $(k+3)$-rd row of M.*

As a consequence of Observation 3.2, we get the following lemma.

Lemma 3.1. *Let M be a binary matrix and k be a positive integer. Then the following two statements are equivalent:*

1. *The matrix M contains an M_{II_k} as a submatrix.*
2. *There exist a column c_j and two rows r_{i_1} and r_{i_2} in M with the following properties:*

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

- *The rows r_{i_1} and r_{i_2} have a 1 in column c_j.*
- *If $\tilde{M}$ is the matrix consisting of*
  - *the row that results from complementing r_{i_1},*
  - *the row that results from complementing r_{i_2}, and*
  - *all rows of M that have a 0 in column c_j,*

  then the representing graph of $\tilde{M}$ contains an induced path P with $2k+5$ vertices whose first vertex corresponds to the complemented row r_{i_1} and whose last vertex corresponds to the complemented row r_{i_2}.

Moreover, statement 2 implies the following:

3. *The column c_j and the rows and columns corresponding to the vertices of the path P together induce an M_{II_k} in M.*

Proof. **1 ⇒ 2:** Let c_j be the column of M that contains the $(k+3)$-rd column of the M_{II_k} submatrix, and let r_{i_1} and r_{i_2} be the two rows of M that contain the $(k+2)$-nd and $(k+3)$-rd rows of the M_{II_k} submatrix. Then the claim follows from Observation 3.2.

2 ⇒ 1 ∧ 3: This claim follows from the fact that the vertex corresponding to column c_j cannot be part of P, because in $\tilde{M}$ (after complementing rows r_{i_1} and r_{i_2}) column c_j contains only 0s. Therefore, the submatrix of M that is induced by the column c_j and the rows and columns corresponding to the vertices of the path P induce an M_{II_k}. □

Lemma 3.1 indicates how to find an induced M_{II_k} of minimum size: Try all combinations of two rows r_{i_1}, r_{i_2} and one column c_j from M such that r_{i_1} and r_{i_2} contain a 1 in column c_j. For each of these combinations, complement the rows r_{i_1} and r_{i_2}, take all rows having a 0 in c_j, and search in the representing graph of the resulting matrix for the shortest induced path that has the properties mentioned in part 2 of Lemma 3.1—in particular, since $k \geq 1$, this path must have length $2k+4 \geq 6$. Each representing graph to be considered has at most $m+n$ vertices and less than $\Delta m + 2n$ edges. Figure 3.7 shows the pseudocode of this approach, and Figure 3.8 shows how to find a shortest induced path with length at least six from a given vertex r_1 to a vertex r_2. The approach of the algorithm displayed in Figure 3.8 is the same as in the algorithm of Figure 3.6 for finding a hole: By trying all possibilities, we select the vertices $(c_{j_1}, r_{i_1}, c_{j_2})$ that shall be located next to r_1 on the desired path, and for each of these possibilities, we search for the shortest path from c_{j_2} to r_2 containing no neighbor (except for c_{j_2}) of the vertices r_1, c_{j_1}, r_{i_1} (lines 4–6). Again we use a breadth first search [CLRS01] for finding shortest paths (line 6), which leads to the following result.

Proposition 3.5. *Let M be a $(*, \Delta)$-matrix M of size $m \times n$. Then a minimum-size submatrix of the type M_{II_k} in M can be found in $O(\Delta^3 m^4 n + \Delta^2 m^3 n^2)$ time.*

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Input: A binary matrix M.
Output: A minimum-size submatrix of the type M_{II_k} occurring in M.

1: $M' := \emptyset$;
2: **for each** pair r_{i_1}, r_{i_2} of rows of M: {
3: **for each** column c having a 1 in rows r_{i_1} and r_{i_2}: {
4: $R_0 :=$ the set of rows having a 0 in column c;
5: $\overline{r_{i_1}} :=$ r_{i_1} complemented; $\overline{r_{i_2}} :=$ r_{i_2} complemented;
6: $\tilde{M} :=$ the matrix consisting of $\overline{r_{i_1}}$, $\overline{r_{i_2}}$, and all rows from R_0;
7: $\tilde{G} :=$ the representing graph of $\tilde{M}$;
8: search for a path H in $\tilde{G}$ that has a minimum length under the property that it contains at least seven vertices and its endpoints are the vertices corresponding to $\overline{r_{i_1}}$ and $\overline{r_{i_2}}$;
9: **if** H exists and $|V(H)| + 1 <$ number of rows and columns in M': {
10: $M' :=$ the submatrix of M that is induced by the column c and the rows and columns corresponding to the vertices of H; }}}
11: **return** M';

Figure 3.7: Algorithm for finding a minimum-size submatrix of the type M_{II_k}.

Finding an induced M_{III_k} of minimum size is very similar to finding an induced M_{II_k}. We start with the following observation (see Figure 3.3).

Observation 3.3. *Let M be the $(k+2) \times (k+3)$ matrix that results from complementing the $(k+2)$-nd row of an M_{III_k}. Then the representing graph of M consists of an isolated vertex, corresponding to the $(k+3)$-rd column of M, and a chordless cycle.*

The following lemma can be obtained from Observation 3.3 in complete analogy to the way how Lemma 3.1 was obtained from Observation 3.2.

Lemma 3.2. *Let M be a binary matrix and k be a positive integer. Then the following two statements are equivalent:*

1. *The matrix M contains an M_{III_k} as a submatrix.*
2. *There exist a column c_j and a row r_i in M with the following properties:*
   - *The row r_i has a 1 in column c_j.*
   - *If $\tilde{M}$ is the matrix consisting of*
     - *the row that results from complementing r_i and*
     - *all rows of M that have a 0 in column c_j,*

     then the representing graph of $\tilde{M}$ contains a chordless cycle H of length $2k+4$ that contains the vertex corresponding to the complemented row r_i.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Input: A bipartite graph $G = (R, C, E)$ and two vertices $r_1, r_2 \in R$.
Output: A subset $H \subseteq R \cup C$ inducing a path from r_1 to r_2 in G whose length is minimum under the property that it contains at least seven vertices.

1: $H := \emptyset$;
2: **for each** 3-tuple $(c_{j_1}, r_{i_1}, c_{j_2})$ with $c_{j_1} \in N(r_1)$, $r_{i_1} \in N(c_{j_1})$, $c_{j_2} \in N(r_{i_1})$: {
3: **if** $r_1, c_{j_1}, r_{i_1}, c_{j_2}$ is an induced P_4 in G and none of the vertices c_{j_1}, c_{j_2} belongs to $N(r_2)$: {
4: $G' := G[(R \cup C) \setminus ((N[r_1] \cup N[c_{j_1}] \cup N[r_{i_1}]) \setminus \{c_{j_2}\})]$;
5: **if** c_{j_2} and r_2 are connected by a path in G': {
6: $P :=$ the vertices of the shortest path in G' from c_{j_2} to r_2;
7: **if** $|\{r_1, c_{j_1}, r_{i_1}, c_{j_2}, r_2\} \cup P| < |H|$: {
8: $H := \{r_1, c_{j_1}, r_{i_1}, c_{j_2}, r_2\} \cup P$; }}}}
9: **return** H;

Figure 3.8: Algorithm for finding a minimum-length induced path with at least seven vertices.

Moreover, statement 2 implies the following:

3. *The column c_j and the rows and columns corresponding to the vertices of the hole H together induce an M_{III_k} in M.*

According to Lemma 3.2, an induced M_{III_k} of minimum size can be found very similarly to the method presented above for finding an induced M_{II_k}; the only differences are that one only has to complement one row instead of two and that one has to search for a hole containing a given vertex instead of a path containing two given vertices—such a hole can be found as described in Figure 3.6. This leads to the algorithm in Figure 3.9, which outputs an induced M_{III_k} of minimum size, and to the following proposition.

Proposition 3.6. *Let M be a $(*, \Delta)$-matrix M of size $m \times n$. Then a minimum-size submatrix of the type M_{III_k} in M can be found in $O(\Delta^3 m^3 n + \Delta^2 m^2 n^2)$ time.*

3.3.3 Combining the Algorithms

The algorithms in Section 3.3.1 lead to fast running times when searching a submatrix of one of the types M_{III_k}, M_{IV} and M_{V}, whereas every algorithm in Section 3.3.2 efficiently finds a submatrix of one of the types M_{I_k}, M_{II_k}, and M_{III_k}. Now, we combine the algorithms from Sections 3.2, 3.3.1, and 3.3.2 to find a submatrix that is isomorphic to *any* of the submatrices from T and has a minimum number of rows, columns, rows and columns, or entries.

Theorem 3.1. *Let M be a $(*, \Delta)$-matrix of size $m \times n$. A forbidden submatrix from T (see Figure 2.8) in M that has a minimum number of rows can be found in*

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Input: A binary matrix M.
Output: A minimum-size submatrix of the type M_{III_k} occurring in M.

1: $M' := \emptyset$;
2: **for each** row r_i of M: {
3: **for each** column c_j of M having a 1 in row r_i: {
4: $R_0 :=$ the set of rows having a 0 in column c_j;
5: $\overline{r_i} :=$ r_i complemented;
6: $\tilde{M} :=$ the matrix consisting of $\overline{r_i}$ and all rows from R_0;
7: $\tilde{G} :=$ the representing graph of $\tilde{M}$;
8: search for a minimum-length hole in $\tilde{G}$ that contains the vertex corresponding to $\overline{r_i}$;
9: **if** H exists and $|V(H)| + 1 <$ number of rows and columns in M': {
10: $M' :=$ the submatrix of M that is induced by the column c_j and the rows and columns corresponding to the vertices of H; }}}
11: **return** M';

Figure 3.9: Algorithm for finding a minimum-size submatrix of the type M_{III_k}.

$O(\Delta^3 m^2 n \cdot (m + n^2))$ time. Within the same time, one can also find a forbidden submatrix from T in M that has a minimum number of columns, a minimum number of rows and columns, or a minimum number of entries.

Proof. The claimed running time can be obtained as follows: First, run the algorithm from Figure 3.1 (Proposition 3.1), which finds a forbidden submatrix of "almost minimum" size, and let A be the returned submatrix. Second, run the algorithm from Figure 3.9 (Proposition 3.6) to find an induced M_{III_k}, and let B be the submatrix found here. Third, run the algorithm shown in Figure 3.5 (Corollary 3.3) two times, once for finding an induced M_{IV} and once for finding an induced M_{V}, and let C and D, respectively, be the found submatrices. Return the matrix with the minimum number of rows (columns, rows and columns, entries) out of A, B, C, and D.

The correctness of this approach is obvious: As shown in the proof of Proposition 3.1, if the forbidden submatrix from T in M with the minimum number of rows (columns, rows and columns, entries) is of the type M_{I_k} or M_{II_k}, then M does not contain a submatrix with less rows (columns, rows and columns, entries) than A. In all other cases, the forbidden submatrix from T in M with the minimum number of rows (columns, rows and columns, entries) must be one of B, C, and D. □

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

3.4 Conclusion

We have presented different algorithms for finding a submatrix that belongs to the set of forbidden submatrices given by Tucker (see Theorem 2.5 and Figure 2.8) and has a minimum number of rows, columns, rows and columns, or entries. Our main result is an algorithm finding such a submatrix in polynomial time. However, an algorithm with a faster running time would be desirable; in particular, it remains open whether there is an algorithm for this task that runs in linear time.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Chapter 4

How to Obtain the C1P: The C1P Submatrix Problem

In this chapter, we develop an algorithmically useful refinement of the forbidden submatrix characterization (Theorem 2.5) of Tucker [Tuc72] for 0/1-matrices with the C1P. As a main result, we obtain, based on this characterization, new polynomial-time approximation algorithms and fixed-parameter tractability results for the NP-hard problems Min-COS-R and Min-COS-C. These problems ask for a minimum number of row or column deletions to transform a given matrix into a matrix having the C1P. Moreover, we consider the maximization versions Max-COS-R and Max-COS-C of the problems (where the task is to find a submatrix that has the C1P and consists of a maximum number of rows or columns, respectively). We complement already known approximation results for Max-COS-C by presenting fixed-parameter algorithms for Max-COS-C and approximation and fixed-parameter results for Max-COS-R.

4.1 Introduction and Overview

The C1P being a desirable property that often leads to efficient algorithms, the natural problem arises what to do if a given matrix does not have the C1P. As a consequence, there has been recently increased interest in matrix modification problems that deal with the transformation of a given 0/1-matrix into a 0/1-matrix fulfilling the C1P [HG02, TZ07]. Applications for problems of this kind can also be found in computational biology [ABH98, AM96, GGKS95, LH03, WR00], see Section 1.1. The following two minimization problems show up naturally in this context:

Min-COS-C: Given a binary matrix M, find a minimum-cardinality set of *columns* to delete such that the resulting matrix has the C1P.

Min-COS-R: Given a binary matrix M, find a minimum-cardinality set of *rows* to delete such that the resulting matrix has the C1P.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

These problems can also be posed as decision problems—in that case, the input does not only consist of a matrix M, but it contains also a positive integer d and the question is whether M can be transformed into a matrix having the C1P by deleting at most d columns or rows, respectively. We will use the problem names Min-COS-C and Min-COS-R for the minimization problem as well as for the corresponding decision problem.

In addition to the two minimization problems defined above, we will also consider the maximization duals of Min-COS-C and Min-COS-R:

Max-COS-C: Given a binary matrix M, find a maximum-cardinality set of *columns* that induces a matrix having the C1P.

Max-COS-R: Given a binary matrix M, find a maximum-cardinality set of *rows* that induces a matrix having the C1P.

Again, we use the names Max-COS-C and Max-COS-R for both the optimization and the decision versions of the problems. In case of the decision versions, we ask whether there is a column set or a row set, respectively, that induces a matrix with the C1P and whose cardinality is greater than or equal to a given value d'. Note that we will always use d to denote the number of columns or rows to be deleted when considering Min-COS-C or Min-COS-R, and d' to denote the number of columns or rows to be selected when considering Max-COS-C or Max-COS-R.

Concerning the computational hardness of these problems in the "classical" complexity theory, it makes obviously no difference if one considers Min-COS-C (Min-COS-R) or Max-COS-C (Max-COS-R): the optimum solution for the maximization problem can be directly derived from the optimum solution for the minimization problem and vice versa. However, the approximability of the minimization and the maximization problems may differ, and also from the parameterized point of view it makes a difference if one considers the problems Min-COS-C and Min-COS-R, where the standard parameter would be d, or the problems Max-COS-C and Max-COS-R with the standard parameter d'.

Whereas previous work [Haj00, HG02, TZ07] focussed on the maximization versions, here we mainly concentrate on the minimization versions of the problems. In particular, from the parameterized point of view and when expecting that large submatrices with the C1P exist, this appears to be the more natural optimization criterion. Unfortunately, even for sparse matrices with few 1-entries, as they occur in many applications [AM96, DER89, KS96, TZ07, WR00], both the maximization and minimization problems quickly become NP-hard [HG02, TZ07]. In this chapter, we therefore explore the algorithmic complexity of these problems in a more fine-grained way, providing new algorithmic results. To this end, based on the forbidden submatrix characterization for the C1P (Theorem 2.5) due to Tucker [Tuc72], our main technical result is a structural theorem dealing with the selection of particularly useful forbidden submatrices.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Before we describe our results in more detail, we have to introduce some notation. Whereas an $m \times n$ matrix is a matrix having m rows and n columns, the term (x, y)-*matrix* will be used to denote a matrix that has at most x 1s in any column and at most y 1s in any row. (This notation was used in previous work [HG02, TZ07].) With $x = *$ or $y = *$, we indicate that there is no upper bound on the number of 1s in columns or in rows, respectively.

The NP-hardness of Max-COS-C was already mentioned by Garey and Johnson [GJ79]. However, Hajiaghayi and Ganjali [Haj00, HG02] observed that in Garey and Johnson's monograph [GJ79] the reference for the NP-hardness proof of Max-COS-C is not correct—indeed, the referenced proof shows the NP-hardness of Max-COS-R on $(3, 2)$-matrices. Then, Max-COS-C has been shown NP-hard for $(2, 4)$-matrices by Hajiaghayi and Ganjali [HG02], and for $(2, 3)$- and $(3, 2)$-matrices by Tan and Zhang [TZ07]. Moreover, Tan and Zhang [TZ07] proved, independently from the conference publication [DGN07] of this chapter, that there exists no polynomial-time constant-factor approximation algorithm for Max-COS-C on $(*, 2)$-matrices unless P = NP [TZ07]; their reduction can also be used to show that Max-COS-C on $(*, 2)$-matrices is W[1]-hard with respect to d'. On the positive side, Tan and Zhang [TZ07] provided polynomial-time approximability results for the sparsest NP-hard cases of Max-COS-C, that is, for $(2, 3)$- and $(3, 2)$-matrices: Restricted to $(3, 2)$-matrices, Max-COS-C can be approximated within a factor of 0.5; for $(2, *)$-matrices, it is approximable within a factor of 0.5; for $(2, 3)$-matrices, the approximation factor is 0.8. Concerning the minimization versions of the problems, we are only aware of positive results for the graph problems 2-Layer Planarization and Linear Arrangement by Deleting Edges, which are equivalent to Min-COS-C on $(2, *)$-matrices without identical columns and to Min-COS-R on $(*, 2)$-matrices without identical rows, respectively. In particular, Suderman and Whitesides [SW05], Fernau [Fer05a, Fer05b] and Suderman [Sud05] gave fixed-parameter search-tree algorithms, and Dujmovic at al. [DFH$^+$06] gave a linear[1] problem kernel for 2-Layer Planarization; moreover, Fernau [Fer05a, Fer08] presented a fixed-parameter algorithm running in $O^*(2.4676^d)$ time and a problem kernel consisting of $6d$ vertices and $6d$ edges for Linear Arrangement by Deleting Edges.

Besides the above mentioned structural theorem, we show the following main results.

1. For any constant $\Delta \geq 2$, Min-COS-C on $(*, \Delta)$-matrices is polynomial-time approximable with a factor of 6 if $\Delta = 3$ and with a factor of $(\Delta + 2)$ if $\Delta \neq 3$, and Min-COS-R on $(*, \Delta)$-matrices is polynomial-time approximable with a factor of $(\Delta + 1)$. In particular, this implies a polynomial-time factor-4 approximation algorithm for Min-COS-C on $(*, 2)$-matrices.

[1] In this context, "linear" means that the number of vertices *and* the number of edges in the problem kernel is linear in the parameter.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Factor 4 seems to be the best one can currently hope for because a factor-δ approximation for Min-COS-C restricted to $(*, 2)$-matrices implies a factor-$\delta/2$ approximation for Vertex Cover. It is commonly conjectured that Vertex Cover is not polynomial-time approximable within a factor of $2-\epsilon$, for any constant $\epsilon > 0$, unless P = NP [KR08]. Moreover, on $(*, \Delta)$-matrices with $\Delta \geq 2$, Min-COS-C and Min-COS-R are fixed-parameter tractable with respect to the combined parameters Δ, d.

2. On $(*, 2)$-matrices, Min-COS-C admits a problem kernel consisting of $O(d^2)$ rows and columns and Min-COS-R admits a problem kernel consisting of $O(d^2)$ rows and $O(d)$ columns.

3. For Min-COS-C and Min-COS-R on $(2, *)$-matrices, we give polynomial-time approximation algorithms achieving approximation factors of 6 and 4, respectively. Moreover, Min-COS-C and Min-COS-R can be solved in $O(6^d \cdot \min\{m^4n, m^2n^3\})$ and $O(4^d \cdot \min\{m^4n, m^2n^3\})$ time, respectively. These results follow directly from Theorems 2.3 and 2.4 due to Tucker [Tuc72].

4. There exists no polynomial-time constant-factor approximation algorithm for Max-COS-C on $(*, 2)$-matrices and Max-COS-R on $(2, *)$-matrices unless P = NP. Moreover, both problems are W[1]-hard, that is, presumably fixed-parameter intractable, with respect to the desired number d' of columns or rows of the submatrix to be found. (A similar reduction showing the hardness of Max-COS-C on $(*, 2)$-matrices was independently given by Tan and Zhang [TZ07].)

5. Max-COS-C on $(2, *)$-matrices can be solved in $2^{O(d')} \cdot |M|^{O(1)}$ time (a factor-0.5 polynomial-time approximation algorithm for this problem was already known [TZ07]). Max-COS-R on $(*, 2)$-matrices can be solved in $2^{O(d')} \cdot |M|^{O(1)}$ time and approximated with a factor of 0.75 in polynomial time. Both problem variants are, hence, fixed-parameter tractable with respect to the parameter d'.

We summarize known and new results for Max-COS-C, Min-COS-C, Max-COS-R, and Min-COS-R in Table 4.1. Moreover, Table 4.2 shows in more detail our results for Min-COS-C and Min-COS-R on $(*, \Delta)$-matrices.

The remainder of this chapter is structured as follows. After some basic facts and definitions, we present in Section 4.3 the hardness results for Max-COS-C and Max-COS-R, followed by approximation and fixed-parameter algorithms for these two problems in Section 4.4. Sections 4.5 and 4.6 deal with the minimization problems Min-COS-C and Min-COS-R: in Section 4.5 we consider these problems on $(*, \Delta)$-matrices, in Section 4.6 we allow only input matrices that have at most two ones per row or per column. Section 4.7 contains the proof for our main structural theorem. Section 4.8 concludes the chapter with some problems that remained open.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Table 4.1: Summary of known and new results for Max-Cos-C, Min-Cos-C, Max-Cos-R and Min-Cos-R. The table shows the factors of the approximation algorithms and the running times of the fixed-parameter algorithms. The type (x, y) of the input matrix describes the maximum number of 1s per row and column: An (x, y)-matrix has at most x 1s per column and at most y 1s per row. With $x = *$ or $y = *$, we indicate that there is no upper bound on the number of 1s in columns or in rows, respectively; Δ stands for an any number between 1 and n. We only emphasize the exponential parts of the running times, that is, the shown running times have to be multiplied with polynomials with respect to the input size. An empty field means that we are not aware of any results concerning the corresponding problem variant.

Type	Max-Cos-C	Min-Cos-C	Max-Cos-R	Min-Cos-R
$(3, 2)$	• 0.5-approx.[2]	Pos. results: see $(*, 2)$	Pos. results: see $(*, 2)$	Pos. results: see $(*, 2)$
$(*, 2)$	• No const. approx.[3] • W[1]-hard[3,4]	• No 2.72-approx. • Poly. kernel[5] More pos. results: $(*, \Delta)$	• 0.75-approx. • $2^{O(d')}$-alg	• Poly. kernel[5,6] More pos. results: $(*, \Delta)$
$(*, \Delta)$	Neg. results: see $(*, 2)$	• $(\Delta + 2)$-approx.[7] • $(\Delta + 2)^d \cdot \Delta^{O(\Delta)}$-alg.[7] Neg. results: see $(*, 2)$		• $(\Delta + 1)$-approx. • $(\Delta + 1)^d \cdot (2\Delta)^{2d}$-alg.
$(2, 3)$	• 0.8-approx.[2] More pos. results: $(2, *)$	Pos. results: see $(2, *)$		Pos. results: see $(2, *)$
$(2, *)$	• 0.5-approx.[2] • $2^{O(d')}$-alg.	• 6-approx. • 6^d-alg.[8]	• No const. approx. • W[1]-hard[4]	• No 2.72-approx. • 4-approx. • 4^d-alg.
$(\Delta, *)$			Neg. results: see $(2, *)$	Neg. results: see $(2, *)$

[2]This result is due to Tan and Zhang [TZ07].

[3]The hardness of approximating Max-Cos-C was shown independently by Tan and Zhang [TZ07]; their reduction allows also to show the W[1]-hardness.

[4]W[1]-hardness is with respect to the parameter d'.

[5]The polynomial problem kernel is with respect to the parameter d.

[6]More results are known for the case where the $(*, 2)$-matrix does not have duplicate rows: the problem is then equivalent to Linear Arrangement By Deleting Edges, for which fixed-parameter algorithms and smaller problem kernels exist [Fer05a, Fer08].

[7]For the ease of presentation, at this point the table ignores the case $\Delta = 3$. Indeed, if $\Delta = 3$, then the factor of the approximation algorithm for Min-Cos-C is 6, and the running time of the fixed-parameter algorithm is $O^*(6^d \cdot \Delta^{O(\Delta)})$.

[8]More results are known for the case where the $(2, *)$-matrix does not have duplicate columns: the problem is then equivalent to 2-Layer Planarization, for which faster running times [Fer05a, Fer05b, Sud05, SW05] and problem kernels [DFH+06] are known.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Table 4.2: Algorithms for MIN-COS-C and MIN-COS-R on $(*, \Delta)$-matrices.

	Approximation algorithms		**Fixed-parameter algorithms**
MIN-COS-C	Factor	Running time	Running time
$\Delta = 2, 4, 5, \ldots$	Δ+2	$O^*(1)$	$O^*((\Delta+2)^d \cdot (3\Delta)^{\min\{d,\Delta\}})$
$\Delta = 3$	6	$O^*(1)$	$O^*(6^d)$
MIN-COS-R	Factor	Running time	Running time
$\Delta \geq 2$	Δ+1	$O^*((2\Delta)^{8\Delta^2})$	$O^*((\Delta+1)^d \cdot (2\Delta)^{2\min\{d,4\Delta^2\}})$

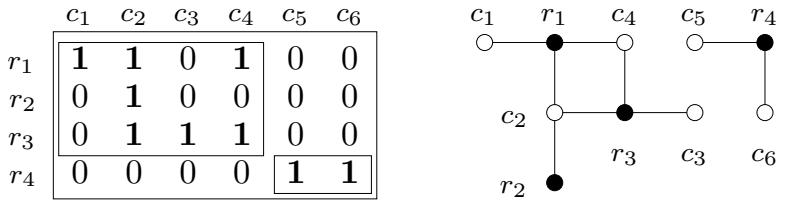


Figure 4.1: A matrix with two components and its representing graph.

4.2 Basics Facts and Definitions

We start with some definitions that are directly deduced from very similar terms in graph theory.

Let M be a binary matrix and G_M its representing graph. Two lines ℓ, ℓ' of M are *connected in* M if there is a path in G_M connecting the vertices corresponding to ℓ and ℓ'. A submatrix M' of M is called *connected* if each pair of lines belonging to M' is connected in M'. A maximal connected submatrix of M is called a *component* of M. A *shortest path* between two connected submatrices M_1, M_2 of M is the shortest sequence $\ell_1, \ldots, \ell_p$ of lines such that $\ell_1 \in M_1$ and $\ell_p \in M_2$ and the vertices corresponding to $\ell_1, \ldots, \ell_p$ form a path in G_M. If such a shortest path exists, then $p - 1$ is called the *distance* between M_1 and M_2.

Note that each submatrix M' of M one-to-one corresponds to an induced subgraph of G_M and that each component of M one-to-one corresponds to a connected component of G_M. An illustration of the components of a matrix is shown in Figure 4.1. If the distance between two lines ℓ_1 and ℓ_p is a positive even number, then ℓ_1 and ℓ_p are either both rows or both columns; if the distance is odd, then exactly one of ℓ_1 and ℓ_p is a row and one is a column.

Observation 4.1. *Let M be a matrix and let ℓ be a line of M. Then ℓ belongs to exactly one component M' of M, and M' contains all 1-entries of ℓ.*

The following corollary is a direct consequence of Observation 4.1.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

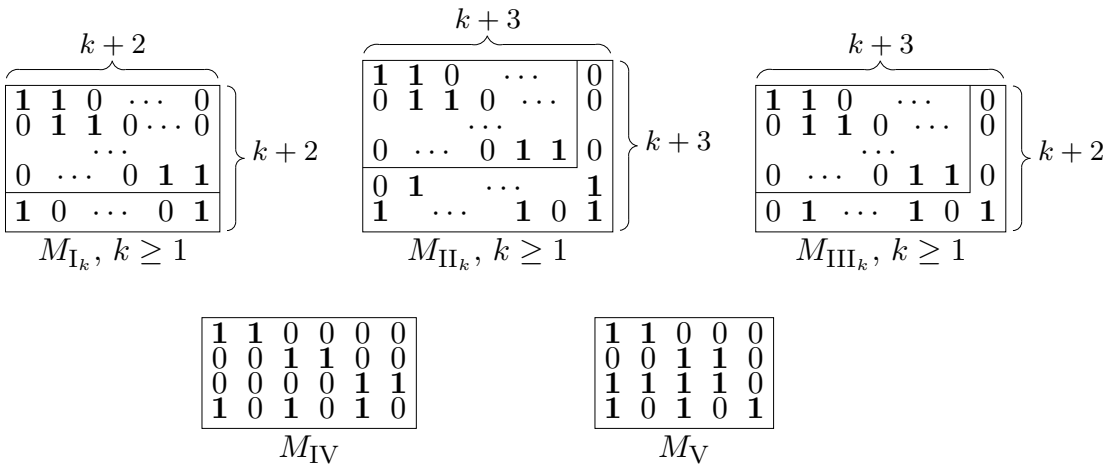


Figure 4.2: The set T of forbidden submatrices for the C1P due to Tucker [Tuc72], given in Theorem 2.5.

Corollary 4.1. *Let M be a matrix and let $M_1, \ldots, M_i$ be the components of M. If the column (or row) sets $F_1, \ldots, F_i$ are optimal solutions for* Min-Cos-C *(or* Min-Cos-R*) on $M_1, \ldots, M_i$, respectively, then $F_1 \cup \ldots \cup F_i$ is an optimal solution for* Min-Cos-C *(or* Min-Cos-R*) on M.*

We will make extensive use of Theorem 2.5, which says that a matrix M has the C1P iff it contains none of the matrices M_{I_k}, M_{II_k}, M_{III_k} (with $k \geq 1$), M_{IV}, and M_{V} as a submatrix. These matrices have been introduced in Chapter 2, see Figure 4.2 for an illustration. Like in the previous chapters, we denote this set of forbidden submatrices with T.

When the input matrix for Min-Cos-C, Min-Cos-R, Max-Cos-C, or Max-Cos-R has exactly two ones per row or two ones per column, then this matrix can be interpreted as a vertex-edge incidence matrix or an edge-vertex incidence matrix (see Section 2.2), respectively, and the matrix problem transfers to a graph problem in a very natural way. In the case of Max-Cos-C, for example, we may assume without loss of generality that the input matrix M has no two identical rows and that no row of M contains only one 1. (If a row appears more than once, we can delete all occurrences except for one, and if a row contains only one 1, we can delete this row; both operations clearly do not change the size of an optimal solution for Max-Cos-C.) Therefore, we interpret M, where each row contains exactly two 1s, as a graph $G = (V, E)$ with V corresponding to the set of columns and E corresponding to the set of rows of M. It is easy to verify that M has the C1P iff G is a union of vertex-disjoint paths—a fact that was already described in Theorem 2.2. Hence, as observed by Tan and Zhang [TZ07], Max-Cos-C with each row containing at most two 1s is equivalent to the problem Max-VIDPS defined as follows.[9]

[9]With *equivalent*, we mean that there are two polynomial-time reductions, one from the

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Maximum Vertex-Induced Disjoint Paths Subgraph (Max-VIDPS)
Input: A graph $G = (V, E)$.
Task: Find a maximum-cardinality vertex subset $V' \subseteq V$ such that $G[V']$ is a union of vertex-disjoint paths.

Analogously, the problem Min-COS-C restricted to matrices with at most two 1s per row is equivalent to the problem Min-VIDPS, where, given a graph $G = (V, E)$, we ask for a minimum-cardinality set $V' \subseteq V$ whose removal transforms G into a union of vertex-disjoint paths.

Not only the column selection problem Max-COS-C on $(*, 2)$-matrices can be interpreted as a graph problem, but also the row selection problem Max-COS-R; in this case the task is to transform a given graph into a union of vertex-disjoint paths by *edge deletions*. However, whereas in the case of Max-COS-C duplicate rows in the input matrix can be ignored (that is, if a row appears several times, all occurrences except for one can be deleted), in the case of Max-COS-R the frequency of a row must be taken into account. Therefore, Max-COS-R turns into the edge-weighted graph problem defined as follows.

Maximum-Weight Edge-Induced Disjoint Paths Subgraph (Max-WEIDPS)
Input: A graph $G = (V, E)$ and a weight function $w : E \to \mathbb{N}$.
Task: Find a maximum-weight edge subset $E' \subseteq E$ such that the graph (V, E') is a union of vertex-disjoint paths.

Again, one can also consider the minimization problem instead of the maximization problem: Min-COS-R, restricted to matrices with at most two 1s per row, is equivalent to the problem Min-WEIDPS, where, given a graph $G = (V, E)$, we ask for a minimum-weight set $E' \subseteq E$ whose removal transforms G into a union of vertex-disjoint paths.

In analogy to the close relation between the problems Max-COS-C, Min-COS-C, Max-COS-R, and Min-COS-R on $(*, 2)$-matrices on the one hand and the graph problems Max-VIDPS, Min-VIDPS, Max-WEIDPS, and Min-WEIDPS on the other hand, Theorem 2.2 also indicates how to transform the matrix problems Max-COS-C and Min-COS-C into graph problems when the input matrix is not a $(*, 2)$-matrix, but a $(2, *)$-matrix. In this case, Max-COS-C translates into the following problem (see also [TZ07]).

Maximum-Weight Edge-Induced Disjoint Caterpillars Subgraph (Max-WEIDCatS)
Input: A graph $G = (V, E)$ and a weight function $w : E \to \mathbb{N}$.
Task: Find a maximum-weight edge subset $E' \subseteq E$ such that the graph (V, E') is a union of vertex-disjoint caterpillars.

matrix problem to the graph problem and one from the graph problem to the matrix problem, such that in both mappings columns and rows one-to-one correspond to vertices and edges.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Table 4.3: Equivalences between submatrix problems on matrices with only two 1s per row or per column and graph problems.

	Max-COS-C	Min-COS-C	Max-COS-R	Min-COS-R
$(*, 2)$	Max-VIDPS	Min-VIDPS	Max-WEIDPS	Min-WEIDPS
$(2, *)$	Max-WEIDCatS	Min-WEIDCatS	–	–

The minimization version of Max-WEIDCatS is called Min-WEIDCatS; here a minimum-weight set of edges shall be deleted to obtain a union of vertex-disjoint caterpillars.

At first sight, one could think that in analogy to the three graph problems introduced above the problem Min-COS-R on $(2, *)$-matrices is equivalent to the problem Minimum Vertex-Induced Disjoint Caterpillars Subgraph *(*Min-VIDCatS*)*, where, given a graph $G = (V, E)$, the task is to find a minimum-cardinality vertex subset $V' \subseteq V$ such that $G[V \setminus V']$ is a union of vertex-disjoint caterpillars. However, this is not the case, and also Max-COS-R on $(2, *)$-matrices is not equivalent to the maximization variant Max-VIDCatS of Min-VIDCatS. The reason is that with deleting a vertex v in a graph we also delete all edges incident to v. Deleting a row r in a $(2, *)$-matrix, however, does not delete all columns that have a 1 in row r, but just removes one 1 from all of these columns. Hence, in each of these columns one 1 remains, and the columns can still be part of an M_{IV}. (When considering Max-COS-C on $(*, 2)$-matrices, the analogous case where one deletes only one 1 from a row behaves differently because the resulting row with only one 1 cannot be part of any forbidden submatrix.)

Table 4.3 summarizes the connections between submatrix problems on matrices with only two 1s per row or per column and graph problems. We use these relationships for obtaining hardness proofs for Max-COS-C and Max-COS-R as well as for demonstrating problem kernels for Min-COS-C and Min-COS-R, because the graph problems allow for an easier understanding.

We end with a straightforward observation: With respect to (2,2)-matrices, the problems Min-COS-C and Min-COS-R are polynomial-time solvable. This holds also for the "editing problem" Min-CO-1E, where, given a binary matrix M, the task is to find a minimum-cardinality set of *1-entries* in M that shall be *flipped* (that is, replaced by 0-entries) such that the resulting matrix has the C1P. (The analogous problem where 0-entries have to be flipped is called Consecutive Ones Augmentation, see [GJ79, Vel85]; the problem where arbitrary entries can be flipped is considered by Oswald and Reinelt [OR03b].) The reason is that, as shown above, any $(*, 2)$-matrix can be interpreted as the vertex-edge or edge-vertex incidence matrix of a graph, and, hence, Min-COS-C, Min-COS-R, and Min-CO-1E on such matrices can be formulated as graph modification problems. These graph problems are polynomial-time solvable on in-

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

put graphs with maximum degree 2, which correspond to $(2, 2)$-matrices. Second, on $(*, 2)$-matrices the problems Min-COS-R and Min-CO-1E are equivalent, because deleting a row one-to-one corresponds to flipping a 1-entry since a row with only one 1-entry can be clearly omitted from further consideration.

4.3 Hardness Results

As observed by Tan and Zhang [TZ07], Max-COS-C with each row containing at most two 1s is equivalent to the problem Max-VIDPS (see Section 4.2), where, given an undirected graph $G = (V, E)$, we ask for a maximum-size set $V' \subseteq V$ of vertices such that the subgraph of G induced by V' is a set of vertex-disjoint paths.

In what follows, we first show that Max-VIDPS is hard to approximate; due to the equivalence between Max-VIDPS and Max-COS-C on $(*, 2)$-matrices, this result carries over to the latter problem. Then we show that the reduction also proves the hardness of approximating Max-VIDCatS and Max-COS-R on $(2, *)$-matrices and the W[1]-hardness of Max-VIDPS, Max-COS-C on $(*, 2)$-matrices, Max-VIDCatS, and Max-COS-R on $(2, *)$-matrices.

To prove the hardness of approximating Max-VIDPS, we give an approximation-preserving reduction from the NP-hard Independent Set problem to Max-VIDPS. (A different reduction from Independent Set to Max-VIDPS was independently achieved by Tan and Zhang [TZ07].) In Independent Set, one is given an undirected graph and asks for a maximum-cardinality set of vertices such that no two vertices in this set are adjacent—such a set of vertices is called an *independent set*.

Theorem 4.1. *There exists no polynomial-time factor-$O(|V|^{(1-\epsilon)})$ approximation algorithm, for any $\epsilon > 0$, for* Max-VIDPS *unless* NP = ZPP*, and no polynomial-time factor-$O(|V|^{(1/2-\epsilon)})$ approximation algorithm unless* NP = P*.*

Proof. We give a simple reduction from Independent Set to Max-VIDPS such that the Independent Set instance has a solution with k vertices iff the resulting Max-VIDPS instance has a solution with $2k$ vertices. The theorem then follows from Håstad's result that Independent Set is not approximable with a factor of $|V|^{(1-\epsilon)}$ for any $\epsilon > 0$ unless NP = ZPP and that Independent Set is not approximable with a factor of $O(|V|^{(1/2-\epsilon)})$ unless NP = P [Hås99].

Given an instance $G = (V, E)$ of Independent Set, we construct a new graph $H = (V_H, E_H)$, where V_H contains all vertices from V and for each vertex $v \in V$ a copy $\overline{v}$ of v. That is, $V_H := V \cup \overline{V}$ with $\overline{V} := \{\overline{v} \mid v \in V\}$. For each vertex $v \in V$, the edge set E_H contains an edge between v and its copy $\overline{v}$. Furthermore, for each edge $\{u, v\} \in E$, the graph H contains the four edges $\{u, v\}, \{\overline{u}, v\}, \{u, \overline{v}\}, \{\overline{u}, \overline{v}\}$.

To prove the correctness of this reduction, we show that G has an independent set I of size k iff there is a solution for Max-VIDPS on H consisting of

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

$2k$ vertices.

"⇒": If G has an independent set I, then the subgraph of H that is induced by I and the copy of I is clearly a union of $|I|$ vertex-disjoint one-edge paths and contains exactly $2|I|$ vertices.

"⇐": For this direction, we show that if there is a solution of size $2k$ for Max-VIDPS on H, then there exists always a solution V'_H of the same size such that $H[V'_H]$ is a set of $|V'_H|/2$ disjoint edges between the vertices in $V'_H \cap V$ and their copies in $\overline{V}$. In particular, this implies that $H[V'_H \cap V]$—and, thus, $G[V'_H \cap V]$—are edgeless graphs, and, hence, $V'_H \cap V$ is an independent set of size $|V'_H|/2$ in G.

To show the existence of a solution V'_H as described, let V''_H be an arbitrary solution for Max-VIDPS on H and let $I_{H[V''_H]}$ be a maximum-cardinality independent set in the graph $H[V''_H]$. Let

$$X := \{v, \overline{v} \mid v \in I_{H[V''_H]} \ \vee \ \overline{v} \in I_{H[V''_H]}\},$$

that is, the set X contains all vertices from $I_{H[V''_H]} \cap V$ together with their copies from $\overline{V}$ and all vertices from $I_{H[V''_H]} \cap \overline{V}$ together with their original vertices from V. We show that the set X has the claimed property of V'_H, that is, the set X is a size-$2k$ solution for Max-VIDPS on H and $H[X]$ is a set of $|X|/2$ disjoint edges between the vertices in $X \cap V$ and their copies in $\overline{V}$. The latter follows directly from the fact that $I_{H[V''_H]}$ is an independent set in $H[V''_H]$: Assume that $H[X]$ contains two edges sharing a common endpoint. Then at least one of these edges has to be of the type $\{u, v\}$, $\{\overline{u}, v\}$, $\{u, \overline{v}\}$, or $\{\overline{u}, \overline{v}\}$. This implies that $I_{H[V''_H]}$ contains at least one of the vertex pairs u and v, $\overline{u}$ and v, u and $\overline{v}$, or $\overline{u}$ and $\overline{v}$. However, these vertices are all adjacent in H and, thus, in $H[V''_H]$, which contradicts the fact that $I_{H[V''_H]}$ is an independent set in $H[V''_H]$.

The structure of X clearly implies that X is a feasible solution for Max-VIDPS on H; hence, it remains to show the size of X. Since $H[V''_H]$ is a set of vertex-disjoint paths, each maximum independent set of $H[V''_H]$ contains at least half of the vertices of $H[V''_H]$, that is, $|I_{H[V''_H]}| \geq |V''_H|/2$. Since $I_{H[V''_H]}$ cannot contain a vertex v together with its copy $\overline{v}$, we have $|X| = 2|I_{H[V''_H]}|$. Therefore, it holds that $|X| \geq |V''_H|$, and we obtain the claimed size of X. □

Since the argumentation in the proof of Theorem 4.1 works also for caterpillars instead of paths, the reduction used in the proof proves also the hardness of the problem Maximum Vertex-Induced Disjoint Caterpillars Subgraph (Max-VIDCatS): Here the task is, given a graph $G = (V, E)$, to find a maximum-cardinality vertex subset $V' \subseteq V$ such that $G[V']$ is a union of vertex-disjoint caterpillars. Hence, we obtain the following result.

Theorem 4.2. *There exists no polynomial-time factor-$O(|V|^{(1-\epsilon)})$ approximation algorithm, for any $\epsilon > 0$, for* Max-VIDCatS *unless* NP = ZPP, *and no polynomial-time factor-$O(|V|^{(1/2-\epsilon)})$ approximation algorithm unless* NP = P.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

As mentioned in Section 4.2, Max-VIDCatS in general is not equivalent to Max-COS-R on $(2,*)$-matrices. Therefore, Theorem 4.2 does not immediately imply the hardness of approximating Max-COS-R on $(2,*)$-matrices. However, in the special case where we have given an instance of Max-VIDCatS as constructed in the proof of Theorem 4.1, we can map this instance to an instance of Max-COS-R on $(2,*)$-matrices such that the size of an optimal solution is preserved:

Lemma 4.1. *Let $G = (V, E)$ be an instance of* Independent Set*, and let $H = (V_H = V \cup \overline{V}, E_H)$ be the graph constructed for G as described in the proof of Theorem 4.1. Then there is a solution for* Max-VIDCatS *on H that has cardinality d' iff there is a solution for* Max-COS-R *on the vertex-edge incidence matrix M of H that has cardinality d'.*

Proof. **"⇒":** Assume that there is a solution V''_H of size d' for Max-VIDCatS on H. Then, as argued in the proof of Theorem 4.1, there is also a solution V'_H of the same size such that $H[V'_H]$ is a set of $|V'_H|/2$ disjoint edges between the vertices in $V'_H \cap V$ and their copies in $\overline{V}$. In particular, this implies that every vertex in $H[V'_H]$ has degree one.

Let M' be the submatrix of M that consists of all columns of M and exactly those rows of M that correspond to the vertices in V'_H, that is, M' consists of the vertex-edge incidence matrix of $H[V'_H]$ plus some additional columns containing at most one 1 each. Since M contains only two 1s per row, the only submatrices from T that can occur in M are the matrices M_{I_k}, $k \geq 1$, and M_{IV}. The submatrix M' of M can only contain an M_{I_k}, $k \geq 1$, if $H[V'_H]$ contains a cycle, and M' can only contain an M_{IV} if $H[V'_H]$ contains a vertex of degree three; however, both settings contradict the structure of $H[V'_H]$, and, hence, M' must have the C1P.

"⇐": Assume that there is a solution—that is, a row set R—of size d' for Max-COS-R on M, and let M' be the submatrix of M that is induced by R. Moreover, let V''_H be the vertices in H corresponding to the rows of R. Then, by Theorem 2.2, the vertices in V''_H induce a set of vertex-disjoint caterpillars in H because the vertex-edge incidence matrix of $H[V''_H]$ is a submatrix of M' and, hence, has the C1P. □

With the equivalence between Max-VIDPS and Max-COS-C on $(*,2)$-matrices and the reduction from Max-VIDCatS to Max-COS-R on $(2,*)$-matrices given in Lemma 4.1, Theorems 4.1 and 4.2 imply the following results.

Theorem 4.3. *1. For* Max-COS-C *on $(*,2)$-matrices as well as for* Max-COS-R *on $(2,*)$-matrices, there exists no polynomial-time factor-$O(|V|^{(1-\epsilon)})$ approximation algorithm for any $\epsilon > 0$ unless* NP = ZPP*, and no polynomial-time factor-$O(|V|^{(1/2-\epsilon)})$ approximation algorithm unless* NP = P*.*

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

2. Max-COS-C *on* $(*, 2)$*-matrices as well as* Max-COS-R *on* $(2, *)$*-matrices is* W[1]*-hard with respect to the parameter* $d' =$ *"number of columns (or rows, respectively) in the submatrix to be found."*

3. *Assuming* P $\neq$ NP*,* Min-COS-C *on* $(*, 2)$*-matrices and* Min-COS-R *on* $(2, *)$*-matrices cannot be approximated within a factor of* 2.7212*.*

Proof. Statement 1 of the theorem follows from Theorems 4.1 and 4.2, from the equivalence between Max-COS-C on $(*, 2)$-matrices and Max-VIDPS, and from Lemma 4.1. For the correctness of statement 2, observe that the reduction given in the proof of Theorem 4.1 is also a parameterized reduction; the statement hence follows directly from the W[1]-completeness of Independent Set [DF99]. Since the minimization dual of Independent Set, that is, Vertex Cover, is not approximable within 1.3606 unless P = NP [DS05], the above reduction also yields statement 3. □

Note that the W[1]-hardness of Max-VIDPS and Max-VIDCatS can also be inferred from a more general result by Khot and Raman [KR02], and that the non-existence of a constant-factor approximation for these problem also follows from a more general result of Lund and Yannakakis [LY93]; however, the above inapproximability results are stronger.

4.4 Maximization on $(*, 2)$- and $(2, *)$-Matrices

In this section, we present a polynomial-time approximation and a fixed-parameter algorithm for Max-COS-R on $(*, 2)$-matrices and a fixed-parameter algorithm for Max-COS-C on $(2, *)$-matrices—for the latter problem, a polynomial-time approximation was already known [TZ07]. Like in the previous section, we use the equivalence between these two problems and the graph problems Max-WEIDPS and Max-WEIDCatS.

4.4.1 Max-COS-R on $(*, 2)$-Matrices

Max-COS-R on $(*, 2)$-matrices is equivalent to Max-WEIDPS (see Section 4.2). Therefore, to achieve the mentioned results for the former problem, it suffices to find a polynomial-time approximation and a fixed-parameter algorithm for Max-WEIDPS.

Approximation algorithm. We show that Max-WEIDPS can be approximated in polynomial time with a factor of 3/4, which immediately implies that Max-COS-R on $(*, 2)$-matrices is approximable with the same factor. For approximating Max-WEIDPS, we use a part of an algorithm of Serdyukov [Ser84] (see [BGS02, HR00] for descriptions of the algorithm in English language), which approximates the problem Max-TSP with a factor of 3/4 in $O(|V|^3)$ time.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Max-TSP
Input: A complete graph $G = (V, E)$, in which every edge $e \in E$ has a nonnegative weight $w(e)$.
Task: Find a cycle of maximum weight that contains every vertex from V exactly once.

Serdyukov's algorithm works on complete graphs, and as an intermediate step it produces a set of vertex-disjoint paths. The idea for approximating Max-WEIDPS is, therefore, to construct for a given instance (G, w) an edge-weighted complete graph G' such that every set of vertex-disjoint paths in G' one-to-one corresponds to a set of vertex-disjoint paths of the same weight in the original graph G. On this modified instance, we run the first phase of the algorithm of Serdyukov, which outputs a set of vertex disjoint paths whose weight is at least 3/4 times the weight of an optimal solution for Max-WEIDPS.

More precisely, for a given instance (G, w) of Max-WEIDPS, we construct an instance $(G' = (V, E'), w')$ of Max-TSP, where G' is the complete graph with vertex set V and where the weight of every edge $\{u, v\} \in E'$ is given by

$$w'(\{u, v\}) = \begin{cases} w(\{u, v\}) & \text{if } \{u, v\} \in E \\ 0 & \text{if } \{u, v\} \notin E. \end{cases}$$

It is obvious that if there is a subgraph H' of G' that consists of vertex-disjoint paths and has weight d', then there is also a subgraph H of G consisting of vertex-disjoint paths and having weight d': To obtain H, just remove all edges from H' that do not exist in G—the weight of these edges is 0.

The approximation algorithm of Serdyukov for Max-TSP constructs two or three (depending on whether $|V|$ is even or odd) subgraphs—so-called *partial tours*—, which consist of vertex-disjoint paths. Out of these partial tours the algorithm selects one with maximum weight and extends it to a cycle of length $|V|$ by adding some edges, which is always possible because the input graph is complete. The approximation factor of 3/4 is shown by proving that the weight of at least one of the partial tours is at least 3/4 times the weight of an optimal solution for Max-TSP [Ser84].

Clearly, the weight of an optimal solution for Max-WEIDPS on (G', w') is a lower bound for Max-TSP on (G', w') because every set of vertex-disjoint paths can be completed to a cycle of length $|V|$. Therefore, Serdyukov's algorithm always finds a set of vertex-disjoint paths whose weight is at least 3/4 times the weight of an optimal solution for Max-WEIDPS on (G', w'). As argued above, from this set of paths we can directly obtain a set of vertex-disjoint paths in G having the same weight, which yields the following result.

Theorem 4.4. Max-WEIDPS *can be approximated in* $O(|V|^3)$ *time with a factor of* 3/4.

With the equivalence between Max-COS-R on $(*, 2)$-matrices and Max-WEIDPS and since the edge-weighted complete graph G' can be constructed in $O(mn)$ time for a given $m \times n$ matrix, we obtain the following corollary.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Corollary 4.2. Max-COS-R *on $(*, 2)$-matrices can be approximated in $O(mn + n^3)$ time with a factor of $3/4$.*

Proof. Given an $m \times n$ matrix M, we need $O(mn)$ time for constructing an edge-weighted complete graph G' as described above. On this graph, which has n vertices, we run the approximation algorithm of Serdyukov in $O(n^3)$ time. □

Fixed-parameter algorithm. We show that Max-WEIDPS is fixed-parameter tractable with respect to the parameter d' = "weight of the vertex-disjoint paths to be found." As a consequence of the equivalence between Max-WEIDPS and Max-COS-R on $(*, 2)$-matrices, the latter problem is fixed-parameter tractable with respect to the number of rows in the desired submatrix.

The fixed-parameter tractability of Max-WEIDPS can easily be shown by using the color-coding approach introduced by Alon et al. [AYZ95] for the NP-complete problem Longest Path, which is defined as follows.

Longest Path
Input: A graph $G = (V, E)$ and a positive integer d'
Question: Is there a *simple* path—that is, a path containing no vertex more than once—in G that consists of at least d' vertices?

The idea of Alon et al. [AYZ95] for solving Longest Path in $f(d') \cdot n^{O(1)}$ time is to repeatedly randomly color the vertices of G with d' colors and to search, for each coloring, for a simple path of d' vertices that contains no two vertices of the same color. For a given coloring, such a path can be found, if existing, in $2^{O(d')} \cdot |E|$ time with a dynamic programming approach [AYZ95]; for every vertex of the input graph the dynamic programming algorithm tries to find a d'-vertex path that contains no color more than once and that starts at v.

If G contains a simple path P consisting of d' vertices, then the probability for the vertices of P being colored with d' different colors is $d'!/d'^{d'}$ because the number of possibilities for mapping the vertices of P to d' different colors is $d'!$ and the number of possibilities for mapping d' vertices to d' colors is $d'^{d'}$. Therefore, after $d'^{d'}/d'! = 2^{O(d')}$ iterations the expected value for the number of returned simple paths consisting of d' vertices is 1. The algorithm can be derandomized, which results in an overhead factor of $\log(|V|)$ and, therefore, in a running time of $2^{O(d')} \cdot \log(|V|) \cdot |E|$ for solving Longest Path deterministically [AYZ95].

The dynamic programming algorithm given by Alon et al. for finding a path that contains no two vertices of the same color can easily be adapted such that it is capable of handling edge weights. Therefore, the problem of finding a simple path consisting of at most d' vertices and having maximum edge weight under this property can also be solved in $2^{O(d')} \cdot \log(|V|) \cdot |E|$ time (see also [HWZ08]).

To solve Max-WEIDPS, we reduce the problem of finding a set of paths of total weight d' to the problem of finding *one* path of weight d'; to this end, we

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

use the same idea as described in the previous paragraph about approximating Max-WEIDPS: Given an instance (G, w) of Max-WEIDPS, we extend the graph G by adding edges of weight 0 in order to obtain a complete graph G'. The graph G' obviously has the property that every path of weight d' in G' corresponds to a weight-d' set of vertex-disjoint paths in G and vice versa. Now we repeatedly randomly color the graph G' and search with a dynamic programming algorithm [AYZ95, HWZ08] for a path of weight at least d' containing no color twice. Since we can assume that the desired path contains no two edges of weight 0 with a common endpoint, we only have to consider paths consisting of at most $2d'$ vertices, and, therefore, we have to use $2d'$ colors. Since the number of edges in G' is $|V|^2$ and since a simple path with at most d' vertices and maximum edge weight can be found deterministically in $2^{O(d')} \cdot \log(|V|) \cdot |E|$ time [AYZ95, HWZ08], we get the following theorem.

Theorem 4.5. Max-WEIDPS*, restricted to instances where every edge has a weight of at least 1, can be solved in* $2^{O(d')} \cdot |V|^2 \cdot \log(|V|)$ *time, where* d' *is the total weight of the paths to be found.*

With the equivalence between Max-COS-R on $(*, 2)$-matrices and Max-WEIDPS and since the edge-weighted complete graph G' can be constructed in $O(mn)$ time for a given $m \times n$ matrix, we obtain the following corollary.

Corollary 4.3. Max-COS-R *on* $(*, 2)$*-matrices can be solved in* $2^{O(d')} \cdot n^2 \cdot \log(n) + O(mn)$ *time, where* d' *is the number of rows in the submatrix to be found.*

We just mention in passing that the bases in the exponential parts of the running times (that is, the constants hidden by the Big-O-Notation in the terms $2^{O(d')}$) can be optimized by using the methods presented by Hüffner et al. [HWZ08] for speeding up color-coding algorithms. Moreover, by using the divide-and-conquer technique described by Chen et al. [CLSZ07, KMRR06] instead of color coding, the factor $\log(n)$ can be eliminated.

4.4.2 Max-COS-C on $(2, *)$-Matrices

For Max-COS-C on $(2, *)$-matrices it is known that there is a polynomial-time factor-0.5 approximation algorithm [TZ07]. We complement this result by showing that Max-COS-C on $(2, *)$-matrices is fixed-parameter tractable with respect to the parameter d' = "number of rows in the submatrix to be found." To this end, we exploit the equivalence between the problems Max-COS-C on $(2, *)$-matrices and Max-WeidCatS and present a fixed-parameter color-coding algorithm for solving the latter problem.

The basic ideas of our fixed-parameter algorithm for Max-WeidCatS are the same as used in the fixed-parameter algorithm for solving Max-WEIDPS in Section 4.4.1: First, we reduce the problem of searching for a set of caterpillars of

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

total weight d' to the problem of searching *one* caterpillar of weight d' by adding edges of weight 0 to the given graph G in order to obtain a complete graph G'. It is obvious that every caterpillar of weight d' in G' one-to-one corresponds to a weight-d' set of vertex-disjoint caterpillars in G. Second, we use the technique of color-coding as described in Section 4.4.1. Again, we have to use $2d'$ colors because we can assume that the desired caterpillar contains no two edges of weight 0 that have a common endpoint.

It remains to show how, given a graph G' whose vertices are colored with $2d'$ colors, one can determine whether there is a caterpillar of weight at least d' that contains no color more than once. We solve this task with a dynamic programming approach very similar to the one described in Section 4.4.1 for finding a path containing no color more than once. The pseudocode of the corresponding algorithm is displayed in Figure 4.3. The algorithm assigns to every vertex $v \in V$ a set X_v of tuples (C_v, ℓ_v), where C_v is a set of colors and ℓ_v is an integer. Each tuple (C_v, ℓ_v) in X_v corresponds to a caterpillar P that contains no color more than once and in which the vertex v is an *endpoint*: this means that v has degree at most 1 in P and that the single neighbor of v, if existing, has at most one neighbor of degree more than one. The set C_v contains all colors occurring in P, and ℓ_v is the total weight of the edges in P. In line 1, the set X_v for every vertex $v \in V$ is initialized with a pair corresponding to the caterpillar consisting of the single vertex v. In lines 2–13, the algorithm tries, for increasing values of i, to extend those caterpillars that contain i vertices. To this end, every endpoint v of each such caterpillar P is considered (line 3), and for each neighbor w of v it is checked whether the color of w already occurs in P (lines 5–6). If this is not the case, then every possibility is tried to add the vertex w together with some other neighbors of v to P, and the information about the resulting caterpillars is stored in X_w (lines 7–13), because w is an endpoint of every resulting caterpillar. The difference between this algorithm and the dynamic programming for finding a maximum-weight simple path [AYZ95, HWZ08] is that in the latter case, when a neighbor w of an endpoint v of a path is considered and the color of w does not occur in the path so far, then one just adds w to the path and does not consider the other neighbors of v. The running time of the algorithm in Figure 4.3 is $O(|V| + 2^{O(d')} \cdot |E|)$, and, with the same considerations as in Section 4.4.1 (in particular, we have to take into account an overhead factor of $\log(|V|)$ for the derandomization), we get the following theorem.

Theorem 4.6. Max-WeidCatS*, restricted to instances where every edge has a weight of at least 1, can be solved in $2^{O(d')} \cdot |V|^2 \cdot \log(|V|)$ time, where d' is the total weight of the caterpillars to be found.*

With the equivalence between Max-COS-C on $(2, *)$-matrices and Max-WeidCatS and since a given $m \times n$ matrix can be turned into an edge-weighted complete graph G' in $O(mn)$ time, we obtain the following corollary.

Corollary 4.4. Max-COS-C *on $(2, *)$-matrices can be solved in $2^{O(d')} \cdot m^2 \cdot$*

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Input: A positive integer d', a graph $G = (V, E)$ where every vertex $v \in V$ is colored with a color $\mathrm{clr}(v)$ chosen from $2d'$ colors and every edge $e \in E$ has a nonnegative integral edge weight $w(e)$.

Output: Is there a caterpillar of weight at least d' that contains no color more than once?

```
1:  assign to every vertex v a set X_v := {({clr(v)}, 0)};
2:  for i := 1 to 2d' − 1: {
3:      while there is a vertex v where X_v contains a pair (C_v, ℓ_v) with |C_v| = i: {
4:          remove (C_v, ℓ_v) from X_v;
5:          for each neighbor w of v: {
6:              if clr(w) ∉ C_v: {
7:                  C := ⋃_{u∈(N(v)\{w})} {clr(u)};
8:                  for each subset C' ⊆ C: {
9:                      if C' ∩ (C_v ∪ {clr(w)}) = ∅: {
10:                         ℓ := ℓ_v + w({v, w}) +
                                ∑_{c∈C'} max{w({v, u}) | u ∈ (N(v) \ {w}) ∧
                                                          clr(u) = c};
11:                         if ℓ ≥ d': return yes;
12:                         add the tuple (C_v ∪ {clr(w)} ∪ C', ℓ) to X_w;
13:                         remove all dominated tuples from X_w; }}}}}}
14: return no;
```

Figure 4.3: Algorithm for deciding whether a graph whose vertices are colored with $2d'$ colors there exists a caterpillar of weight d' that contains no color more than once. A pair $(C_w, \ell_w) \in X_w$ is *dominated* if there is another pair $(C'_w, \ell'_w) \in X_w$ with $C'_w = C_w$ and $\ell'_w \geq \ell_w$.

$\log(m) + O(mn)$ *time, where* d' *is the number of columns in the submatrix to be found.*

Again, the methods of Hüffner et al. [HWZ08] can be used for improving the running time of the the algorithm.

4.5 Minimization on $(*, \Delta)$-Matrices

In this section, we present our main algorithmic results of the chapter: polynomial-time approximations and fixed-parameter algorithms for Min-COS-C and Min-COS-R on $(*, \Delta)$-matrices. The section is structured as follows. The main idea behind the algorithms is explained in Section 4.5.1. In Section 4.5.2 we show how to deal with a subproblem: handling matrices that already have the Circ1P, but not the C1P. The analysis of the running times and approximation factors of our algorithms for $(*, \Delta)$-matrices is provided in Section 4.5.3. The proof for

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

the main structural theorem, which is introduced in Section 4.5.1 and on which all algorithms presented here are based, is deferred to Section 4.7.

4.5.1 Outline of the Algorithmic Framework

The basic algorithmic approach underlying all our algorithms reads as follows. In order to derive constant-factor polynomial-time approximation algorithms or fixed-parameter algorithms for Min-COS-C and Min-COS-R on $(*, \Delta)$-matrices, we exploit Theorem 2.5 by iteratively searching and destroying in the given input matrix every submatrix that is isomorphic to one of the forbidden submatrices from the set T given in Theorem 2.5: In the approximation scenario all columns or rows belonging to a forbidden submatrix are deleted, whereas in the fixed-parameter setting a search tree algorithm [HNW08, Nie06] branches recursively into several subcases, deleting in each case one of the columns or rows of the forbidden submatrix.

To show the performance guarantees of the thus derived algorithms, observe that a $(*, \Delta)$-matrix cannot contain submatrices of types M_{II_k} and M_{III_k} with arbitrarily large sizes. Therefore, the main difficulty is that every problem instance can contain submatrices of type M_{I_k} of unbounded size—the approximation factor or the number of cases to branch into would therefore not be bounded from above by Δ. To overcome this difficulty, we use the following two-phase approach:

1. Destroy only those forbidden submatrices that belong to a certain *finite* subset X of the set T (and whose sizes are upper-bounded, therefore).

2. Solve Min-COS-C or Min-COS-R for each component of the resulting matrix. According to Corollary 4.1, these solutions can be combined into a solution for the whole input matrix.

Concerning phase 2, we will see in Section 4.5.2 that Min-COS-C and Min-COS-R on these components can be solved in $f(\Delta) \cdot |M|^{O(1)}$ time because of the special structure of the components.

The finite set $X \subseteq T$ used in phase 1 is specified in the following theorem, the main structural contribution of this chapter. The technical proof is presented in Section 4.7.

Theorem 4.7. *Let* $X := \{M_{\mathrm{I}_k} \mid 1 \leq k \leq \Delta-1\} \cup \{M_{\mathrm{II}_k} \mid 1 \leq k \leq \Delta-2\} \cup \{M_{\mathrm{III}_k} \mid 1 \leq k \leq \Delta-1\} \cup \{M_{\mathrm{IV}}, M_{\mathrm{V}}\}$. *If a $(*, \Delta)$-matrix M contains none of the matrices in X as a submatrix, then each component of M has the Circ1P.*

We have already shown in Chapter 3 how small submatrices from T, if existing, can be found efficiently. Hence, according to Theorem 4.7, there remains the challenge of transforming a matrix with Circ1P into a matrix with C1P. This point will be addressed in the next section; a more detailed description of the algorithmic results for Min-COS-C and Min-COS-R follows in Section 4.5.3.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

4.5.2 From Circ1P to C1P

Here, we consider the problems Min-COS-C and Min-COS-R restricted to input matrices that have the Circ1P; these matrices arise in the second phase of the algorithmic skeleton described in Section 4.5.1. Since a Circ1-ordering for the columns of a matrix that has the Circ1P can be found in linear time (see Section 2.3), we can without loss of generality assume that our input matrices do not only have the Circ1P, but also the strong Circ1P.

Min-COS-C (Min-COS-R) asks to delete a minimum-cardinality set of columns (rows) in such a way that in the resulting matrix the 1s can be placed consecutively in every row by permuting the columns. We will show that if the number n of columns is big enough compared to Δ and if one starts with a matrix having the strong Circ1P, then the optimal solutions for Min-COS-C (Min-COS-R) have a special structure: It is always optimal to delete a set of columns (rows) in such a way that in the resulting matrix the 1s can be placed consecutively in every row *by a number of "cyclic shifts"*; that is, the circular column ordering of the resulting matrix must be a C1-circular ordering.

We will first prove this special structure of optimal solutions and then show how to exploit it when solving Min-COS-C and Min-COS-R on matrices with Circ1P.

Circ1-Orderings and C1-Orderings

In what follows, it is often helpful to imagine the matrices as wrapped around a vertical cylinder. Therefore, we have to use the concept of the circular column ordering (see Chapter 2.1) of a matrix M: the circular column ordering describes the order of M's columns when M is imagined as wrapped around a vertical cylinder. In addition, we need the following notions.

Definition 4.1. 1. If the column ordering of a binary matrix M is shift-equivalent to a C1-ordering for M's columns, then M has the *shifted strong C1P* and its column ordering is called a *shifted C1-ordering*.

2. Let M be a binary matrix with the strong Circ1P. If there is a column pair $(c_j, c_{\text{succ}_n(j)})$ such that in every row r_i containing both 1s and 0s at most one of $m_{i,j}$ and $m_{i,\text{succ}_n(j)}$ is 1, then the column pair $(c_j, c_{\text{succ}_n(j)})$ is called a *C1-cut*.

It follows directly from these definitions that a binary matrix M has the C1P iff there is a shifted C1-ordering for M's columns, and this is the case iff there is a Circ1-ordering for M's columns that yields a C1-cut: If the column ordering $c_1, \ldots, c_n$ of a matrix is a Circ1-ordering and $(c_j, c_{\text{succ}_n(j)})$ is a C1-cut, then the column ordering $c_{\text{succ}_n(j)}, \ldots, c_n, c_1, \ldots, c_j$ places the 1s consecutively in every row of the resulting matrix (see Fig. 4.4). Intuitively speaking, wrapping M around a vertical cylinder, cutting the matrix on the cylinder vertically from top

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

c_1	c_2	c_3	c_4	c_5
0	0	**1**	**1**	0
0	0	0	**1**	**1**
1	0	0	0	**1**
1	**1**	0	0	0

c_3	c_4	c_5	c_1	c_2
1	**1**	0	0	0
0	**1**	**1**	0	0
0	0	**1**	**1**	0
0	0	0	**1**	**1**

Figure 4.4: The matrix on the left has the shifted strong C1P because it is shift-equivalent to the matrix on the right, which has the strong C1P. For both matrices, the pair (c_2, c_3) is a C1-cut.

to bottom between c_j and $c_{\mathrm{succ}_n(j)}$, and unwrapping it from the cylinder places the 1s consecutively.

To prove the claimed structure of optimal solutions for Min-COS-C and Min-COS-R on matrices with the strong Circ1P, we show that if a matrix M has the C1P and the column number is big enough compared to Δ, then every Circ1-ordering for M's columns is a shifted C1-ordering; in other words, if the matrix has the strong Circ1P, then it also has the shifted strong C1P. To this end, we show that each Circ1-ordering for the columns of the matrix can be obtained from a shifted C1-ordering by a series of column reversal operations that do not destroy the shifted strong C1P. This reversal operation was originally used by Hsu and McConnell [HM03] to transform a circular column ordering into another circular column ordering.

Definition 4.2. Let $[c_1, \ldots, c_n]$ be the circular column ordering of a matrix. Given two column indices j_1, j_2, the operation reverse(c_{j_1}, c_{j_2}), applied to this circular ordering, reverses the order of the columns $c_{j_1}, \ldots, c_{j_2}$ if $j_1 < j_2$, and it reverses the order of the columns $c_{j_1}, \ldots, c_n, c_1, \ldots, c_{j_2}$ if $j_1 > j_2$.

The intuition of the reverse operation is that for a given matrix, which is wrapped around a vertical cylinder and represented by the circular ordering of its columns, we cut out a vertical stripe of this cylinder, reverse the order of the columns on this stripe, and put the stripe back into the cylinder. For example, reverse(c_{11}, c_3) applied to the circular ordering

$$[c_1, \ldots, c_{13}]$$

leads to the circular ordering

$$[c_{13}, c_{12}, c_{11}, \quad c_4, c_5, c_6, c_7, c_8, c_9, c_{10}, \quad c_3, c_2, c_1].$$

We define the reverse operation analogously for matrices and column orderings (instead of circular column orderings): For reversing the columns from j_1 to j_2 in a matrix M, we first wrap M around a vertical cylinder, then apply the reverse operation as described in Definition 4.2, and finally cut the matrix on the cylinder vertically from top to bottom and unwrap it from the cylinder. If c_1 and c_n are

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

still neighbors after reversing the columns, then this cut is made between c_1 and c_n; otherwise, there are two cases: if $j_2 = n$, then the cut is made to the left of c_1, and if $j_1 = 1$, then the cut is made to the right of c_n.

Definition 4.3 ([HM03])**.** Let M be a binary matrix, and let C be the set of its columns. A subset $C' \subseteq C$ is *uniform in row r* if all entries in row r in the columns of C' are the same. A *circular module* of M is a subset $C' \subseteq C$ such that in every row r the subset C' is uniform in r or $C \setminus C'$ is uniform in r.

Clearly, if a matrix M has the strong Circ1P, then applying the reverse operation to a set of columns that form a circular module does not destroy the strong Circ1P:

Observation 4.2. *Let M be a binary matrix with the strong Circ1P, and let M' be the matrix obtained from M by reversing an arbitrary circular module whose columns are consecutive in M's circular column ordering. Then M' has the strong Circ1P.*

Observation 4.2 is strengthened in the following Theorem due to Hsu and McConnell [HM03].

Theorem 4.8 ([HM03, Theorem 3.8])**.** *Let M be a matrix having the Circ1P. Then* every *Circ1-circular ordering for M's columns can be obtained by starting from some Circ1-circular ordering and applying a sequence of reverse operations, each of them reversing a circular module whose columns are consecutive in the respective circular column ordering.*

Note that reversing a circular module as described in Theorem 4.8 corresponds to flipping a subtree at a C-node in a PC-tree as described in Section 2.3.

We have defined the reverse operation not only for circular column orderings, but also for matrices and column orderings. Clearly, Theorem 4.8 also applies to the Circ1-orderings of M's columns instead of the Circ1-circular orderings: Each Circ1-ordering for the columns of a matrix can be obtained by starting from an arbitrary Circ1-ordering and applying a sequence of reverse operations, each of them reversing a circular module.

We can now state a useful relation between the Circ1-orderings and the shifted C1-orderings for the columns of matrices having the C1P. This observation is crucial for our algorithms solving Min-COS-C and Min-COS-R, since it implies that if n is big compared to Δ then it is optimal to delete a set of columns or rows, respectively, in such a way that the resulting matrix has the shifted strong C1P.

Lemma 4.2. *Let M be a $(*, \Delta)$-matrix of size $m \times n$, $n \geq 2\Delta - 1$, that has the C1P. Then every Circ1-ordering for M's columns is also a shifted C1-ordering.*

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Proof. Since M has the C1P, its columns can be permuted such that the resulting matrix M' has the shifted strong C1P. By definition, then M' also has the strong Circ1P. We will prove the following claim:

Claim: Let M' be a matrix with the shifted strong C1P, and let M'' be a matrix obtained from M' by applying the reverse operation to an arbitrary circular module of M'. Then M'' has the shifted strong C1P.

Due to Theorem 4.8, this claim suffices to prove the lemma, because every Circ1-ordering for M's columns can be obtained from M' by a series of reverse operations, and by the claim, none of these operations destroys the shifted strong C1P.

Proof of the claim: Let C be the column set of M', and let $c_1, \ldots, c_n$ be the column ordering of M' (which is a shifted C1-ordering). Moreover, let $C' \subseteq C$ be the circular module of M' whose reversal leads to M''. Since M' has the shifted strong C1P, there is at least one C1-cut in M'. Without loss of generality, let (c_n, c_1) be this C1-cut, that is, there is no row r_i in M' with $m_{i,n} = 1$ and $m_{i,1} = 1$ and $m_{i,j} = 0$ for at least one $j \in \{2, \ldots, n-1\}$.

If C' does not contain c_1 and c_n, then (c_n, c_1) clearly is still a C1-cut after the reversal. Moreover, in this case M'' has also the strong Circ1P because, due to Observation 4.2, the reversal of C' does not destroy this property. The shifted strong C1P and the existence of a C1-cut together imply the shifted strong C1P of M''. If C' contains both of c_1 and c_n, we can argue analogously because then (c_1, c_n) is a C1-cut in M''.

Now, assume that C' contains exactly one of c_1 and c_n, say c_1. Then $C' = \{c_1, \ldots, c_h\}$ with $h < n$, and M'' has the column ordering $c_h, \ldots, c_1, c_{h+1}, \ldots, c_n$. Assume for the sake of contradiction that none of (c_n, c_h) and (c_1, c_{h+1}) is a C1-cut in M''. Then there must be two rows r_{i_1} and r_{i_2} such that, on the one hand, $m_{i_1,n} = 1$ and $m_{i_1,h} = 1$, and, on the other hand, $m_{i_2,1} = 1$ and $m_{i_2,h+1} = 1$. Since (c_n, c_1) is a C1-cut in M', we have $m_{i_1,1} = 0$ and $m_{i_2,n} = 0$. Therefore, $m_{i_1,j} = 1$ for every $j \in \{h+1, \ldots, n-1\}$ and $m_{i_2,j} = 1$ for every $j \in \{2, \ldots, h\}$—otherwise, the set C' would not be a circular module. Since there are at most Δ 1s in each row, $|\{c_h, \ldots, c_n\}| \leq \Delta$ and, therefore, $h > n - \Delta \geq 2\Delta - 1 - \Delta = \Delta - 1$. For the same reason $|\{c_1, \ldots, c_{h+1}\}| \leq \Delta$ and, therefore, $h \leq \Delta - 1$, contradicting $h > \Delta - 1$. Hence, at least one of (c_n, c_h) and (c_1, c_{h+1}) must be a C1-cut in M'', which together with Observation 4.2 implies the shifted strong C1P of M''. □

Since deleting rows or columns from a matrix that has the strong Circ1P does not destroy the strong Circ1P, Lemma 4.2 has the following consequences for solving Min-COS-C and Min-COS-R: If the input matrix has the strong Circ1P, then deleting any set of columns or rows such that the C1P is obtained yields a matrix that has the shifted strong C1P (provided that in the resulting matrix Δ is small compared to the column number).

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Solving Min-COS-C on Matrices with the Circ1P

Here, we show how to use the results of Section 4.5.2 in order to solve Min-COS-C on matrices that have the Circ1P. We first give an upper bound on the solution size and then, exploiting Lemma 4.2, characterize the structure of optimal solutions and show how to find them efficiently.

Lemma 4.3. *Let M be a $(*, \Delta)$-matrix that has the Circ1P. Then* Min-COS-C *on input M can be solved by deleting at most Δ columns.*

Proof. Order the rows of M such that the resulting matrix M' has the strong Circ1P. Since each row of M' contains at most Δ 1s, the submatrix resulting from removing the leftmost Δ columns from M' has the (strong) C1P. □

Now we can show that there is always an optimal solution for Min-COS-C with some nice structure, provided that the input matrix has the strong Circ1P and Δ is small enough compared to n.

Lemma 4.4. *Let M be a $(*, \Delta)$-matrix of size $m \times n$, $n \geq 3\Delta - 1$, that has the strong Circ1P, let $c_1, \ldots, c_n$ be its column ordering, let the set C' of columns be an optimal solution for* Min-COS-C *on input M, and let M' be the matrix resulting from deleting C' from M.*[10] *Then,*

1. *M' has the shifted strong C1P and*
2. *the columns from C' are consecutive in the circular ordering of M's columns, and if c_α and c_β are the two columns to the left and to the right of C', (that is, $c_\alpha, c_\beta \notin C'$ and $c_{\mathrm{succ}_n(\alpha)}, c_{\mathrm{pred}_n(\beta)} \in C'$), then (c_α, c_β) is a C1-cut in M'.*

Proof. First, we show that the matrix M' fulfills the conditions of Lemma 4.2. Hence, by deleting C' from M, one obtains a matrix M' that does not only have the C1P, but also the shifted strong C1P. More precisely, due to Lemma 4.3, $|C'| \leq \Delta$, and, therefore, M' has at least $2\Delta - 1$ columns. By Lemma 4.2, this implies that M' has the shifted strong C1P, because the strong Circ1P is preserved when deleting columns. This proves statement 1.

To prove statement 2, let $c_{j_1}, \ldots, c_{j_{n'}}$ be the columns of M', that is, the columns of M that do not belong to C'. Without loss of generality, assume that $(c_{j_{n'}}, c_{j_1})$ is a C1-cut of M', that is, the column ordering $c_{j_1}, \ldots, c_{j_{n'}}$ places the 1s consecutively in every row. Suppose, for the sake of contradiction, that statement 2 does not hold, that is, there exists a column $c_x \in C'$ such that when M is wrapped around a vertical cylinder, the column c_x appears to the right of c_{j_1} and to the left of $c_{j_{n'}}$.

Let M'' be the matrix that results from M' by inserting the column c_x at its "old position", that is, M'' results from M by deleting the columns $C' \setminus \{c_x\}$.

[10] When columns are deleted, the remaining columns retain the numbering scheme of the original matrix.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Clearly, M'' has the strong Circ1P because M has the strong Circ1P. Moreover, the insertion of c_x into M' does not affect the fact that $(c_{j_{n'}}, c_{j_1})$ is a C1-cut. Hence, the matrix M'' also has the shifted strong C1P. This means that $C' \setminus \{c_x\}$ is also a solution of Min-COS-C, contradicting the optimality of C' as a solution. □

By Lemma 4.4, the columns of an optimal solution C' are consecutive in *every* Circ1-ordering for M's columns. Hence, an optimal solution can easily be found:

Theorem 4.9. Min-COS-C*, restricted to $(*, \Delta)$-matrices of size $m \times n$ that have the Circ1P, can be solved in $O(\Delta mn)$ time if $n \geq 3\Delta - 1$, and in $O((3\Delta)^{\min\{d,\Delta\}} \cdot \Delta m)$ time otherwise, where d is the number of allowed column deletions.*

Proof. Let M be a $(*, \Delta)$-matrix of size $m \times n$ that has the Circ1P. Due to Lemma 4.3, an optimal solution for Min-COS-C on M has size at most Δ. If $n < 3\Delta - 1$, then an optimal solution can be found by trying all possibilities to delete at most $\min\{d, \Delta\}$ columns (there are $\binom{n}{\min\{d,\Delta\}} = O((3\Delta)^{\min\{d,\Delta\}})$ possibilities) and checking in $O(\Delta m+n)$ time [BL76] whether the resulting matrix has the C1P. If $n \geq 3\Delta - 1$, then assume that M has the strong Circ1P (a Circ1-ordering for M's columns can be found in $O(\Delta m + n)$ time [BL76]). Due to Lemma 4.4, there exists an optimal solution C' that is consecutive in the circular column ordering of M and that is enclosed by the columns of a C1-cut in the matrix resulting from the deletion of C'. This solution can be found by checking, for every column pair $(c_j, c_{j'})$ with at most Δ columns lying between c_j and $c_{j'}$ in the circular column ordering of M, whether the submatrix of M that consists of the columns $c_1, \ldots, c_j, c_{j'}, \ldots, c_n$ has the strong C1P with $(c_j, c_{j'})$ being a C1-cut. For such a check, simply test in $O(m)$ time whether for every row r_i at least one of $m_{i,j}$ and $m_{i,j'}$ is 0. □

Solving Min-COS-R on Matrices with the Circ1P

In the case of Min-COS-R, we cannot upper-bound the size of an optimal solution as we did in Lemma 4.3. However, Lemma 4.2 yields a characterization of optimal solutions for Min-COS-R that is very similar to the one given in Lemma 4.4 for Min-COS-C.

Lemma 4.5. *Let M be a $(*, \Delta)$-matrix of size $m \times n$, $n \geq 2\Delta - 1$, that has the strong Circ1P, let the set R' of rows be an optimal solution for* Min-COS-R *on input M, let M' be the matrix that results from deleting R' from M, and let $c_1, \ldots, c_n$ be the column ordering of M and M'. Then,*

1. *M' has the shifted strong C1P and*
2. *there is a C1-cut $(c_j, c_{\mathrm{succ}_n(j)})$ in M' such that*

$$R' = \{r_i \mid (1 \leq i \leq m) \wedge (r_i \text{ contains 0s and 1s}) \wedge (m_{i,j} = m_{i,\mathrm{succ}_n(j)} = 1)\}.$$

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Proof. Lemma 4.2 implies that M' has the shifted strong C1P because M' obviously has the strong Circ1P. This proves statement 1. To prove statement 2, let $(c_j, c_{\mathrm{succ}_n(j)})$ be an arbitrary C1-cut in M'. On the one hand, due to the definition of a C1-cut, there can be no row in M' that contains a 1 in both c_j and $c_{\mathrm{succ}_n(j)}$ and that contains at least one 0. Hence, all rows r_i with $m_{i,j} = 1$ and $m_{i,\mathrm{succ}_n(j)} = 1$ and $m_{i,j'} = 0$ for at least one j' must be part of R'. On the other hand, suppose, for the sake of a contradiction, that there exists a row in R' that does not contain a 1 in both c_j and $c_{\mathrm{succ}_n(j)}$ or that contains only 1s or only 0s. Then, re-inserting this row into M' results in a matrix that still has the strong C1P—a contradiction to the optimality of R'. □

In analogy to Min-COS-C (Theorem 4.9), an optimal solution can now easily be found by exploiting Lemma 4.5.

Theorem 4.10. Min-COS-R*, restricted to* $(*, \Delta)$*-matrices of size* $m \times n$ *that have the Circ1P, can be solved in* $O(mn)$ *time if* $n \geq 2\Delta - 1$*, and in* $O((2\Delta)^{2\min\{d, 4\Delta^2\}} \cdot \Delta m)$ *time otherwise, where* d *is the number of allowed row deletions.*

Proof. Let M be a $(*, \Delta)$-matrix of size $m \times n$ that has the Circ1P. If $n < 2\Delta - 1$, then first eliminate duplicate rows: assign weights to the rows such that every row gets as weight the number of its occurrences, and delete all occurrences except for one of every row. The number of rows of the resulting matrix is bounded from above by $(2\Delta)^2$ because if M has the strong Circ1P, then every row can be described uniquely by the index of the first and last column containing a 1 in this row, which yields $(2\Delta)^2$ possibilities. The task is now to find a row set of minimum weight whose deletion yields the C1P. An optimal solution can be found by trying all possibilities to delete at most $\min\{d, (2\Delta)^2\}$ rows and checking in $O(\Delta m + n)$ time [BL76] whether the resulting matrix has the C1P; the number of possibilities to try is $\binom{(2\Delta)^2}{\min\{d,(2\Delta)^2\}}$.

If $n \geq 2\Delta - 1$, then assume that M has the strong Circ1P (a Circ1-ordering for M's columns can be found in $O(\Delta m + n)$ time [BL76]). Due to Lemma 4.5, an optimal solution can be found by counting for every column pair $(c_j, c_{\mathrm{succ}_n(j)})$ in $O(m)$ time the number of rows r_i with $m_{i,j} = 1$ and $m_{i,\mathrm{succ}_n(j)} = 1$ and $m_{i,j'} = 0$ for at least one j'; deleting these rows results in a matrix with C1-cut $(c_j, c_{\mathrm{succ}_n(j)})$. □

4.5.3 The Algorithms in Detail

As sketched in the algorithmic skeleton of Section 4.5.1, our approximation algorithms for Min-COS-C and Min-COS-R consist of two phases: First, they search in every step for a matrix belonging to the set X of forbidden submatrices given by Theorem 4.7, and, then, they delete all columns (rows) of the found submatrix. Since an optimal solution has to delete at least one column (row)

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

of every forbidden submatrix from X, the approximation factor is bounded from above by the maximum number of columns (rows) of a submatrix found during this phase. Thereafter, due to Theorem 4.7, all components of the remaining matrix have the Circ1P. In case of Min-COS-C, a solution of size at most Δ can be found for every component by permuting its columns such that the strong Circ1P is obtained and then deleting the first Δ columns (as shown in the proof of Lemma 4.3)—clearly, this yields a factor-Δ approximation for every component. The overall approximation factor is determined by the one achieved in the first phase of the algorithm. In case of Min-COS-R, we do not have such a simple factor-Δ approximation for solving the problem on the components of the matrix resulting from the first phase. Hence, we use the approach of Theorem 4.10 for exactly solving Min-COS-R on every component resulting from the first phase. Note that to derive polynomial running times for fixed Δ, we can ignore the term d in the running time of Theorem 4.10.

The fixed-parameter search tree algorithms search in every step for a forbidden submatrix of X and then branch on which column (row) belonging to the found submatrix shall be deleted. A solution for the resulting matrices without submatrices from X (but possibly still with submatrices of the type M_{I_k}, $k \geq \Delta$) can be found without branching, see Theorem 4.9 and Theorem 4.10.

To find a submatrix from X, we use the algorithms presented in Chapter 3. If we use the search algorithm specified in Theorem 3.1, then the maximum number of columns or rows in a forbidden submatrix found during the first phase of the algorithm is identical to the maximum number of columns or rows of a matrix in X. If the (slightly faster) algorithm specified in Proposition 3.1 is used, then we need the following corollary of Proposition 3.1 to determine this number of columns or rows.

Corollary 4.5. *Let M be a $(*, \Delta)$-matrix of size $m \times n$.*

1. *If $\Delta = 3$ or $\Delta = 4$ and the algorithm in Figure 3.1 does not find a forbidden submatrix from T consisting of at most 9 rows (columns), or*
2. *if $\Delta = 2$ or $\Delta \geq 5$ and the algorithm in Figure 3.1 does not find a forbidden submatrix from T consisting of at most $\Delta + 4$ rows (columns),*

then M does not contain a forbidden submatrix from the set X specified in Theorem 4.7.

Proof. Assume that M contains a submatrix M' from X consisting of m' rows and n' columns. The number of rows and the number of columns of the matrix returned by the algorithm is upper-bounded by

$$\begin{array}{ll} k+2 & \text{if } M' = M_{\mathrm{I}_k}, \\ k+3 & \text{if } M' = M_{\mathrm{II}_k}, \\ k+5 & \text{if } M' = M_{\mathrm{III}_k}, \\ 9 & \text{if } M' = M_{\mathrm{IV}}, \text{ and} \\ 5 & \text{if } M' = M_{\mathrm{V}}, \end{array}$$

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Table 4.4: Summary of results for MIN-COS-C and MIN-COS-R on $(*, \Delta)$-matrices.

Approximation algorithms

MIN-COS-C	Factor	Running time	based on
$\Delta = 2$	4	$O(m^2n^3)$	Cor. 4.6, Lem. 4.3
$\Delta = 3$	6	$O(m^3n^2 + m^2n^4)$	Thm. 3.1, Lem. 4.3
$\Delta \geq 4$	Δ+2	$O(\Delta^3m^3n^2 + \Delta^3m^2n^4)$	Thm. 3.1, Lem. 4.3
$\Delta = 2, 5, 6, \ldots$	Δ+4	$O(\Delta mn^3 + n^4)$	Cor. 4.5, Lem. 4.3
$\Delta = 3, 4$	9	$O(mn^3 + n^4)$	Cor. 4.5, Lem. 4.3
MIN-COS-R	Factor	Running time	
$\Delta = 2$	3	$O(m^3n^2)$	Cor. 4.6, Thm. 4.10
$\Delta \geq 3$	Δ+1	$O((2\Delta)^{8\Delta^2} \cdot \Delta m^2 + \Delta^3m^4n + \Delta^3m^3n^3)$	Thms. 3.1, 4.10
$\Delta = 2, 5, 6, \ldots$	Δ+4	$O((2\Delta)^{8\Delta^2} \cdot \Delta m^2 + \Delta m^2n^2 + mn^3)$	Cor. 4.5, Thm. 4.10
$\Delta = 3, 4$	9	$O(m^2n^2 + mn^3)$	Cor. 4.5, Thm. 4.10

Fixed-parameter algorithms

MIN-COS-C	Running time	based on
$\Delta = 2$	$O(4^d \cdot m^2n^2)$	Cor. 4.6, Thm. 4.9
$\Delta = 3$	$O(6^d \cdot (m^2n \cdot (m + n^2)))$	Thms. 3.1, 4.9
$\Delta \geq 4$	$O((\Delta+2)^d \cdot ((3\Delta)^{\min\{d,\Delta\}} \cdot \Delta dm + \Delta^3m^3n + \Delta^3m^2n^3))$	Thms. 3.1, 4.9
$\Delta = 2, 5, 6, \ldots$	$O((\Delta+4)^d \cdot ((3\Delta)^{\min\{d,\Delta\}} \cdot \Delta dm + \Delta mn^2 + n^3))$	Cor. 4.5, Thm. 4.9
$\Delta = 3, 4$	$O(9^d \cdot (mn^2 + n^3))$	Cor. 4.5, Thm. 4.9
MIN-COS-R	Running time	
$\Delta = 2$	$O(3^d \cdot m^2n^2)$	Cor. 4.6, Thm. 4.10
$\Delta \geq 3$	$O((\Delta+1)^d \cdot ((2\Delta)^{2\cdot\min\{d,4\Delta^2\}} \cdot \Delta dm + \Delta^3m^3n + \Delta^3m^2n^3))$	Thms. 3.1, 4.10
$\Delta = 2, 5, 6, \ldots$	$O((\Delta+4)^d \cdot ((2\Delta)^{2\cdot\min\{d,4\Delta^2\}} \cdot \Delta dm + \Delta mn^2 + n^3))$	Cor. 4.5, Thm. 4.10
$\Delta = 3, 4$	$O(9^d \cdot (mn^2 + n^3))$	Cor. 4.5, Thm. 4.10

as described in the proof of Proposition 3.1. Since, on the one hand, the matrices of the type M_{I_k} in X have $k \leq \Delta - 1$, the matrices of the type M_{II_k} in X have $k \leq \Delta - 2$, and the matrices of the type M_{III_k} in X have $k \leq \Delta - 1$, and, on the other hand, the matrix M can contain an M_{IV} as a submatrix only if $\Delta \geq 3$, it follows that the algorithm returns a matrix that has the claimed number of rows and columns. □

A second corollary, following from Proposition 3.3, helps us to find matrices from X very fast in the case where $\Delta = 2$:

Corollary 4.6. *Let M be a $(*, 2)$-matrix of size $m \times n$. A forbidden submatrix from X in M that has a minimum number of columns (rows) can be found in $O(m^2n^2)$ time.*

Proof. This corollary follows directly from Proposition 3.3 because X contains only the matrices M_{I_1} andM_{III_1}. □

Finally, we can precisely analyze the performance of our algorithms.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Theorem 4.11. Min-COS-C *and* Min-COS-R*, restricted to $(*, \Delta)$-matrices, have constant-factor approximation algorithms as shown in Table 4.4. Moreover,* Min-COS-C *and* Min-COS-R *are fixed-parameter tractable with respect to the combined parameter Δ, d, where d denotes the number of allowed column deletions and row deletions, respectively. The running times are given in Table 4.4.*

Proof. In the case of the approximation algorithms, the approximation factor is determined by the number of columns (rows) of the submatrices found in the first phase of the algorithm. If the algorithm in Figure 3.1 with the running time given in Proposition 3.1 is used for searching forbidden submatrices from X, then the column number (row number) is determined by Corollary 4.5. If, otherwise, the algorithm behind Theorem 3.1 (or Corollary 4.6 in the case of $\Delta = 2$) is used, then the column number (row number) is equal to the maximum taken over the column numbers (row numbers) of the matrices in X. Since at most n columns (m rows) can be deleted, the running time for every algorithm is n times (m times) the time needed for searching a forbidden submatrix (see Proposition 3.1, Theorem 3.1, and Corollary 4.6) plus n times (m times) the time needed for approximating Min-COS-C (solving Min-COS-R) on a component that has the Circ1P (see Lemma 4.3, Theorem 4.10).

In case of the search-tree algorithms, the number of branches depends on the maximum number of columns (rows) of a forbidden submatrix found during the first phase of the algorithm, which destroys all submatrices from X, and is either determined by Corollary 4.5 or by the maximum taken over the column numbers (row numbers) of the matrices in X—depending on which algorithm is used for searching the forbidden submatrices. The time needed in each node of the search tree is given by the time needed to search for a submatrix from X (see Proposition 3.1, Theorem 3.1, and Corollary 4.6) plus, in the case that no submatrix from X was found, the time needed for solving Min-COS-C (Min-COS-R) (see Theorems 4.9 and 4.10) on at most d components that have the Circ1P. □

4.6 Minimization on $(*, 2)$- and $(2, *)$-Matrices

We turn our attention to the still NP-hard special cases of $(*, 2)$- and $(2, *)$-matrices and present some kernelization and approximation results for Min-COS-C and Min-COS-R. Section 4.6.1 describes a kernelization for these two problems in the case of $(*, 2)$-matrices; Sect 4.6.2 describes approximation and fixed-parameter algorithms for $(2, *)$-matrices.

4.6.1 $(*, 2)$-Matrices

We show that the problems Min-COS-C and Min-COS-R, restricted to $(*, 2)$-matrices, admit problem kernels with respect to the parameter $d =$ "number of column/row deletions."

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Min-COS-C

As described in Section 4.2, Min-COS-C with each row containing at most two 1s is equivalent to the problem Min-VIDPS, where, given a graph $G = (V, E)$ and a positive integer d, the question is whether there is a vertex subset $V' \subseteq V$ with $|V'| \leq d$ whose removal transforms G into a union of vertex-disjoint paths.

We achieve the kernelization result for Min-COS-C by showing a problem kernel for Min-VIDPS. Without loss of generality, we assume that G is a connected graph because otherwise we can solve the problem on each connected component separately. Given an instance $(G = (V, E), d)$ of Min-VIDPS, we perform the following polynomial-time data reduction:

Rule 1: If a vertex v has more than $d+2$ neighbors, then remove v from G, add v to V', and decrease d by one.

Rule 2: If a degree-two vertex v has two degree-at-most-two neighbors u and w with $\{u, w\} \notin E$, then remove v from G and connect u, w by an edge.

Rule 3: If a degree-1 vertex v is adjacent to a degree-at-most-two vertex, then remove v from G.

Rule 4: If there is a vertex of degree 0, delete it.

A graph to which none of the four rules applies is called *reduced*.

Lemma 4.6. *The data reduction rules are correct, and a graph $G = (V, E)$ can be transformed into a reduced instance in $O(|V| \cdot \log(|V|) + |E|)$ time.*

Proof. Concerning the correctness of Rule 1, without removing v from G and adding it to the solution set V', we would have to remove at least $d + 1$ of v's neighbors from G to make v a degree-two vertex. However, with at most d vertex deletions allowed, this is impossible. Thus, Rule 1 is correct.

To see the correctness of Rule 3, let $G = (V, E)$ be a graph containing a degree-1 vertex v that is adjacent to a degree-at-most-two vertex w. We have to show that G has a solution of size at most d iff $G[V \setminus \{v\}]$ has a solution of size at most d. Clearly, if V' is a solution of size at most d for G, then $V' \setminus \{v\}$ is a solution of size at most d for $G[V \setminus \{v\}]$. To see the other direction, let $G'' = (V'', E'')$ be a union of vertex-disjoint paths that results from deleting at set V' of at most d vertices from $G[V \setminus \{v\}]$. The vertex w has degree at most 1 in G''. Adding the vertex v and the edge $\{v, w\}$ to G'' does neither lead to a cycle nor to a degree-at-least-3 vertex. Hence, the resulting graph is still a union of vertex-disjoint paths. This implies that every solution V' for Min-VIDPS on $G[V \setminus \{v\}]$ is also a solution for Min-VIDPS on G.

The correctness of Rule 2 can be seen in a similar way as the correctness of Rule 3: Let $G = (V, E)$ be a graph containing a degree-2 vertex v whose two neighbors u, w are both degree-at-most-two vertices and are not directly

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

connected by an edge. Here, we have to show that G has a solution of size at most d iff the graph $\tilde{G} := (V \setminus \{v\}, E \cup \{\{u, w\}\} \setminus \{\{u, v\}, \{v, w\}\})$ has a solution of size at most d. To prove the "only if" direction, let V' be a solution of size at most d for G. Clearly, if $v \notin V'$, then V' is also a solution for $\tilde{G}$. For the case $v \in V'$, observe that we can assume that V' does not contain all three vertices u, v, w, because if $G[V \setminus \{u, v, w\}]$ is a union of vertex-disjoint paths, then this also holds for $G[V \setminus \{u, w\}]$. Without loss of generality, let $u \notin V'$. It is easy to see that $V' \cup \{u\} \setminus \{v\}$ is a solution of size at most d for $\tilde{G}$. The "if" direction can be seen in analogy to the correctness argument for Rule 3.

The correctness of Rule 4 is obvious.

Using a data structure that allows to determine the degree of a vertex in constant time and to delete a vertex v from G in $O(\deg(v))$ time, G can be transformed into a reduced instance in $O(|V| \cdot \log(|V|) + |E|)$ time: First, try, once for each vertex, to apply Rule 1; thereby, use a priority queue [CLRS01] and always select a vertex of highest degree. Thereafter, for every $i \in \{2, 3, 4\}$ and for every vertex v, try *once* to apply Rule i to v. Note that, for any $i \in \{1, 2, 3, 4\}$, if Rule i cannot be applied to a vertex v at a certain point in time, then Rule i still cannot be applied to v after applying any series of rules from Rules $i, i + 1, \ldots, 4$ to the other vertices in G. Applying Rule 1 to all vertices needs $O(|V| \cdot \log(|V|) + |E|)$ time when using a priority queue; applying every other rule to all vertices needs $O(|V| + |E|)$ time. □

Theorem 4.12. Minimum Vertex-Induced Disjoint Paths Subgraph *with parameter d denoting the allowed vertex deletions admits a problem kernel with $O(d^2)$ vertices and $O(d^2)$ edges.*

Proof. Suppose that a given Min-VIDPS instance (G, d) is reduced with respect to Rules 1–4 and has a solution, that is, by deleting a vertex subset V' with $|V'| \leq d$, the resulting graph $H = (V_H, E_H)$ becomes a union of vertex-disjoint paths. Then, H only contains degree-one and degree-two vertices, denoted by V_H^1 and V_H^2, respectively: $V_H = V_H^1 \cup V_H^2 = V \setminus V'$.

On the one hand, since Rule 1 has removed all vertices of degree greater than $d + 2$ from G, the vertices in V' are adjacent to altogether at most $d^2 + 2d$ V_H-vertices in G. On the other hand, consider a V_H^1-vertex v. If v is not a degree-one vertex in G, then v is adjacent to at least one vertex from V'; otherwise, due to Rule 3, v's neighbor is adjacent to at least one vertex from V'. Moreover, due to Rule 2 and the fact that H is a union of vertex-disjoint paths, at least one of three consecutive degree-two vertices on a path from H is adjacent in G to at least one vertex from V'. Hence, at least $|V_H|/3$ vertices in V_H are adjacent in G to V'-vertices. Thus, the number of V_H-vertices can be upper-bounded by $3 \cdot (d^2 + 2d)$.

Since H is a union of vertex-disjoint paths, there can be at most $|V_H| - 1 = 3d^2 + 6d - 1$ edges in H. Moreover, as shown above, each vertex from V' can have at most $d + 2$ incident edges in G. Thus, we can conclude $|E| = O(d^2)$. □

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Note that for obtaining the quadratic kernel we do not have to transform G into a *reduced instance*; it is sufficient to try once for every $i \in \{1, 2, 3, 4\}$ and every vertex v to apply Rule i to v—the order in which the vertices are considered is irrelevant for obtaining the kernel. The resulting graph consists of $O(d^2)$ vertices and $O(d^2)$ edges, although maybe Rule 1 could be applied to some vertex v of the kernel. (This can happen if the degree of v is at most $d + 2$ when trying to apply Rule 1 during the kernelization. Since the value of d decreases when Rule 1 is applied to vertices different from v, the degree of v may be greater then the "new" value of $d + 2$ after the kernelization.) Thus, the kernelization can be performed in $O(|V| + |E|)$ time since the priority queue is not needed.

Corollary 4.7. Min-COS-C *on* $(*, 2)$*-matrices admits a problem kernel consisting of a matrix with* $O(d^2)$ *rows and columns.*

Min-COS-R

The kernelization algorithm for Min-COS-R is based on the equivalence between Min-COS-R on $(*, 2)$-matrices and the graph problem Min-WEIDPS described in Section 4.2, where the task is to transform an edge-weighted given graph into a union of vertex-disjoint paths by *edge deletions*.

A problem kernel for Min-WEIDPS can be obtained as follows. Given an instance $(G = (V, E), d)$ of Min-WEIDPS, perform the following linear-time data reduction similar to the data reduction for Min-VIDPS shown above:

Rule 1: If a vertex v has more than $d+2$ neighbors, then report that the instance is a *no*-instance.

Rule 2: If a degree-two vertex v has two degree-at-most-two neighbors u and w with $\{u, w\} \notin E$, then remove v from G and connect u and w by an edge of weight $\min\{w(\{u, v\}), w(\{v, w\})\}$.

Rule 3: If a degree-1 vertex v is adjacent to a degree-at-most-two vertex, then remove v from G.

Rule 4: If there is a vertex of degree 0, delete it.

Lemma 4.7. *The data reduction rules are correct and a graph* $G = (V, E)$ *can be transformed into a reduced instance in* $O(|V| + |E|)$ *time.*

Proof. This lemma can be proven analogously to the proof of Lemma 4.6. Note that we do not have to use a priority queue here; therefore, the data reduction takes only $O(|V| + |E|)$ time. □

Theorem 4.13. Min-WEIDPS *admits a problem kernel with at most* $8d$ *vertices and* $9d$ *edges.*

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Proof. Suppose that a given Min-WEIDPS-instance $(G = (V, E), d)$ is reduced with respect to Rules 1–4 and has a solution, that is, deleting an edge subset E' of total weight at most d yields a graph $H = (V_H, E_H)$ that is a union of vertex-disjoint paths. Let V_1 be the set of degree-1 vertices in G, let V_2 be the set of degree-2 vertices in G, and let V_3 be the set of degree-at-least-3 vertices in G. Every vertex in V_3 must be incident to at least one edge of E', which implies that $|V_3| \leq 2d$. Moreover, $\sum_{v \in V_3}(\deg(v) - 2) \leq 2d$ because otherwise there exists no solution of weight at most d for G. To see this, assume, for the sake of a contradiction, that $\sum_{v \in V_3}(\deg(v) - 2) > 2d$. Note that E' consists of at most d edges because $w(e) \geq 1$ for every edge in G. Therefore, after deleting the edges of E' from G, we would have $\sum_{v \in V_3}(\deg(v) - 2) > 2d - 2|E'| \geq 0$, and, hence, $\sum_{v \in V_3} \deg(v) > 2|V_3|$; however, this would be a contradiction to the fact that the degree of every vertex in H is at most two.

Since, due to Rule 3, every vertex in V_1 is adjacent to a vertex in V_3, and, due to Rules 2 and 3, every vertex in V_2 is also adjacent to a vertex in V_3, it holds that $|V_1 \cup V_2| \leq \sum_{v \in V_3} \deg(v) \leq 6d$. Hence, the number of vertices in V is at most $8d$. Since H is a union of vertex-disjoint paths, there can be at most $|V_H| - 1 \leq 8d - 1$ edges in H. Therefore, the number of edges in G is at most $|V_H| - 1 + d < 9d$. □

By Theorem 4.13 and the equivalence between Min-COS-R and Min-WEIDPS, we arrive at the following corollary.

Corollary 4.8. Min-COS-R *on $(*, 2)$-matrices with parameter d denoting the number of allowed row deletions admits a problem kernel consisting of a matrix with with less than $9d$ different rows, less than $8d$ columns, and an overall number of at most $9d^2 + 9d$ rows.*

Proof. The first statement follows directly from Theorem 4.13 and the one-to-one correspondence between Min-COS-R and Min-WEIDPS. Moreover, if a row appears more than $d + 1$ times, all occurrences except for $d + 1$ can be deleted. To see this, consider a set R' of at least $d + 1$ identical rows. Clearly, with d row deletions allowed, not all rows of R' can be deleted. This observation still holds for the problem instance with all but $d+1$ rows from R' deleted, which leads to the upper bound for the overall number of rows. □

Similar observations as described here have been used by Fernau [Fer05a, Fer08] for solving a graph problem called Linear Arrangement by Deleting Edges—this problem is equivalent to Min-WEIDPS without edge weigths. For this problem, Fernau gives a fixed-parameter algorithm running in $O^*(2.4676^d)$ time and a problem kernel consisting of $6d$ vertices and $6d$ edges.

Note that due to the equivalence between Min-COS-R and Min-CO-1E in the case of $(*, 2)$-matrices (see Section 4.2), our kernelization result also applies to Min-CO-1E (where the parameter is the number of allowed flippings of 1s).

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

4.6.2 $(2, *)$-Matrices

In contrast to Sections 4.5 and 4.6.1, we are now going to examine matrices that have no bound on the number of 1s per row. We obtain fixed-parameter tractability and approximation results for matrices having at most two 1s per column.

By considering the matrices from the set T given in Theorem 2.5, one can immediately see that if a matrix M does not have the C1P, then M can only contain an M_{IV} or an M_{I_k} (see Figure 4.2), because all other matrices in T contain a column with more than two 1s. The following lemma is the central observation for the subsequent results.

Lemma 4.8. *Let M be a $(2, *)$-matrix without identical columns. If M neither contains M_{IV} nor M_{I_1} as submatrices and does not have the C1P, then the matrices of type M_{I_k} that are contained in M are pairwise disjoint, that is, they have no common row or column.*

Proof. This lemma can easily be seen by considering the representing graph G_M of a $(2, *)$-matrix M without identical columns. First, note that vertices in the bipartite graph G_M that correspond to columns of M have degree two; the absence of identical columns in M implies that there are no cycles of length 4 in G_M.

Now assume that M neither contains M_{IV} nor M_{I_1} as submatrices, but that M contains two matrices of type M_{I_k} that share at least one common row or column. This means that in G_M there are two induced cycles that have at least one vertex in common. Since vertices corresponding to columns of M have degree 2, these two cycles must share at least one vertex corresponding to a row of M. Moreover, since M contains no M_{I_1} and, therefore, G_M contains no cycle of length 6 as an induced subgraph, the two cycles have length at least 8. It is now easy to see that G_M must contain a G_{IV} (see Figure 2.7) as an induced subgraph such that the center vertex of the G_{IV} subgraph is one of the vertices shared by the two cycles in G_M. Since G_{IV} is the representing graph of M_{IV}, the matrix M contains an M_{IV}—a contradiction. □

Min-COS-C

Based on Lemma 4.8, we can easily derive a search tree algorithm for Min-COS-C restricted to $(2, *)$-matrices:

1. Merge identical columns of the given matrix M into one column; assign to this column a weight equal to the number of columns it represents.

2. As long as M contains an M_{IV} or an M_{I_1} as a submatrix, branch into at most six subcases, each corresponding to deleting a column from the M_{IV}- or the M_{I_1}-submatrix. By deleting a column, the parameter d is decreased by the weight of that column.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

3. If there is no M_{IV} and M_{I_1} contained in M, then, by Lemma 4.8, the remaining submatrices of type M_{I_k} in M are pairwise disjoint. Then, Min-Cos-C is solvable in polynomial time on such a matrix: Delete a column with minimum weight from each remaining matrix of type M_{I_k}.

Clearly, the search tree size is $O(6^d)$, and a matrix of type M_{I_1} or M_{IV} can be found in $O(\min\{m^4n, m^2n^3\})$ time (see Propositions 3.2 and 3.3). By removing all columns of every M_{I_1}- and M_{IV}-submatrix found in M instead of branching into several subcases, we get a polynomial-time constant-factor approximation algorithm. The following theorem summarizes these results.

Theorem 4.14. Min-Cos-C*, restricted to $(2,*)$-matrices with m rows and n columns, can be solved in $O(6^d \cdot \min\{m^4n, m^2n^3\})$ time where d denotes the number of allowed column deletions. Moreover,* Min-Cos-C*, restricted to $(2,*)$-matrices, can be approximated in polynomial time with a factor of 6.*

Note that similar observations have been used for solving a graph problem called 2-Layer Planarization [Fer05a, Fer05b, Sud05, SW05]—this problem is equivalent to Min-Cos-C restricted to $(2,*)$-matrices without duplicate columns; the currently fastest algorithm for this problem has a running time of $O(3.562^d + |M|)$ [Sud05]. Note also that there is a linear[11] problem kernel for 2-Layer Planarization [DFH+06].

Min-Cos-R

The search tree algorithm for Min-Cos-R restricted to $(2,*)$-matrices is also based on Lemma 4.8:

1. If a column of the given matrix M appears more than once, delete all occurrences except for one.

2. As long as M contains an M_{IV} or an M_{I_1} as a submatrix, branch into at most four subcases, each corresponding to deleting a row from the M_{IV}- or M_{I_1}-submatrix found in M. By deleting a row, the parameter d is decreased by one.

3. If there is no M_{IV} and no M_{I_1} contained in M, then, by Lemma 4.8, the remaining submatrices of type M_{I_k} in M are pairwise disjoint. Then, Min-Cos-R is solvable in polynomial time on such a matrix: Delete an arbitrary row from each remaining matrix of type M_{I_k}.

The search tree size is $O(4^d)$, and a matrix of type M_{IV} can be found in $O(\min\{m^4n, m^2n^3\})$ steps. Again, by removing all rows of every M_{IV}- and M_{I_1}-submatrix instead of branching, we get a polynomial-time approximation algorithm.

[11] See footnote 1 on page 81.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

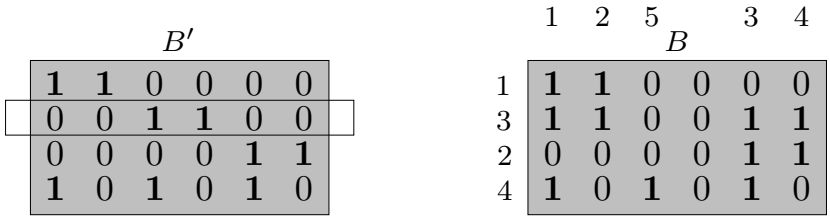


Figure 4.5: Illustration for Case 1 in the proof of Theorem 4.7. Complementing the second row of an M_{IV} (left matrix) generates an M_{V} (right matrix). (The rows and columns of the M_{V} are labeled with numbers according to the ordering of the rows and columns of the M_{V} in Figure 4.2.) Note that complementing also the fourth row of the matrix B' does not affect the existence of an M_{V} in B.

Theorem 4.15. Min-Cos-R*, restricted to* $(2,*)$*-matrices with* m *rows and* n *columns, can be solved in* $O(4^d \cdot \min\{m^4 n, m^2 n^3\})$ *time where* d *denotes the number of allowed row deletions. Moreover,* Min-Cos-R*, restricted to* $(2,*)$*-matrices, can be approximated in polynomial time with a factor of* 4*.*

4.7 Proof of the Structural Theorem

Finally, we give the proof of Theorem 4.7, which serves as the basis of all algorithms in Section 4.5. The theorem states that if a $(*,\Delta)$-matrix M contains none of the matrices from $X := \{M_{\mathrm{I}_k} \mid 1 \leq k \leq \Delta - 1\} \cup \{M_{\mathrm{II}_k} \mid 1 \leq k \leq \Delta - 2\} \cup \{M_{\mathrm{III}_k} \mid 1 \leq k \leq \Delta - 1\} \cup \{M_{\mathrm{IV}}, M_{\mathrm{V}}\}$ as a submatrix, then each component of M has the Circ1P.

Proof. We prove Theorem 4.7 by contraposition. More precisely, we show that if a component of a $(*,\Delta)$-matrix M does not have the Circ1P, then this component contains a submatrix from X. To this end, let A be a component of M not having the Circ1P. Then, by Corollary 2.1, there must be a column c of A such that the matrix A', resulting from A by complementing those rows that have a 1 in column c, does not have the C1P and, therefore, contains one of the submatrices in T (Theorem 2.5). In the following, we will make a case distinction based on which of the forbidden submatrices in T is contained in A' and which rows of A have been complemented, and show that in each case the matrix A contains a forbidden submatrix from X.

We denote the forbidden submatrix contained in A' with B' and the submatrix of A that corresponds to B' with B. Note that the matrix A' must contain a 0-column due to the fact that all 1s in column c have been complemented. Since no forbidden submatrix in T contains a 0-column, column c cannot belong to B' and, hence, not to B. We call c the *complementing column* of A.

When referencing to row or column indices of B', we will always assume that the rows and columns of B' are ordered as shown in Figure 4.2.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

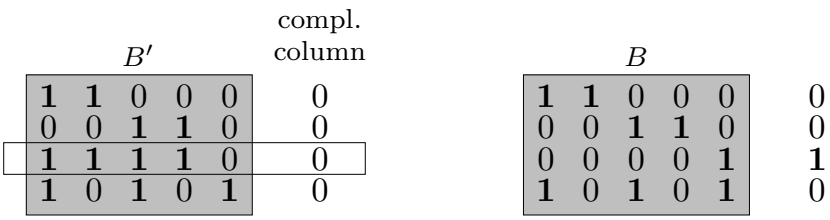


Figure 4.6: Illustration for Case 2 in the proof of Theorem 4.7. Suppose that only the third row of B is complemented. Then B together with the complementing column forms an M_{IV}.

Case 1: The submatrix B' is isomorphic to M_{IV}.

If no row of B has been complemented, then $B = B'$, and A also contains a submatrix M_{IV}, which belongs to X.

If exactly one of the first three rows of B has been complemented such that the resulting matrix is isomorphic to M_{IV}, then B contains one 0-column, and B without the 0-column forms an M_{V}, independent of whether the fourth row of B also has been complemented (see Figure 4.5 for an example). Again, we have shown that A contains a submatrix from X.

If two or three of the first three rows of B have been complemented, then A contains an $M_{\mathrm{I}_1} \in X$ as a submatrix: Assume, for instance, that the first two rows have been complemented. If the fourth row has also been complemented, then there is an M_{I_1} consisting of the rows r_1, r_2, r_4 and the columns c_2, c_4, c_5 of B. Otherwise, there is an M_{I_1} consisting of the rows r_1, r_2, r_4 and the columns c_1, c_3, c_6 of B.

Case 2: The submatrix B' is isomorphic to M_{V}.

Analogously to Case 1 we can make a case distinction on which rows of A have been complemented, and in every subcase we can find a forbidden submatrix from X in A. In some of the subcases the forbidden submatrix can only be found in A if in addition to B also the complementing column of A is considered. We will present only one representative example for all subcases of Case 2: If only the third row of B has been complemented, then the complementing column of A contains a 0 in all rows that belong to B except for the third. Then B forms an M_{IV} together with the complementing column of A (see Figure 4.6).

Case 3: The submatrix B' is isomorphic to M_{I_k} with $k \leq \Delta - 1$.

Subcase 3.1: No row of B has been complemented. Then $B = B'$, and A also contains a submatrix M_{I_k}.

Subcase 3.2: Exactly one row of B has been complemented. Then, together with the complementing column of A, the matrix B forms an M_{III_k}.

Subcase 3.3: At least two, but not all rows of B have been complemented. If $k = 1$, then B contains a 0-column, and B with the 0-column deleted forms an M_{I_1} together with the complementing column of A. Otherwise, let $r_i, r_{i'}$ with $i' > i + 1$ be two complemented rows where no row $r_{i''}$ with $i < i'' < i'$

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

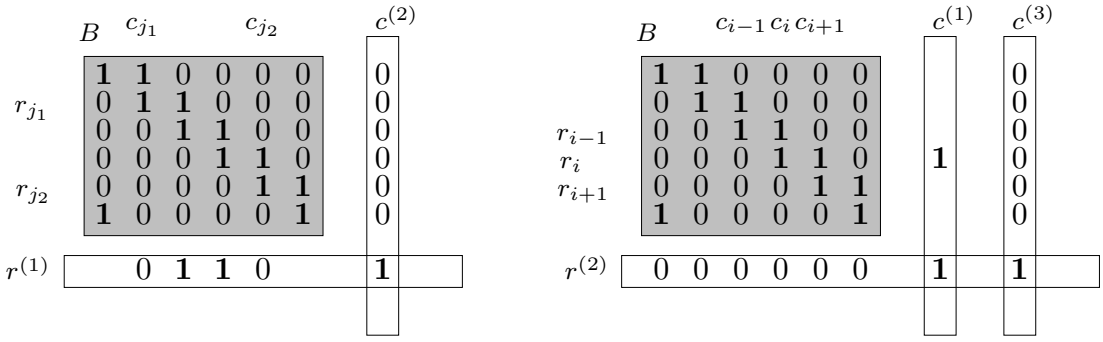


Figure 4.7: Illustration of Case 4 in the proof of Theorem 4.7.

has been complemented. (We can assume that two such rows r_i and $r_{i'}$ exist because we can permute the rows and columns belonging to B' in an appropriate way due to the symmetry of B'.) If $i' = i + 2$, then the rows r_i, r_{i+1}, r_{i+2} and columns c_{i+1}, c_{i+2} of B form an M_{I_1} together with the complementing column of A. Otherwise, the rows $r_i, \ldots, r_{i'}$ and columns $c_{i+1}, \ldots, c_{i'}$ of B form an $M_{\mathrm{II}_{i'-i-2}}$ together with the complementing column of A. Note that $M_{\mathrm{II}_{i'-i-2}} \in X$ because $i' - i - 2 \leq k - 1 \leq \Delta - 2$.

Subcase 3.4: All rows of B have been complemented. If $k = 1$, then B forms an M_{III_1} together with the complementing column of A; if $k = 2$, then B forms an M_{I_2}; otherwise, there is an M_{I_1} consisting of the rows r_1, r_2, r_4 and the columns c_1, c_3, c_4 of B.

Case 4: The submatrix B' is isomorphic to M_{I_k} with $k \geq \Delta$.

Then no row of B has been complemented, because otherwise there would be a row in A that contains more than Δ 1s (note that the complementing column of A contains a 1 in every row that is complemented). Therefore, $B = B'$, and A also contains an M_{I_k}—but note that $k \geq \Delta$ and, therefore, $M_{\mathrm{I}_k} \notin X$.

Let c be the complementing column of A. Since no row of B has been complemented, the column c contains no 1 in a row that belongs to B—hence, the distance between c and B is greater than 1. However, column c must be connected to B due to the definition of a component, and, therefore, there must be a shortest path from c to B.

Now, make a case distinction on the parity of the distance between c and B.

Subcase 4.1: The distance between c and B is even. Then there is a shortest path $c^{(0)}, r^{(1)}, c^{(2)}, \ldots, c$ in A between B and c with $c^{(0)} \in B$. (If the distance between c and B is two, then $c = c^{(2)}$.) Since the distance between c and B is even, the line $c^{(0)}$ must be a column. This means that the row $r^{(1)}$ does not belong to B, but has a 1 in a column that belongs to B and a 1 in column $c^{(2)}$. Column $c^{(2)}$ does neither belong to B nor does it have a 1 in a row that belongs to B. This constellation is displayed in the left part of Figure 4.7. Since $k \geq \Delta$, the matrix B has at least $\Delta + 2$ columns, and at least three columns of B must contain a 0 in row $r^{(1)}$. Without loss of generality, let c_{j_1} and c_{j_2} with $j_2 >$

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

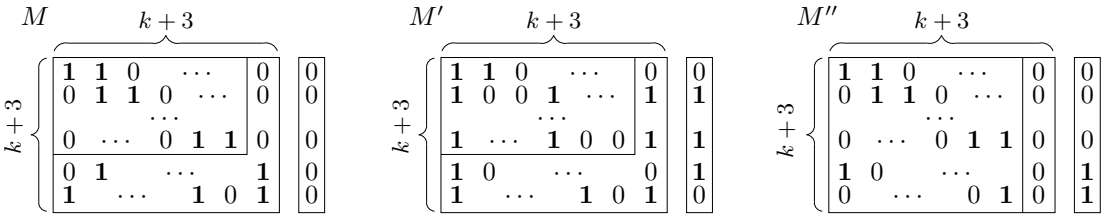


Figure 4.8: An illustration of the claim used in Case 5 of the proof of Theorem 4.7. Matrix M is composed of an M_{II_k} and a 0-column. Complementing rows r_2, r_{k+1}, and r_{k+2} of M leads to the matrix M'. Complementing the rows of M' that have a 1 in column c_{k+3}, namely, r_2, r_{k+1}, and r_{k+3}, transforms M' to matrix M'' which contains an $M_{\mathrm{I}_{k+1}}$ and a 0-column c_{k+3}.

$j_1 + 1$ be two columns containing a 0 in row $r^{(1)}$ such that all entries of row $r^{(1)}$ between c_{j_1} and c_{j_2} are 1s. (We can assume that such two columns c_{j_1} and c_{j_2} exist due to the symmetry of B'.) Then there is an $M_{\mathrm{III}_{j_2-j_1-1}}$ consisting of the rows $r_{j_1}, \ldots, r_{j_2-1}, r^{(1)}$ and columns $c_{j_1}, \ldots, c_{j_2}, c^{(2)}$. Since there can be at most Δ 1s in a row, we have $j_2 - j_1 - 1 \leq \Delta - 1$, and, therefore, $M_{\mathrm{III}_{j_2-j_1-1}} \in X$.

Subcase 4.2: The distance between c and B is odd. Then there is a shortest path $r^{(0)}, c^{(1)}, r^{(2)}, c^{(3)}, \ldots, c$ in A between B and c with $r^{(0)} \in B$. (If the distance between c and B is three, then $c = c^{(3)}$.) This means that the column $c^{(1)}$ does not belong to B, but it has a 1 in a row $r_i = r^{(0)}$ that belongs to B. (We can assume that $i > 1$ and $i < k + 2$ due to the symmetry of B'.) Row $r^{(2)}$ neither belongs to B nor does it have a 1 in a column that belongs to B, but it has 1s in the columns $c^{(1)}$ and $c^{(3)}$. Column $c^{(3)}$ does neither belong to B nor does it have a 1 in a row that belongs to B. This constellation is displayed in the right part of Figure 4.7. If column $c^{(1)}$ contains a 0 in row r_{i-1} as well as in row r_{i+1}, then there is an M_{IV} consisting of the rows $r_{i-1}, r_{i+1}, r^{(2)}, r_i$ and columns $c_{i-1}, \ldots, c_{i+2}, c^{(1)}$, $c^{(3)}$. If column $c^{(1)}$ contains a 1 in at least one of the rows r_{i-1} and r_{i+1}, say in r_{i-1}, then there is an M_{III_1} consisting of the rows $r_{i-1}, r_i, r^{(2)}$ and columns $c_{i-1}, c_{i+1}, c^{(1)}, c^{(3)}$.

Case 5: The submatrix B' is isomorphic to M_{II_k} with $k \geq 1$.

Here, we can re-use the argumentation for A' containing an $M_{\mathrm{I}_{k+1}}$ (Case 3 and Case 4), since the matrix type M_{II_k} is closely related to M_{I_k}, as shown in the following claim.

Claim: For an integer $k \geq 1$, let M be a $(k+3) \times (k+4)$ matrix composed of an M_{II_k} and an additional 0-column, and let M' be any matrix resulting from M by complementing a subset of its rows. Then, complementing all rows of M' that have a 1 in column c_{k+3} results in a matrix containing $M_{\mathrm{I}_{k+1}}$ and an additional 0-column.

Proof of the claim: Let $R \subseteq \{1, 2, \ldots, k+3\}$ be the set of the indices of the rows that have been complemented in M in order to form M'. After complementing the rows r_i with $i \in R$ in M, the column c_{k+3} of M' contains 1s

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

in all rows r_i with $i \in (\{1, \ldots, k+1\} \cap R) \cup (\{k+2, k+3\} \setminus R)$. It is easy to see that complementing these rows in M' results in the described matrix, proving the claim. See Figure 4.8 for an illustration of the claim.

We return to the proof of Case 5. The matrix B' together with a 0-column has been created by complementing a subset of the rows belonging to B. Applying the above claim, regarding B' together with the 0-column as the matrix M mentioned in the claim, shows that there is a column c_j in A such that complementing all rows that contain a 1 in column c_j results in an $M_{\mathrm{I}_{k+1}}$ and a 0-column. Then, A must contain a submatrix from X as we have shown in Case 3 and Case 4.

Case 6: The submatrix B' is isomorphic to M_{III_k} with $k \geq 1$.

Similarly to Case 5, this case can be reduced to Case 3 or Case 4 by applying the following claim, which reveals the relationship between matrix types M_{III_k} and M_{I_k}. This claim can be proven in analogy to the claim in Case 5.

Claim: For an integer $k \geq 1$, let $M = M_{\mathrm{III}_k}$, and let M' be any matrix resulting from M by complementing a subset of its rows. Then, complementing all rows of M' that have a 1 in column c_{k+3} results in a $(k+2) \times (k+3)$ matrix containing M_{I_k} and an additional 0-column. □

4.8 Conclusion

We have presented a systematic analysis of the complexity of various restricted variants of the in general NP-hard problems Min-COS-C, Min-COS-R, Max-COS-C, and Max-COS-R. Besides hardness for some of the problem variants, we could achieve polynomial-time approximation algorithms and exact fixed-parameter algorithms in many cases. In addition, our algorithms for Min-COS-C and Min-COS-R on $(*, \Delta)$-matrices can easily be adapted to work for the problem of deleting a minimum number of rows *and* columns. However, there remain questions unanswered: Our main results focus on Min-COS-C and Min-COS-R with a restricted number of 1s per row, but no restriction on the number of 1s in the columns; similar studies would be desirable for the case that we have a restricted number of 1s per column, but no restriction for the rows. Moreover, it should be investigated whether the running times for Min-COS-R and Max-COS-C (see Table 4.4) can be improved. In particular, we think that approximating Min-COS-R with a factor of $\Delta + 1$ should be possible within a running time that is polynomial in the input size and has no exponential factor depending on Δ. An interesting research direction is also to consider the problem Min-CO-1E (flipping a minimum number of 1-entries). We conjecture that for $(*, \Delta)$-matrices the presented approximation and fixed-parameter tractability results for Min-COS-C should extend to Min-CO-1E—however, we could not prove that (the reason is that, in the case of Min-CO-1E, we have no statements similar to Lemmas 4.4 and 4.5). Only for $\Delta = 2$ we have algorithmic results simply based on the equivalence to Min-COS-R in this case.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Chapter 5

Red-Blue Covering Problems

Chapters 3 and 4 dealt with finding and destroying those parts of a matrix that conflict with the C1P—the task was to establish the C1P. In contrast, the following two chapters study combinatorial problems where the objective has nothing to do with the C1P. We analyze the behavior of these problems on input matrices that have the C1P or are "close" to having the C1P.

Chapter 5 deals with variants of the subset selection problem Set Cover. Each of these Set Cover variants can be modelled as a matrix problem on a matrix with two types of rows ("red" and "blue" rows) and is NP-complete in general. We examine whether the problems become polynomial-time solvable when some or all rows of the input matrix contain at most one block of 1s. Thereby, we explore a sharp border between polynomial-time solvability and NP-hardness.

5.1 Introduction and Overview

Covering problems are of central importance in algorithm theory and combinatorial optimization. Two of the most prominent examples for this type of problem are Set Cover and Hitting Set. In both problems, the input consists of a set S and a collection $\mathcal{C}$ of subsets of S. For Set Cover, one tries to find a minimum-size subcollection $\mathcal{C}' \subseteq \mathcal{C}$ that *covers* S, that is, it satisfies $\bigcup_{C \in \mathcal{C}'} C = S$ (see Section 2.5 for an alternative definition of Set Cover as a matrix problem). For Hitting Set, one tries to find a minimum-size subset $S' \subseteq S$ that *covers* $\mathcal{C}$, that is, each set in $\mathcal{C}$ contains at least one element from S'. It is well-known that both problems are equivalent[1] in this general setting [ADP80]. Due to their practical importance, there is a lot of literature on Set Cover and Hitting Set [CLRS01, CP93, CTF00]. Set Cover is NP-hard and only allows for a logarithmic-factor polynomial-time approximation [Fei98]. Moreover,

[1] Generally, a set cover problem, where elements have to be covered by sets, can be equivalently formulated as a hitting set problem, where sets have to be covered by elements, by simply exchanging elements and sets.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

it is W[2]-complete (that is, parameterized intractable) with respect to the parameter "solution size" [DF99]. Due to the equivalence between Set Cover and Hitting Set, these results also apply to Hitting Set.

Generalizations as well as restrictions of Set Cover and Hitting Set played a prominent role in algorithmics. In this chapter, we are going to study two covering problems with an important generalization called "red-blue" together with the consecutive-ones property, which we apply to both problems. The first covering problem is called Minimum-Degree Hypergraph (MDH) and is defined as follows:

Minimum-Degree Hypergraph (MDH)
Input: A set S, two collections $\mathcal{C}_{blue}$ and $\mathcal{C}_{red}$ of subsets of S, and a nonnegative integer k.
Question: Is there a subset $S' \subseteq S$ such that

$$\forall C \in \mathcal{C}_{blue} : |S' \cap C| \geq 1, \text{ and}$$
$$\forall C \in \mathcal{C}_{red} : |S' \cap C| \leq k?$$

Feder et al. [FMZ06] introduced this problem and gave a factor-$O(s \cdot \log(|\mathcal{C}_{blue}|))$ polynomial-time approximation algorithm for its minimization version, where s denotes the maximum number of occurrences of an element from S in the sets of $\mathcal{C}_{red}$. Motivated by applications concerning interference reduction in cellular networks (see also Section 1.1), Kuhn et al. [KRW+05] introduced the Minimum Membership Set Cover problem, a special case of MDH. Here, given a set S and a collection $\mathcal{C}$ of subsets of S, one wants to determine a subcollection $\mathcal{C}' \subseteq \mathcal{C}$ that covers S but where the maximum number of occurrences of each element from S in the subsets in $\mathcal{C}'$ shall be minimized. MMSC is the special case of MDH where $\mathcal{C}_{blue} = \mathcal{C}_{red}$.

Our second covering problem within the "red-blue setting", the so-called Red-Blue Set Cover (RBSC) problem, has been introduced by Carr et al. [CDKM00] and is defined as follows.

Input: Two disjoint sets B (blue elements) and R (red elements), a collection $\mathcal{C}$ of subsets of $B \cup R$, and a nonnegative integer k.
Question: Is there a subcollection $\mathcal{C}' \subseteq \mathcal{C}$ such that

$$\forall b \in B \; \exists C \in \mathcal{C}' : b \in C, \text{ and}$$
$$\Big|\Big(\bigcup_{C \in \mathcal{C}'} C\Big) \cap R\Big| \leq k?$$

Set Cover is the special case of RBSC where each set in $\mathcal{C}$ contains exactly one red element and no red element is contained in more than one set. Carr et al. provided several natural application scenarios such as data mining for

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

RBSC and several positive and negative results concerning the polynomial-time approximability of RBSC.

To emphasize the close relationship between RBSC and MDH, we present the following, equivalent definition of RBSC.[2] This definition will be made use of in the remainder of this chapter.

Red-Blue Set Cover (RBSC)
Input: A set S, two collections $\mathcal{C}_{blue}$ and $\mathcal{C}_{red}$ of subsets of S, and a nonnegative integer k.
Question: Is there a subset $S' \subseteq S$ such that

$$\forall C \in \mathcal{C}_{blue} : |S' \cap C| \geq 1, \text{ and}$$
$$|\{C \in \mathcal{C}_{red} \mid S' \cap C \neq \emptyset\}| \leq k?$$

The difference between RBSC and MDH is that in the case of RBSC the number of red sets containing elements of the solution set is restricted, whereas in the case of MDH the maximum number of elements of a red set being contained in the solution set is restricted. Consider, for example, the subset system

$$S = \{a, b, c\},$$
$$\mathcal{C}_{blue} = \{\{a\}, \{b, c\}\},$$
$$\mathcal{C}_{red} = \{\{a, b\}, \{c\}\}.$$

The only solution for RBSC with $k = 1$ on this subset system is $S' = \{a, b\}$, whereas the only solution for MDH with $k = 1$ is $S' = \{a, c\}$.

As to the consecutive-ones property, this property was defined for matrices so far. Applied to instances of the problems MDH and RBSC, the C1P means that the elements of S can be ordered in a linear arrangement such that each set in $\mathcal{C}_{blue}$ and $\mathcal{C}_{red}$ contains only a whole "chunk" of that arrangement, that is, without any gaps. This definition of the C1P as a property of a subset system is consistent with the C1P of matrices as we know it from the previous chapters: one just has to think of the subset system in an MDH or RBSC instance as a coefficient matrix M where the elements in the ground set correspond to columns and the sets in the subset collection correspond to rows (see also Section 2.3, where the C1P was also applied to subsets of a ground set in this way).

Set Cover instances with the C1P are solvable in polynomial time, a fact which is made use of in many practical applications [MSW05, MW04, NW88, RS04, VW62] (see also Sections 2.4 and 2.5). In applications of MDH or RBSC with geographic background (such as the interference reduction problem considered by Kuhn et al. [KRW+05] or the very similar problem about cellular networks that we described in Section 1.1), the problem instances may have the C1P or be

[2] This equivalence can be seen, similar to the equivalence between Set Cover and Hitting Set, by exchanging elements and sets. The sets B and R in the original definition correspond to the collections $\mathcal{C}_{blue}$ and $\mathcal{C}_{red}$ in the equivalent formulation.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

"close" to having the C1P [MSW05, MW04, RS04]. Katz et al. [KMN05] considered geometric Set Cover stabbing problems that are also related to covering problems under the C1P restriction: some of their problems can be reduced to Set Cover on matrices with the C1P.

Contributions. In this chapter, we bring together the concepts of "red-blue" and the C1P, that is, we investigate the time complexity of the two red-blue covering problems with the C1P. The formulations of MDH and RBSC open a wide field of natural investigations concerning the C1P, the point being that the C1P may apply to either $\mathcal{C}_{blue}$, $\mathcal{C}_{red}$, $\mathcal{C}_{blue} \cup \mathcal{C}_{red}$, or none of $\mathcal{C}_{blue}$ and $\mathcal{C}_{red}$.

On the positive side, we show polynomial-time solvability for MDH and RBSC in the case that $\mathcal{C}_{blue} \cup \mathcal{C}_{red}$ possesses the C1P. In addition, we provide a simple greedy algorithm that approximates RBSC with $\mathcal{C}_{blue} \cup \mathcal{C}_{red}$ having the C1P to an additive term of one. On the negative side, we prove several NP-completeness results in case that at most one of $\mathcal{C}_{red}$ and $\mathcal{C}_{blue}$ has the C1P. More specifically, we indicate several sharp borders between polynomial-time solvability and NP-completeness of MDH depending on the subset sizes (the main point being, roughly speaking, a distinction between subset sizes two and three). Moreover, we show that if at most one of $\mathcal{C}_{red}$ and $\mathcal{C}_{blue}$ has the C1P, then also RBSC becomes NP-complete.

5.2 Basic Facts and Definitions

Formally, the consecutive-ones property on subset systems is defined as follows.

Definition 5.1. Given a set $S = \{s_1, \ldots, s_n\}$ and a collection $\mathcal{C}$ of subsets of S, the collection $\mathcal{C}$ is said to have the *C1P* if there exists a linear order $\prec$ on S such that for every set $C \in \mathcal{C}$ and $s_i \prec s_k \prec s_j$, it holds that $s_i \in C \wedge s_j \in C \Rightarrow s_k \in C$.

Given a subset system $(S, \mathcal{C})$, the linear order $\prec$ in Definition 5.1 can be found in $O(|S| + |\mathcal{C}| + \sum_{C \in \mathcal{C}} |C|)$ time (see Section 2.3). Therefore, in all our algorithmic results except Theorem 5.2 we can without loss of generality assume that the elements of the set S in the input are already sorted according to the order $\prec$.

The following simple observation is useful for our NP-completeness proofs.

Observation 5.1. *Given a set $S = \{s_1, \ldots, s_n\}$ and a collection $\mathcal{C}$ of subsets of S such that all sets in $\mathcal{C}$ are mutually disjoint, the collection $\mathcal{C}$ has the C1P.*

We say that a set $S' \subseteq S$ has the *minimum overlap property* if each set in $\mathcal{C}_{blue}$ contains at least one element from S'. (The term "minimum overlap property" expresses that the solution S' must have an "overlap" of at least one element with every set of $\mathcal{C}_{blue}$.) Moreover, for a given instance $(S, \mathcal{C}_{blue}, \mathcal{C}_{red}, k)$ of MDH and RBSC, we will call k the *maximum overlap* and the *maximum*

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

intersection, respectively. A set S' has the *maximum overlap property* if each set in $\mathcal{C}_{red}$ contains at most k elements from S'. Analogously, a set S' has the *maximum intersection property* if at most k sets in $\mathcal{C}_{red}$ contain elements from S'.

It is easy to see that the problems considered in this chapter are all in NP: just guess a subset $S' \subseteq S$ and check in polynomial time whether it is a solution. Therefore, all our NP-completeness proofs will only show the NP-hardness of the corresponding problems.

We continue with two observations concerning MDH without C1P. Being a generalization of Set Cover, MDH is of course NP-hard in general. This even holds for a rather strongly restricted variant:

Observation 5.2. MDH *is* NP*-complete even if* $|\mathcal{C}_{red}| = 1$ *and* $\forall C \in \mathcal{C}_{blue} : |C| = 2$.

The observation can be seen by a reduction from the NP-complete Vertex Cover problem: Given a graph $G = (V, E)$ and a nonnegative integer k, this problem asks for a size-k subset $V' \subseteq V$ such that for every edge in E, at least one of its endpoints is in V' (see Section 1.3). Given an instance (G, k) of Vertex Cover, construct an instance of MDH by setting $S := V$, $\mathcal{C}_{blue} := E$, $\mathcal{C}_{red} := \{V\}$ (that is, the collection $\mathcal{C}_{red}$ consists of one set containing all elements of S), and setting the maximum overlap equal to k. The correctness of this construction is straightforward.

Polynomial-time solvable instances of MDH arise when the cardinalities of all sets in the collection $\mathcal{C}_{blue}$ are restricted to 2 and the maximum overlap k is 1:

Observation 5.3. MDH *can be solved in polynomial time if* $k = 1$ *and* $\forall C \in \mathcal{C}_{blue} : |C| \leq 2$.

This observation can be shown by stating the restricted MDH instance equivalently as an instance of the 2-Sat problem, where, given a Boolean formula F in conjunctive normal form with at most two literals per clause, the question is whether there is a satisfying truth assignment for F. 2-Sat is well-known to be solvable in linear time [APT79]. For the reduction, construct the following instance F of 2-Sat for a given instance $(S, \mathcal{C}_{blue}, \mathcal{C}_{red}, 1)$ of MDH:

- For each element $s_i \in S$, where $1 \leq i \leq n$, F contains the variable x_i.
- For each set $\{s_{i_1}, s_{i_2}\} \in \mathcal{C}_{blue}$, F contains the clause $(x_{i_1} \vee x_{i_2})$.
- For each set $\{s_{i_1}, \ldots, s_{i_d}\} \in \mathcal{C}_{red}$, F contains $d(d-1)/2$ clauses $(\neg x_{i_a} \vee \neg x_{i_b})$ with $1 \leq a < b \leq d$.

The correctness of the reduction is obvious: Every satisfying truth assignment for F yields a solution for MDH by setting $S' = \{s_i \mid x_i \text{ is set to } \textit{true}\}$. Due to the construction of the clauses, the set S' contains at least one element from every set $\{s_{i_1}, s_{i_2}\} \in \mathcal{C}_{blue}$ and at most one element from every set $\{s_{i_1}, \ldots, s_{i_d}\} \in \mathcal{C}_{red}$:

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

if S' contained two elements s_{i_a}, s_{i_b} from a set $\{s_{i_1}, \ldots, s_{i_d}\} \in \mathcal{C}_{red}$, then the clause $(\neg x_{i_a} \vee \neg x_{i_b})$ would not evaluate to *true*. Similarly, every solution S' for the MDH instance yields a satisfying truth assignment for F by setting every variable to *true* that corresponds to an element in S'.

Corollary 5.1. MDH *can be solved in polynomial time if* $\forall C \in \mathcal{C}_{blue} \cup \mathcal{C}_{red} : |C| \leq 2$.

To see this, first note that if $k \geq 2$ then the corresponding MDH instance is trivially solvable by setting $S' := S$, because then no set in $\mathcal{C}_{red}$ has more than k elements in common with the solution set S'. Hence, we only need to deal with the case $k = 1$, for which the claim is true by Observation 5.3.

Note that the restrictions imposed by Observation 5.3 and Corollary 5.1 are "tight." If we allow $\mathcal{C}_{blue}$ to contain cardinality-3 subsets, then MDH becomes NP-complete (Theorems 5.5 and 5.7). If $\mathcal{C}_{red}$ contains cardinality-3 subsets and the maximum overlap is 2, then we can also prove the NP-completeness (Theorems 5.6 and 5.8).

5.3 Minimum-Degree Hypergraph and Red-Blue Set Cover with the C1P

In this section, we require that in all instances $(S, \mathcal{C}_{blue}, \mathcal{C}_{red}, k)$ of MDH or RBSC, the set $\mathcal{C} := \mathcal{C}_{blue} \cup \mathcal{C}_{red}$ has the C1P. We call these restricted problems variants "MDH with C1P" and "RBSC with C1P," respectively.

By using known linear programming techniques, MDH with C1P can be solved in polynomial time; we will describe this approach in Section 5.3.1, followed by a much simpler greedy approximation algorithm in Section 5.3.2. The polynomial-time solvability of RBSC with C1P is more difficult to see; for this problem, we will present an exact polynomial-time dynamic programming algorithm in Section 5.3.3.

To simplify our subsequent considerations, we assume that the elements in $S = \{s_1, \ldots, s_n\}$ are sorted according to the C1P. This sorting can be done in $O(|S| + |\mathcal{C}| + \sum_{C \in \mathcal{C}} |C|)$ time (see Section 2.3). For each subset $C \in \mathcal{C}_{red} \cup \mathcal{C}_{blue}$, its *left index* $\mathrm{lx}(C)$ is defined as $\min\{i \mid s_i \in C\}$ and its *right index* $\mathrm{rx}(C)$ is defined as $\max\{i \mid s_i \in C\}$.

5.3.1 Linear Programming for Minimum-Degree Hypergraph

Here, we give a formulation of MDH with C1P as an integer linear program (ILP). By using the techniques explained in Section 2.4, this ILP can be solved in polynomial time. Again, we again refer to Schrijver [Sch86] for basics about (integer) linear programming.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Given an instance of MDH with the C1P, we introduce for each element $s_i \in S$ a variable x_i which, if set to 1, expresses that s_i has to be part of an optimal solution. Every feasible solution for the following integer linear program (ILP) then obviously yields a solution for MDH with C1P:

$$\begin{aligned} -x_{\mathrm{lx}(C)} - x_{\mathrm{lx}(C)+1} - \ldots - x_{\mathrm{rx}(C)} &\leq -1 && \forall C \in \mathcal{C}_{blue} \\ x_{\mathrm{lx}(C)} + x_{\mathrm{lx}(C)+1} + \ldots + x_{\mathrm{rx}(C)} &\leq k && \forall C \in \mathcal{C}_{red} \\ x_i &\in \{0,1\} && \forall i \in \{1,\ldots,|S|\} \end{aligned}$$

Note that the coefficient matrix of this ILP has the C1P, that is, every row of the matrix contains only either 0s and 1s or 0s and -1s, and in every row the non-zero entries appear consecutively. Therefore, as stated in Section 2.4, the ILP is totally unimodular (Theorem 2.10), and, thus, can be solved in polynomial time (Theorem 2.11).

In order to solve the ILP faster than by only using the fact that its coefficient matrix is totally unimodular, one can exploit the C1P by transforming the ILP into an edge-weighted graph and solving a shortest path problem on this graph as shown in Section 2.4. For applying this approach to the ILP above, take the relaxation of the ILP, that is, replace the constraints $x_i \in \{0,1\}\ \forall i \in \{1,\ldots,|S|\}$ by $-x_i \leq 0\ \forall i \in \{1,\ldots,|S|\}$ and $x_i \leq 1\ \forall i \in \{1,\ldots,|S|\}$. Then, replace the n variables $x_1,\ldots,x_n$ by $n+1$ variables $y_0,\ldots,y_n$ such that $x_i = y_i - y_{i-1}$ for all $i \in \{1,\ldots,n\}$, which yields the following inequation system.

$$\begin{aligned} y_{\mathrm{lx}(C)-1} - y_{\mathrm{rx}(C)} &\leq -1 && \forall C \in \mathcal{C}_{blue} \\ -y_{\mathrm{lx}(C)-1} + y_{\mathrm{rx}(C)} &\leq k && \forall C \in \mathcal{C}_{red} \\ -y_i + y_{i-1} &\leq 0 && \forall i \in \{1,\ldots,|S|\} \\ y_i - y_{i-1} &\leq 1 && \forall i \in \{1,\ldots,|S|\} \end{aligned}$$

Now interpret every row of this inequation system as a directed edge in an edge weighted graph G, which yields the directed edge-weighted graph $G = (V, E)$ with

$$\begin{aligned} V &= \{v_i \mid \text{ the ILP contains a variable } y_i\}, \\ E &= \{(v_i, v_j) \mid \text{ the ILP contains an inequation whose left side is } -y_i + y_j\}, \end{aligned}$$

where every edge $e \in E$ has a weight that is equal to the right side of the inequation corresponding to e in the ILP.

Farkas' Lemma (Lemma 2.2) states that if A is an $m \times n$ matrix with entries from $\mathbb{R}$ and $\vec{b} \in \mathbb{R}^m$ is a vector, then the inequation system $A\vec{y} \leq \vec{b}$ has a solution $\vec{y} \in \mathbb{R}^n$ iff the inequation system $\vec{z}^{\,\mathrm{T}}A = (0^n)^{\mathrm{T}}, \vec{z}^{\,\mathrm{T}}\vec{b} < 0, \vec{z} \geq 0^m$ has *no* solution $\vec{z} \in \mathbb{R}^m$. This implies that the given MDH instance is a *yes*-instance iff G contains no directed cycle whose edge weight sum is negative. By using the Bellmann-Ford-Moore-Algorithm [CLRS01], it can be decided in $O(|V| \cdot |E|)$ time if G contains such a negative cycle, and, hence, MDH with C1P can be decided in $O(|S| \cdot (|\mathcal{C}_{blue}| + |\mathcal{C}_{red}| + 2 \cdot |S|)) = O(|S|^3)$ time.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

If G contains no cycle with negative edge weight sum and the values of the y_i shall be computed, then just set y_0 to 0 and y_i, $i \in \{1, \ldots, n\}$, to the length of the shortest path in G from v_0 to v_i, which can be computed by the Bellmann-Ford-Moore-Algorithm in $O(|S|^3)$ time.

Altogether, we summarize our observations in the following theorem.

Theorem 5.1. MINIMUM-DEGREE HYPERGRAPH *can be solved in* $O(|S| \cdot (|\mathcal{C}_{blue}| + |\mathcal{C}_{red}| + |S|)) = O(|S|^3)$ *time if* $\mathcal{C}_{blue} \cup \mathcal{C}_{red}$ *has the C1P.*

5.3.2 Greedy Algorithm for Minimum-Degree Hypergraph

As we have seen, MDH with C1P can be solved in polynomial time with an ILP approach. By way of contrast, here we describe a simple greedy algorithm for MDH with C1P that has an absolute approximation guarantee of an additive term "+1." To this end, we consider the optimization version of MDH: Given S, $\mathcal{C}_{blue}$, and $\mathcal{C}_{red}$, find a subset $S' \subseteq S$ with $S' \cap C \neq \emptyset$ for all $C \in \mathcal{C}_{blue}$ which minimizes $\max\{|C' \cap S'| \mid C' \in \mathcal{C}_{red}\}$.

The idea of the greedy algorithm is to search in each step for the set $C \in \mathcal{C}_{blue}$ with the leftmost right index $\mathrm{rx}(C)$ such that no element of C is contained in the current solution set, and to add the rightmost element of C to the solution:

1: $S' := \emptyset$; $\mathcal{C}'_{blue} := \mathcal{C}_{blue}$;
2: **while** $\mathcal{C}'_{blue} \neq \emptyset$: {
3: $C :=$ set from $\mathcal{C}'_{blue}$ with minimum right index;
4: $S' := S' \cup \{s_{\mathrm{rx}(C)}\}$;
5: $\mathcal{C}'_{blue} := \mathcal{C}'_{blue} \setminus \{C \in \mathcal{C}'_{blue} : C \cap S' \neq \emptyset\}$; }
6: **return** S';

Theorem 5.2. *For* MDH *with C1P, the greedy algorithm approximates an optimal solution within an additive term of one in* $O(|S| \cdot |\mathcal{C}_{blue}|)$ *time, provided that the elements in* S *are sorted such that all subsets in* $\mathcal{C}_{blue}$ *have the C1P.*

Proof. Obviously, the output S' of the greedy algorithm has the minimum overlap property. It is also clear that the algorithm runs in $O(|S| \cdot |\mathcal{C}_{blue}|)$ time. It remains to determine $\max\{|C' \cap S'| \mid C' \in \mathcal{C}_{red}\}$.

Let $C_{\max}$ denote one subset in $\mathcal{C}_{red}$ with $|C_{\max} \cap S'| = \max\{|C' \cap S'| \mid C' \in \mathcal{C}_{red}\}$. Due to the C1P, all sets C chosen in step 3 are pairwise disjoint, and, hence, the set $C_{\max}$ contains at least $|C_{\max} \cap S'| - 1$ pairwise disjoint sets from $\mathcal{C}_{blue}$ as subsets. This implies that *any* solution for this instance has to contain at least $|C_{\max} \cap S'| - 1$ elements from $C_{\max}$ to satisfy the minimum overlap property for these pairwise disjoint $\mathcal{C}_{blue}$-sets. Therefore, $|C_{\max} \cap S'_{\mathrm{opt}}| \geq |C_{\max} \cap S'| - 1$ for any optimal solution S'_{opt}. □

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

5.3.3 Dynamic Programming for Red-Blue Set Cover

In the case of RBSC with C1P, we do not know an ILP formulation whose coefficient matrix is totally unimodular. We present a polynomial-time dynamic programming algorithm that solves the optimization version of Red-Blue Set Cover with C1P: Given S, $\mathcal{C}_{blue}$, and $\mathcal{C}_{red}$, find a subset $S' \subseteq S$ with $S' \cap C \neq \emptyset$ for all $C \in \mathcal{C}_{blue}$ which minimizes $|\{C \in \mathcal{C}_{red} \mid S' \cap C \neq \emptyset\}|$.

We assume that the sets in $\mathcal{C}_{blue}$ are ordered according to their left indices and denote them with $B_1, \ldots, B_{|\mathcal{C}_{blue}|}$; the sets of $\mathcal{C}_{red}$ are ordered analogously and denoted with $R_1, \ldots, R_{|\mathcal{C}_{red}|}$. If a set in $\mathcal{C}_{blue}$ is a superset of another set in $\mathcal{C}_{blue}$, it can be removed. Therefore, for any two sets $B_i, B_j \in \mathcal{C}_{blue}$ it holds that

$$\mathrm{lx}(B_i) < \mathrm{lx}(B_j) \Leftrightarrow \mathrm{rx}(B_i) < \mathrm{rx}(B_j).$$

Given a subset $S' \subseteq S$, we denote with $w(S')$ the number of sets from $\mathcal{C}_{red}$ that are covered by S'.

The idea of the dynamic programming algorithm is to compute so-called *optimal partial solutions* $S_{\mathrm{opt}}(i_1, i_2, j)$. Each optimal partial solution $S_{\mathrm{opt}}(i_1, i_2, j)$ has the following properties:

1. $S_{\mathrm{opt}}(i_1, i_2, j) \subseteq \{s_1, \ldots, s_{i_1}\}$,
2. $S_{\mathrm{opt}}(i_1, i_2, j)$ covers all sets $B_1, \ldots, B_j$,
3. if $i_2 > 0$, then $S_{\mathrm{opt}}(i_1, i_2, j)$ contains at least one element from $\{s_{i_2}, \ldots, s_n\}$ (where $n := |S|$), and
4. the cost $w(S_{\mathrm{opt}}(i_1, i_2, j))$ is minimum under all subsets of S that have the first three properties.

A subset of S that has the first three properties is called a *feasible partial solution*.

The algorithm uses a three-dimensional table $S_{\mathrm{opt}}(i_1, i_2, j)$ with $1 \leq i_1 \leq n$, $0 \leq i_2 \leq n$, and $1 \leq j \leq |\mathcal{C}_{blue}|$ for storing optimal partial solutions, and a table $W_{\mathrm{opt}}(i_1, i_2, j)$ of the same size where the cost of every optimal partial solution is stored. Then, the entry $S_{\mathrm{opt}}(n, 0, |\mathcal{C}_{blue}|)$ contains an optimal solution for the RBSC instance.

The two tables are filled with three nested loops, iterating over i_1, i_2, and j. To compute table entries $S_{\mathrm{opt}}(i_1, i_2, j), W_{\mathrm{opt}}(i_1, i_2, j)$ with $i_1 = 1$ is simple. All other entries are computed as follows: If $\mathrm{lx}(B_j) > i_1$ or $i_2 > i_1$, then there is no partial solution $S_{\mathrm{opt}}(i_1, i_2, j)$, and $W_{\mathrm{opt}}(i_1, i_2, j)$ is set to ∞. Otherwise, we consider two cases: the optimal partial solution contains s_{i_1} or not. (Note that if $i_2 = i_1$, then all feasible partial solutions have to contain s_{i_1}.) In the first case, the optimal partial solution $S_{\mathrm{opt}}(i_1, i_2, j)$ can only contain elements from $\{s_1, \ldots, s_{i_1-1}\}$, and, hence, $S_{\mathrm{opt}}(i_1, i_2, j) = S_{\mathrm{opt}}(i_1 - 1, i_2, j)$. In the second case, the optimal partial solution $S_{\mathrm{opt}}(i_1, i_2, j)$ is computed as follows: By choosing s_{i_1}, property 3 is clearly

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

1: **if** $(\mathrm{lx}(B_j) > i_1) \vee (i_2 > i_1)$: {
2: $W_{\mathrm{opt}}(i_1, i_2, j) := \infty$; $S_{\mathrm{opt}}(i_1, i_2, j) := \emptyset$; **break**; }
3: $W_{\mathrm{opt}}(i_1, i_2, j) := W_{\mathrm{opt}}(i_1 - 1, i_2, j)$; // Lines 3–4: partial solution not containing s_{i_1}
4: $S_{\mathrm{opt}}(i_1, i_2, j) := S_{\mathrm{opt}}(i_1 - 1, i_2, j)$;
5: $j' := \max\{p \in \{1, \ldots, j\} \mid \mathrm{rx}(B_p) < i_1\}$; // Lines 5–14: partial sol. containing s_{i_1}
6: $x := W_{\mathrm{opt}}(i_1 - 1, 0, j') + |\mathcal{C}_{red}(i_1)|$;
7: **if** $x < W_{\mathrm{opt}}(i_1, i_2, j)$: {
8: $W_{\mathrm{opt}}(i_1, i_2, j) := x$;
9: $S_{\mathrm{opt}}(i_1, i_2, j) := S_{\mathrm{opt}}(i_1 - 1, 0, j') \cup \{s_{i_1}\}$; }
10: **for** $k := 1$ **to** $|\mathcal{C}^{\leftarrow}_{red}(i_1)|$: {
11: $x := W_{\mathrm{opt}}(i_1 - 1, \mathrm{lx}(R^{\leftarrow}(i_1, k)), j') + |\mathcal{C}_{red}(i_1)| - k$;
12: **if** $x < W_{\mathrm{opt}}(i_1, i_2, j)$: {
13: $W_{\mathrm{opt}}(i_1, i_2, j) := x$;
14: $S_{\mathrm{opt}}(i_1, i_2, j) := S_{\mathrm{opt}}(i_1 - 1, \mathrm{lx}(R^{\leftarrow}(i_1, k)), j') \cup \{s_{i_1}\}$; }}

Figure 5.1: Dynamic programming algorithm for RBSC. The procedure computes an optimal partial solution $S_{\mathrm{opt}}(i_1, i_2, j)$ and its cost $W_{\mathrm{opt}}(i_1, i_2, j)$ for $i_1 > 1$.

obtained because we can assume that $i_2 \leq i_1$. Moreover, all sets in $\mathcal{C}_{blue}$ that contain s_{i_1} are covered by s_{i_1}. Therefore, to obtain property 2, it remains to cover those sets $B_p \in \{B_1, \ldots, B_j\}$ that have $\mathrm{rx}(B_p) < i_1$. Hence, adding s_{i_1} to an optimal partial solution $S_{\mathrm{opt}}(i_1 - 1, i_2', j')$, where i_2' is chosen from $\{0, \ldots, i_2\}$ and j' is the maximum possible index such that $\mathrm{rx}(B_{j'}) < i_1$, yields an optimal partial solution $S_{\mathrm{opt}}(i_1, i_2, j)$. The value for i_2' has to be chosen such that $W(i_1, i_2, j) = W(i_1 - 1, i_2', j') + |\mathcal{C}_{red}(i_1)| - |X|$ is minimum, where $\mathcal{C}_{red}(i_1)$ denotes the sets from $\mathcal{C}_{red}$ that are covered by s_{i_1} and X denotes the sets from $\mathcal{C}_{red}$ that are covered by both s_{i_1} and $S_{\mathrm{opt}}(i_1 - 1, i_2', j')$.

Before showing the details of our algorithm and proving its correctness, we introduce some more notations:

$$\mathcal{C}_{red}(i) := \{C \in \mathcal{C}_{red} \mid s_i \in C\}, 1 \leq i \leq n, \text{ and}$$
$$\mathcal{C}^{\leftarrow}_{red}(i) := \{C \in \mathcal{C}_{red} \mid s_i \in C \wedge s_{i-1} \in C\}, 1 < i \leq n.$$

With $R^{\leftarrow}(i, k)$ we denote the kth set from $\mathcal{C}^{\leftarrow}_{red}(i)$, where we assume that the sets $C \in \mathcal{C}^{\leftarrow}_{red}(i)$ are ordered according to $\mathrm{lx}(C)$. The pseudocode in Figure 5.1 shows how an optimal partial solution $S_{\mathrm{opt}}(i_1, i_2, j)$ together with its cost $W_{\mathrm{opt}}(i_1, i_2, j)$ is computed for $i_1 > 1$.

Theorem 5.3. RBSC *can be solved in* $O(|\mathcal{C}_{blue}| \cdot |\mathcal{C}_{red}| \cdot |S|^2)$ *time if* $\mathcal{C}_{blue} \cup \mathcal{C}_{red}$ *has the C1P.*

Proof. We show the correctness of the pseudocode shown in Figure 5.1. In lines 3–4 the algorithm searches for an optimal partial solution $S_{\mathrm{opt}}(i_1, i_2, j)$ that

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

does not contain s_{i_1}. Lines 5–14 handle the case that the optimal partial solution $S_{\text{opt}}(i_1, i_2, j)$ contains s_{i_1}. Clearly the procedure outputs a feasible partial solution, and it is easy to verify that the value of $W_{\text{opt}}(i_1, i_2, j)$ computed by the procedure upper-bounds the cost of the partial solution $S_{\text{opt}}(i_1, i_2, j)$ computed by the procedure. It remains to show that the value of $W_{\text{opt}}(i_1, i_2, j)$ computed by the procedure actually equals the cost of an optimal partial solution in the case that the optimal partial solution contains s_{i_1}. To this end, let $S_{\text{opt}}(i_1, i_2, j)$ be an optimal partial solution where $s_{i_1} \in S_{\text{opt}}(i_1, i_2, j)$. Moreover, let $S' := S_{\text{opt}}(i_1, i_2, j) \setminus \{s_{i_1}\}$, let $j' := \max\{p \in \{1, \ldots, j\} \mid \text{rx}(B_p) < i_1\}$, and let $i' := \max\{q \in \{1, \ldots, n\} \mid s_q \in S'\}$. We distinguish two cases.

Case 1: For all $C \in \mathcal{C}_{red}(i_1)$ it holds that $s_{i'} \notin C$. Then $W_{\text{opt}}(i_1, i_2, j) = w(S') + |\mathcal{C}_{red}(i_1)|$. The set S' must have the following properties: S' consists of elements from $\{s_1, \ldots, s_{i_1-1}\}$, and S' covers all sets $B_1, \ldots, B_{j'}$. Under all subsets of S having these two properties, the set $S_{\text{opt}}(i_1 - 1, 0, j')$ is, by definition, the one with minimum cost, and, hence, choosing $S_{\text{opt}}(i_1, i_2, j) = S_{\text{opt}}(i_1 - 1, 0, j') \cup \{s_{i_1}\}$ is optimal. In this case, the procedure finds the correct value of $W_{\text{opt}}(i_1, i_2, j)$ in lines 6–9.

Case 2: There exists a $k \in \{1, \ldots, |\mathcal{C}^{\leftarrow}_{red}(i_1)|\}$ such that $R^{\leftarrow}(i_1, k) \cap S' \neq \emptyset$. We assume that k is maximum under this property. Due to the order of the sets in $\mathcal{C}_{red}$, we have $R^{\leftarrow}(i_1, k') \cap S' \neq \emptyset$ for all $k' < k$, and, hence, $W_{\text{opt}}(i_1, i_2, j) = w(S') + |\mathcal{C}_{red}(i_1)| - k$. The set S' must have the following properties: S' consists of elements from $\{s_1, \ldots, s_{i_1-1}\}$, and S' covers all sets $B_1, \ldots, B_{j'}$. Moreover, the maximum index i' of an element in S' has to satisfy $i' \geq \text{lx}(R^{\leftarrow}(i_1, k))$, because otherwise $R^{\leftarrow}(i_1, k)$ would not be covered by S'. Under all subsets of S having these three properties, the set $S_{\text{opt}}(i_1 - 1, \text{lx}(R^{\leftarrow}(i_1, k)), j')$ is the one with minimum cost, and, hence, choosing $S_{\text{opt}}(i_1, i_2, j) := S_{\text{opt}}(i_1 - 1, \text{lx}(R^{\leftarrow}(i_1, k)), j') \cup \{s_{i_1}\}$ is optimal. In this case, the procedure finds the correct value of $W_{\text{opt}}(i_1, i_2, j)$ in lines 10–14.

It remains to show the running time. The table size is $O(|S|^2 \cdot |\mathcal{C}_{blue}|)$, and to compute an entry of the table, at most $O(|\mathcal{C}_{red}|)$ other entries have to be considered in lines 10–14. Line 5 can be executed in constant time if in a preprocessing step (which can be implemented similar to bucket sort and needs $O(|\mathcal{C}_{blue}| + |S|)$ time) for every possible value of i_1 the corresponding value of j' is computed and stored in an extra table. This yields the claimed running time. □

5.4 Minimum-Degree Hypergraph and Red-Blue Set Cover with Partial C1P

Whereas the C1P case always leads to polynomial-time solvability, in case of only partially holding C1Ps we typically face NP-hardness as shown in this section.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

5.4.1 Minimum-Degree Hypergraph with Partial C1P

Here, we prove that MDH remains NP-complete even under the requirement that either $\mathcal{C}_{blue}$ or $\mathcal{C}_{red}$ is to have the C1P. To this end, we give reductions from the following restricted variant of the SATISFIABILITY problem:

RESTRICTED 3-SAT (R-3SAT)
Input: An n-variable, m-clause Boolean formula F in conjunctive normal form where each variable x_i, $1 \leq i \leq n$, appears at most three times, each literal appears at most twice, and each clause contains at most three literals.
Question: Is there a satisfying truth assignment for F?

R-3SAT is NP-complete (see, for example, [Pap94, page 183]).[3] Without loss of generality, we assume that no variable appears in F solely positively or negatively, and F contains no singleton clause.

Our reductions show the NP-completeness of MINIMUM-DEGREE HYPERGRAPH variants that have several further restrictions apart from the C1P for $\mathcal{C}_{blue}$ or $\mathcal{C}_{red}$. To emphasize the correlation between the hardness of the problem and the value of k and the subset sizes in $\mathcal{C}_{blue}$ and $\mathcal{C}_{red}$, we summarize some of the results in the following statement, which is a corollary of Observations 5.2 and 5.3, Corollary 5.1, and Theorems 5.5, 5.6, 5.7, and 5.8.

Theorem 5.4. MDH *is* NP*-complete even if all of the following restrictions apply:*

1. *One of the collections $\mathcal{C}_{blue}$ and $\mathcal{C}_{red}$ has the consecutive-ones property,*
2. *$k = 1$, and*
3. *$\forall C \in \mathcal{C}_{blue} : |C| \leq 3$ and $\forall C \in \mathcal{C}_{red} : |C| \leq 2$.*

However, replacing restriction 2 by $k = 0$, replacing restriction 3 by $\forall C \in \mathcal{C}_{blue} : |C| \leq 2$, or replacing restriction 3 by $\forall C \in \mathcal{C}_{red} : |C| \leq 1$ leads to polynomial-time solvability.

MDH *is* NP*-complete even if all of the following restrictions apply:*

1. *One of the collections $\mathcal{C}_{blue}$ and $\mathcal{C}_{red}$ has the consecutive-ones property,*
2. *$k = 2$, and*
3. *$\forall C \in \mathcal{C}_{blue} : |C| \leq 2$ and $\forall C \in \mathcal{C}_{red} : |C| \leq 3$.*

However, replacing restriction 2 by $k \leq 1$, replacing restriction 3 by $\forall C \in \mathcal{C}_{blue} : |C| \leq 1$, or replacing restriction 3 by $\forall C \in \mathcal{C}_{red} : |C| \leq 2$ leads to polynomial-time solvability.

[3] It is essential for the NP-completeness of R-3SAT that the Boolean formula F may contain size-2 clauses, otherwise, the problem is solvable in polynomial time [Pap94, page 207].

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Consecutive-Ones Property for $\mathcal{C}_{blue}$

The following two theorems (Theorems 5.5 and 5.6) show that the requirement of $\mathcal{C}_{blue}$ obeying the C1P does not make MDH tractable. The theorems complement each other in the sense that they impose different restrictions on the cardinalities of the sets $\mathcal{C}_{blue}$ and $\mathcal{C}_{red}$; Theorem 5.5 needs size-3 sets in $\mathcal{C}_{blue}$ and size-2 sets in $\mathcal{C}_{red}$ (the reduction encodes clauses of a given R-3SAT instance in $\mathcal{C}_{blue}$) while the converse holds true for Theorem 5.6 (the reduction encodes clauses in $\mathcal{C}_{red}$).

Theorem 5.5. MDH *is* NP*-complete even if all of the following restrictions apply:*

1. *The collection* $\mathcal{C}_{blue}$ *has the consecutive-ones property,*
2. $k = 1$*,*
3. $\forall C \in \mathcal{C}_{blue} : |C| \leq 3$ *and* $\forall C \in \mathcal{C}_{red} : |C| \leq 2$*, and*
4. $\forall s \in S : |\{C \in \mathcal{C}_{blue} \mid s \in C\}| = 1$ *and* $|\{C \in \mathcal{C}_{red} \mid s \in C\}| \leq 2$*.*

Proof. We prove the theorem by a reduction from R-3SAT. Given an m-clause Boolean formula F that is an instance of R-3SAT, construct the following instance $(S, \mathcal{C}_{blue}, \mathcal{C}_{red}, k)$ of MDH:

- The set S consists of the elements $s_1^1, s_1^2, s_1^3, \ldots, s_m^1, s_m^2, s_m^3$. The element s_j^i corresponds to the ith literal in the jth clause of F. If the jth clause has only two literals, then S contains only s_j^1 and s_j^2.
- Each set in $\mathcal{C}_{blue}$ corresponds to a clause in F, that is, for the ith clause in F, we add $\{s_i^1, s_i^2, s_i^3\}$ to $\mathcal{C}_{blue}$ if it contains three literals and $\{s_i^1, s_i^2\}$ if it contains two literals.
- For all variables x and for all pairs of literals $l_1 = x, l_2 = \neg x$ in F: If l_1 is the ith literal in the jth clause and l_2 is the pth literal in the qth clause of F, then $\mathcal{C}_{red}$ contains the set $\{s_j^i, s_q^p\}$.
- The maximum overlap k is set to one.

The construction is illustrated in Figure 5.2. It is easy to see that, by the definition of R-3SAT, the constructed instance satisfies the restrictions claimed in the theorem; note that $\mathcal{C}_{blue}$ has the consecutive-ones property due to Observation 5.1. It remains to be shown that the constructed instance of MDH has a solution iff F has a satisfying truth assignment T.

"$\Rightarrow$" Assume that the constructed instance of MDH has a solution set S'. Let T be a truth assignment such that, for every $s_j^i \in S'$, the variable represented by s_j^i is set to *true* if the literal represented by s_j^i is positive, and *false* otherwise. This truth assignment is well defined because S' must have the maximum overlap

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

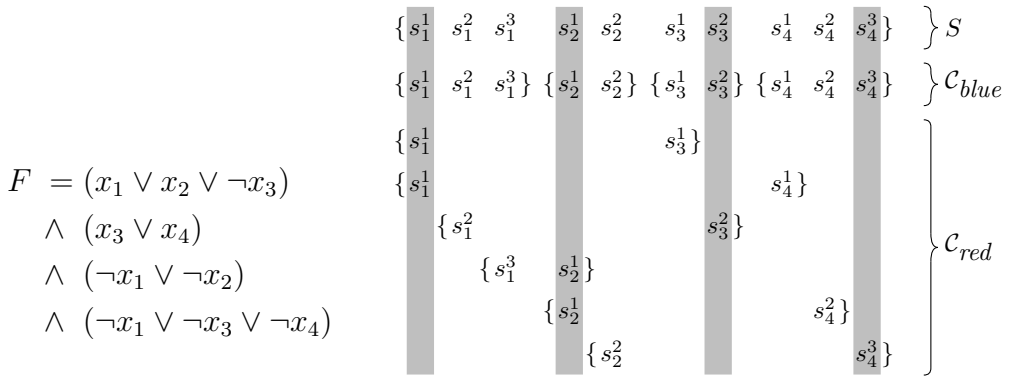


Figure 5.2: Example of encoding an instance of R-3Sat into an instance of MDH (proof of Theorem 5.5). Each clause of the Boolean formula F is represented by a set in $\mathcal{C}_{blue}$. The sets in $\mathcal{C}_{red}$ and the maximum overlap $k = 1$ ensure that no two elements from S that correspond to conflicting truth assignments of the same variable can be chosen into a solution. Observe how $S' = \{s_1^1, s_2^1, s_3^2, s_4^3\}$ (grey columns) constitutes a valid solution to the MDH instance; accordingly, a truth assignment T which makes all the corresponding literals evaluate to *true* satisfies F.

property with $k = 1$—therefore, it cannot happen that two elements $s_j^i, s_q^p \in S'$ correspond to different literals of the same variable.

To show that T constitutes a satisfying truth assignment for F, observe that, for each clause of F, at least one element from S' corresponds to a literal in this clause because S' has the minimum overlap property. On the one hand, if this element corresponds to a positive literal x_i, then $T(x_i) = \mathit{true}$, satisfying the clause. On the other hand, if the element corresponds to a negative literal $\neg x_i$, then $T(x_i) = \mathit{false}$, satisfying the clause.

"$\Leftarrow$" Let T be a satisfying truth assignment for F. Let S' be the set of elements in S that correspond to literals that evaluate to *true* under T. Then, S' has the minimum overlap property because at least one literal in every clause of F must evaluate to true under T and each set in $\mathcal{C}_{blue}$ represents exactly one clause of F. Also, S' has the maximum overlap property with $k = 1$ because T is well-defined for every variable that occurs in F. Since S' has both the minimum and maximum overlap property, it is a valid solution to the MDH instance. □

Theorem 5.6. MDH *is* NP*-complete even if all of the following restrictions apply:*

1. *The collection $\mathcal{C}_{blue}$ has the consecutive-ones property,*
2. *$k = 2$,*
3. *$\forall C \in \mathcal{C}_{blue} : |C| \leq 2$ and $\forall C \in \mathcal{C}_{red} : |C| \leq 3$, and*

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

4. $\forall s \in S : |\{C \in \mathcal{C}_{blue} \mid s \in C\}| = 1$ *and* $|\{C \in \mathcal{C}_{red} \mid s \in C\}| \leq 2$.

Proof. We prove the theorem by a reduction from R-3SAT. The reduction is similar to the one used in the proof of Theorem 5.5, but this time one uses the sets of $\mathcal{C}_{red}$ instead of those of $\mathcal{C}_{blue}$ to model the clauses of F, and one uses the sets of $\mathcal{C}_{blue}$ to enforce the consistency between literals representing the same variable. Moreover, in contrast to the reduction used in the proof of Theorem 5.5, here each element chosen into the solution set—if a solution exists—stands for a literal that is set to *false* by a satisfying truth assignment for F. Hence, not more than two elements per red set may be chosen into the solution set if the corresponding truth assignment for F shall be satisfying; this is expressed by setting k to two. In order to prevent both literals of a size-2 clause from being set to *false*, we add to each set in $\mathcal{C}_{red}$ corresponding to a size-2 clause a dummy element which has to be part of every solution.

The instance $(S, \mathcal{C}_{blue}, \mathcal{C}_{red}, k)$ of MDH is constructed as follows:

- We set $S := \{s_1, \bar{s}_1, \ldots, s_n, \bar{s}_n\} \cup \{s_1^c, \ldots, s_m^c\}$. Herein, n denotes the number of variables in F and m denotes the number of clauses in F. For a variable x_i in F, s_i represents the literal x_i and $\bar{s}_i$ represents the literal $\neg x_i$. We use the elements s_i^c to ensure that each set in $\mathcal{C}_{red}$ has size three.
- $\mathcal{C}_{blue} := \left(\bigcup_{1 \leq i \leq n} \{\{s_i, \bar{s}_i\}\}\right) \cup \{\{s_1^c\}, \ldots, \{s_m^c\}\}$.
- For each clause c in F, $\mathcal{C}_{red}$ contains a set C of those elements from S that represent the literals of c: If the jth clause in F contains only two literals, then s_j^c is added to its representing set in $\mathcal{C}_{red}$ as the third element.
- The maximum overlap k is set to two.

See Figure 5.3 for an illustration of the construction. Clearly, this MDH instance satisfies all restrictions as claimed by the theorem. The correspondence between the solutions of the constructed instance and the satisfying truth assignments for F follows from the following two observations.

First, if the constructed MDH instance is solvable, then it has always a solution set S' such that, for each variable x_i, exactly one of s_i and $\bar{s}_i$ is in S'. This can easily be seen because if a solution set S' contains both of s_i and $\bar{s}_i$ for a variable x_i, then S' without s_i (or S' without $\bar{s}_i$) is also a solution for the MDH instance. This observation guarantees that we can always construct a well-defined truth assignment for F from S' and vice versa as follows: $T(x_i) = \mathit{true} \Leftrightarrow s_i \notin S'$.

Second, F is satisfiable with a truth assignment T iff every clause of size three has at most two literals that are evaluated to *false* by T and every clause of size two has at most one literal that is evaluated to *false*. By the correspondence between T and S', it is then easy to observe that T satisfies F iff S' fulfills the maximum overlap property with $k = 2$, that is, S' meets, for each clause c, the set in $\mathcal{C}_{red}$ corresponding to c at most twice. □

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

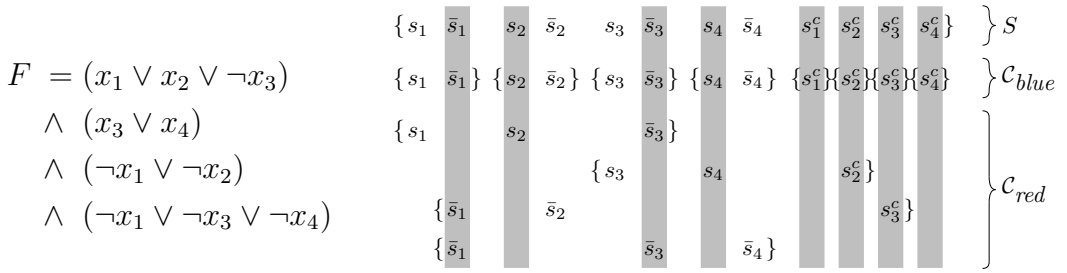


Figure 5.3: Example of encoding an instance of R-3SAT into an instance of MDH (proof of Theorem 5.6). Each clause of the Boolean formula F is represented by a set in $\mathcal{C}_{red}$. The sets in $\mathcal{C}_{blue}$ ensure that for each variable one element of the two elements corresponding to its positive and negative literal is chosen into a solution; the maximum overlap $k = 2$ ensures that for each clause at most two elements corresponding to its literals are chosen. Observe how $S' = \{\bar{s}_1, s_2, \bar{s}_3, s_4, s_1^c, s_2^c, s_3^c, s_4^c\}$ (grey columns) constitutes a valid solution to the MDH instance; accordingly, a truth assignment T with $T(x_i) =$ *true* iff $s_i \notin S'$ satisfies F.

Consecutive-Ones Property for $\mathcal{C}_{red}$

Note that by the reduction from VERTEX COVER in Section 5.2, MDH is NP-complete already if $\mathcal{C}_{red}$ contains just a single set and, hence, has the C1P. However, this requires an unrestricted maximum overlap k and an unrestricted cardinality of the (single) set contained in $\mathcal{C}_{red}$. Therefore, if we want to show the NP-completeness of MDH with the additional restriction that the maximum overlap k is fixed and the sets in $\mathcal{C}_{blue}$ and $\mathcal{C}_{red}$ have small cardinality, another reduction is needed. Analogously to Theorems 5.5 and 5.6, the following two theorems impose different restrictions on the cardinalities of the sets in $\mathcal{C}_{blue}$ and $\mathcal{C}_{red}$.

Theorem 5.7. MDH *is* NP*-complete even if all of the following restrictions apply:*

1. *The collection* $\mathcal{C}_{red}$ *has the consecutive-ones property,*
2. $k = 1$*,*
3. $\forall C \in \mathcal{C}_{blue} : |C| \leq 3$ *and* $\forall C \in \mathcal{C}_{red} : |C| \leq 2$*, and*
4. $\forall s \in S : |\{C \in \mathcal{C}_{blue} \mid s \in C\}| \leq 2$ *and* $|\{C \in \mathcal{C}_{red} \mid s \in C\}| = 1$*.*

Proof. Again, we give a reduction from R-3SAT. For a n-variable Boolean formula F that is an instance of R-3SAT, construct the following instance $(S, \mathcal{C}_{blue}, \mathcal{C}_{red}, k)$ of MDH:

- The set S is equal to $\{s_1, \bar{s}_1, \ldots, s_n, \bar{s}_n\}$, that is, for each variable x_i in F, S contains an element s_i representing the literal x_i and an element $\bar{s}_i$ representing the literal $\neg x_i$.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

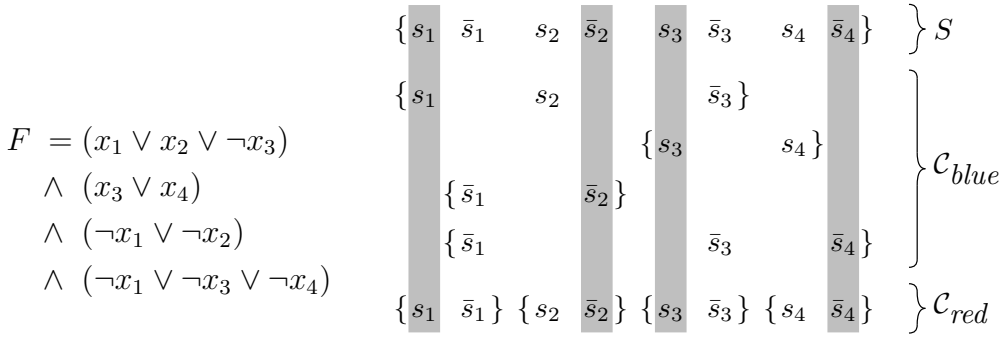


Figure 5.4: Example of encoding an instance of R-3SAT into an instance of MDH (proof of Theorem 5.7). Each clause of the Boolean formula F is encoded into one set of $\mathcal{C}_{blue}$. Observe how $S' = \{s_1, \bar{s}_2, s_3, \bar{s}_4\}$ (grey columns) constitutes a valid solution to the MDH instance; accordingly, a truth assignment T with $T(x_i) =$ *true* iff $s_i \in S'$ satisfies F.

- For each clause in F, $\mathcal{C}_{blue}$ contains a set of those elements from S that represent the literals of that clause.
- $\mathcal{C}_{red} = \bigcup_{1 \leq i \leq n} \{\{s_i, \bar{s}_i\}\}$.
- The maximum overlap k is set to one.

Observe that this MDH instance satisfies all restrictions claimed in the theorem. The reduction is illustrated by an example in Figure 5.4. The correctness of the reduction can be proven in a similar way as in the proof of Theorem 5.5. □

Theorem 5.8. MDH *is* NP*-complete even if all of the following restrictions apply:*

1. *The collection* $\mathcal{C}_{red}$ *has the consecutive-ones property,*
2. $k = 2$*,*
3. $\forall C \in \mathcal{C}_{blue} : |C| \leq 2$ *and* $\forall C \in \mathcal{C}_{red} : |C| \leq 3$*, and*
4. $\forall s \in S : |\{C \in \mathcal{C}_{blue} \mid s \in C\}| \leq 2$ *and* $|\{C \in \mathcal{C}_{red} \mid s \in C\}| = 1$*.*

Proof. The reduction used in this proof is a combination of the reductions used in the proofs of Theorems 5.5 and 5.6: We encode clauses and variables in a similar way as in the proof of Theorem 5.5. But here clauses are encoded in $\mathcal{C}_{red}$ and variables in $\mathcal{C}_{blue}$. As in the proof of Theorem 5.6, each element chosen into the solution set—if one exists—stands for a literal that is set to *false* by a satisfying truth assignment for F.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

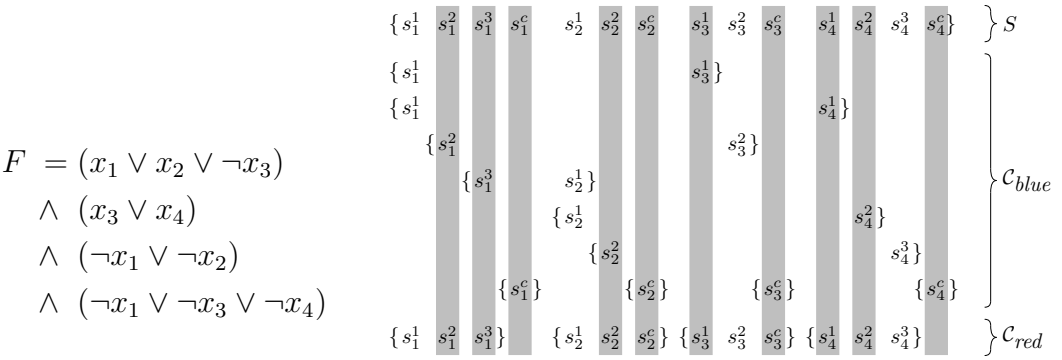


Figure 5.5: Example of encoding an instance of R-3SAT into an instance of MDH (proof of Theorem 5.8). Each clause of the Boolean formula F is represented by a set in $\mathcal{C}_{red}$. Observe how $S' = \{s_1^2, s_1^3, s_1^c, s_2^2, s_2^c, s_3^1, s_3^c, s_4^1, s_4^2, s_4^c\}$ (grey columns) constitutes a valid solution to the MDH instance; accordingly, a truth assignment T which makes all the literals *not* corresponding to one of the chosen elements evaluate to *true* satisfies F.

- We set $S := \{s_1^1, s_1^2, s_1^3, \ldots, s_m^1, s_m^2, s_m^3\} \cup \{s_1^c, \ldots, s_m^c\}$. The element s_j^i represents the ith literal in the jth clause of F. If the jth clause has only two literals, then S contains only s_j^1 and s_j^2. The elements s_i^c are used to ensure that each set in $\mathcal{C}_{red}$ has size three.
- For all variables x in F and for all pairs of literals $l_1 = x, l_2 = \neg x$ in F: If l_1 is the ith literal in the jth clause and l_2 is the pth literal in the qth clause of F, $\mathcal{C}_{blue}$ contains the set $\{s_j^i, s_q^p\}$. Moreover, we add $\{s_i^c\}$ with $1 \leq i \leq m$ to $\mathcal{C}_{blue}$.
- For each clause in F, $\mathcal{C}_{red}$ contains a set of those elements from S that represent the literals of that clause. If the jth clause in F contains only two literals, then s_j^c is added to the corresponding set in $\mathcal{C}_{red}$ as the third element.
- The maximum overlap k is set to two.

An example of the reduction is shown in Figure 5.5. The correctness of the reduction can be proven in a similar way as in the proof of Theorem 5.6. □

5.4.2 Red-Blue Set Cover with Partial C1P

The problem RED-BLUE SET COVER was introduced by Carr et al. [CDKM00]; here we use the problem definition given in Section 5.1: Find a subset $S' \subseteq S$ containing at least one element from each blue set, such that the number of red sets containing elements from S' is at most k, We will show the NP-completeness of RBSC when restricted to instances where the sets in $\mathcal{C}_{blue}$ or the sets in $\mathcal{C}_{red}$ have the C1P.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Theorem 5.9. RBSC *is* NP*-complete even if*

1. $|C| \leq 2$ *for all* $C \in \mathcal{C}_{blue}$, $|C| = 1$ *for all* $C \in \mathcal{C}_{red}$ *(which trivially implies that* $\mathcal{C}_{red}$ *has the consecutive-ones property), and for all* $s \in S$, $|\{C \in \mathcal{C}_{blue} \mid s \in C\}| \leq 3$ *and* $|\{C \in \mathcal{C}_{red} \mid s \in C\}| = 1$*, or*

2. *the collection* $\mathcal{C}_{blue}$ *has the consecutive-ones property,* $|C| \leq 2$ *for all* $C \in \mathcal{C}_{blue}$, $|C| \leq 3$ *for all* $C \in \mathcal{C}_{red}$*, and for all* $s \in S$, $|\{C \in \mathcal{C}_{blue} \mid s \in C\}| = 1$ *and* $|\{C \in \mathcal{C}_{red} \mid s \in C\}| = 1$.

Proof. We show both cases of the theorem by reductions from Vertex Cover restricted to cubic graphs, that is, graphs with maximum vertex degree three. Vertex Cover restricted to cubic graphs is NP-hard [GJ79].

To prove case 1, let $G = (V, E)$ be a cubic graph. For the reduction, set $S := V$, $\mathcal{C}_{blue} := E$, and $\mathcal{C}_{red} := \{\{v\} \mid v \in V\}$. Clearly, the constructed instance satisfies all restrictions of this case. The one-to-one correspondence between the solutions follows directly from the construction.

To show case 2, let $G = (V, E)$ be a cubic graph with $V = \{v_1, v_2, \ldots, v_n\}$ and $E = \{e_1, e_2, \ldots, e_m\}$. Construct the following instance $(S, \mathcal{C}_{blue}, \mathcal{C}_{red}, k)$ of RBSC:

- $S := \{s_l^i, s_l^j \mid e_l = \{v_i, v_j\} \in E\}$, that is, S contains, for every edge e_l, two elements corresponding to e_l's endpoints.
- $\mathcal{C}_{blue} := \{\{s_l^i, s_l^j\} \mid e_l = \{v_i, v_j\} \in E\}$.
- For every vertex $v_i \in V$ we add to $\mathcal{C}_{red}$ a set C_i consisting of the at most three elements "corresponding" to v_i. More precisely, $s_l^i \in C_i$ for every edge e_l that has v_i as one endpoint.

Since the sets in $\mathcal{C}_{blue}$ are pairwise disjoint, $\mathcal{C}_{blue}$ has the consecutive-ones property. The other restrictions of this case are also clearly satisfied.

It is easy to see that G has a vertex cover with at most k vertices iff the constructed RBSC-instance has a solution with maximum intersection k: Given a vertex cover V' of G, the RBSC-instance has a solution $S' := \cup_{v_i \in V'} C_i$; conversely, given a solution S' of the RBSC-instance, the set $V' := \{v_i \mid C_i \cap S' \neq \emptyset, C_i \in \mathcal{C}_{red}\}$ is clearly a size-$\leq k$ vertex cover of G. □

The restriction on the cardinality of $\mathcal{C}_{blue}$-sets in case 1 of Theorem 5.9 is clearly tight: For cardinality-one $\mathcal{C}_{blue}$-sets we have only one choice, that is, taking the element into the solution.

Finally, we mention in passing that our reduction also implies that the optimization version of RBSC as restricted above can only be approximated up to a constant factor unless P = NP, that is, it is APX-hard (and MAXSNP-hard [PY91]). This is due to the fact that the reductions in the proof of Theorem 5.9 are clearly approximation-preserving reductions. Thus, the claim follows from the fact that Vertex Cover restricted to cubic graphs still is MAXSNP-hard [PY91].

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

5.5 Conclusion

We disclosed a sharp border between the polynomial-time solvable and NP-hard special cases of Minimum-Degree Hypergraph and Red-Blue Set Cover. To reduce our findings to a common denominator, one could say that being "close" to having the C1P does not suffice to obtain polynomial-time solvability—except for some special cases with extremely sparse input matrices, both problems MDH and RBSC remain NP-hard when only a part of the input matrix has the C1P. In contrast, for the case where the input matrix has the C1P, we could find polynomial-time algorithms for MDH and RBSC where the polynomials have low degrees ($O(|S| \cdot (|\mathcal{C}_{blue}| + |\mathcal{C}_{red}| + |S|))$ and $O(|\mathcal{C}_{blue}| \cdot |\mathcal{C}_{red}| \cdot |S|^2)$, respectively) and the O-notation does not hide any big constants.

In the case of MDH, our reductions show that the problem is NP-hard even if k is constant, the size of the sets in $\mathcal{C}_{blue}$ and $\mathcal{C}_{red}$ is constant, and every element from S appears only in a constant number of sets from $\mathcal{C}_{blue}$ and $\mathcal{C}_{red}$. Hence, none of these measurements can serve as a parameter for developing fixed-parameter algorithms. However, our reductions do not imply the non-approximability of MDH. Also, the approximability and the parameterized complexity of the variants of Minimum-Degree Hypergraph and Red-Blue Set Cover now proven to be NP-complete could be explored in future research.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Chapter 6

Rectangle Stabbing

This chapter presents a further study of how "almost having the C1P" influences the complexity of a combinatorial problem. The problem considered here is the matrix-based formulation of Set Cover, and "almost having the C1P" means that in every row of the input matrix there are only a small constant number of blocks of 1s. Set Cover, restricted in this way, can also be interpreted as a geometric covering problem called (d-Dimensional) Rectangle Stabbing, which is subject of a number of studies in the literature; our results complement and complete the already known results about this geometric problem. The input for d-Dimensional Rectangle Stabbing consists of a set of axis-parallel d-dimensional hyperrectangles, a set of axis-parallel $(d-1)$-dimensional hyperplanes, and a positive integer k, and the task is to select at most k hyperplanes such that every hyperrectangle is intersected (*"stabbed"*) by at least one of them. The problem is well-studied from the approximation point of view, while its parameterized complexity remained unexplored so far. Here, we show, by giving a nontrivial parameterized reduction from the W[1]-complete problem Multicolored Clique, that for $d \geq 2$ the problem is W[1]-hard with respect to the parameter k. For the case $d = 2$ we consider several natural restrictions and show them to be fixed-parameter tractable.

6.1 Introduction and Overview

A geometric covering problem, in the broadest sense, consists of a set of geometric objects and a set of "resources"; the goal is to find a small set of resources that "covers" all the objects. Geometric covering problems arise in many practical applications (for example, train station location, reducing interference in cellular networks) and are subject of intensive research (see [AS98, CV07, GKW08, KMN05, KN96, LM05, MT82, Nus97, Seg99, SW96] and the references given below).

In this chapter, we consider the following problem.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

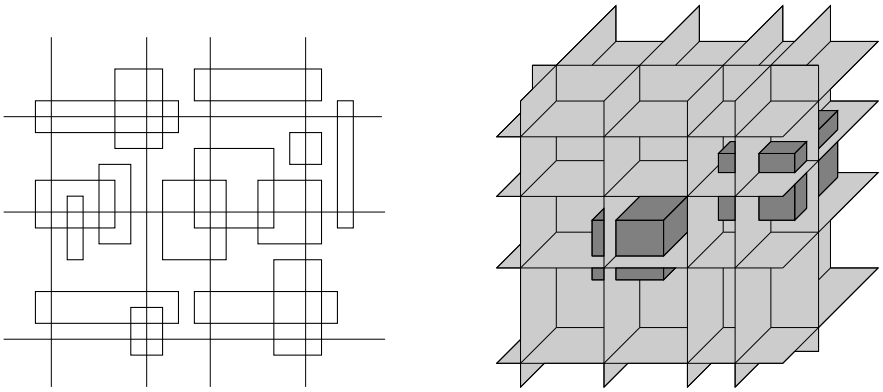


Figure 6.1: Left: An instance of (2-Dimensional) Rectangle Stabbing, consisting of a set of axis-parallel rectangles and a set of axis-parallel lines. Right: An instance of 3-Dimensional Rectangle Stabbing, consisting of a set of axis-parallel boxes and a set of axis-parallel planes.

d-**Dimensional Rectangle Stabbing**

Input: A set R of axis-parallel d-dimensional hyperrectangles, a set L of axis-parallel $(d-1)$-dimensional hyperplanes, and a positive integer k.

Question: Is there a set $L' \subseteq L$ with $|L'| \leq k$ such that every hyperrectangle from R is intersected by at least one hyperplane from L'?

In the case $d = 2$, the set R consists of axis-parallel rectangles in the plane, and L consists of vertical and horizontal lines; we call this variant Rectangle Stabbing for short. Since even the case $d = 2$ is NP-hard (see [GIK02, MSW05]), d-Dimensional Rectangle Stabbing is clearly NP-complete for every constant $d \geq 2$. In the polynomial-time approximation setting, the optimization version of d-Dimensional Rectangle Stabbing is considered, which asks for a *minimum-cardinality* set $L' \subseteq L$ to cover all hyperrectangles from R. See Figure 6.1 for examples.

Applications of d-Dimensional Rectangle Stabbing range from radiotherapy [HM91] to embedded sensor networks, spatial data organization, and statistical analysis [CDKW05, KSPS02] (see also Section 1.1). Moreover, the problem of stabbing arbitrary connected closed shapes (instead of rectangles) in the plane with axis-parallel lines can easily be reduced to (2-Dimensional) Rectangle Stabbing by replacing each shape by its bounding box (that is, by the smallest possible rectangle containing the shape). Similarly, the stabbing problem where only the rectangles in the plane are given and k horizontal and vertical lines shall be inserted that stab all rectangles is equivalent to Rectangle Stabbing: To transform an instance of the former problem into an instance of Rectangle Stabbing, insert a set of $O(|R|)$ lines. For each given rectangle,

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

this set contains four lines: two horizontal lines going through the upper and lower boundary of the rectangle, and two vertical lines going through the left and right boundary of the rectangle. To reduce in the other direction, shrink each rectangle until each of its boundaries coincides with a line, and remove all lines.

Concerning d-Dimensional Rectangle Stabbing and its variants, the literature so far mainly considers polynomial-time approximability. Hassin and Megiddo [HM91] give a factor-$d2^{d-1}$ approximation for stabbing d-dimensional, identical objects (that is, translates of one object) with axis-parallel lines in d-dimensional space. Gaur et al. [GIK02] describe a factor-d approximation for d-Dimensional Rectangle Stabbing; the two-dimensional case Rectangle Stabbing, hence, can be approximated with a factor of two. A similar result was obtained by Mecke et al. [MSW05]; they give a factor-d approximation algorithm for Set Cover, restricted to instances whose matrix representations have at most d blocks of 1s per row. This restricted variant of Set Cover, which we call d-C1P-Set Cover, is a generalization of d-Dimensional Rectangle Stabbing (see Section 6.2). The factor-d approximation also works for the weighted version of d-C1P-Set Cover [MSW05]. Weighted and capacitated versions of d-Dimensional Rectangle Stabbing have been considered by Even et al. [ELR$^+$08] and by Xu and Xu [XX07], also leading to several approximation algorithms. A restricted, but still NP-complete variant of (2-Dimensional) Rectangle Stabbing is called Interval Stabbing; here, every rectangle in the input is intersected by at most one horizontal line, but arbitrarily many vertical lines (that is, every rectangle is just a horizontal interval in the plane). Kovaleva and Spieksma [KS01, KS06] give constant-factor approximation algorithms for several variants of Interval Stabbing. Approximation algorithms for the more general variant of Interval Stabbing where the input contains horizontal *and* vertical intervals have been developed by Hassin and Megiddo [HM91].

Surprisingly, there are no results concerning the fixed-parameter tractability of d-Dimensional Rectangle Stabbing. In this chapter, therefore, we study d-Dimensional Rectangle Stabbing from the viewpoint of parameterized complexity. More specifically, we analyze whether d-Dimensional Rectangle Stabbing is fixed-parameter tractable with respect to the parameter k = "solution size", that is, whether there is an algorithm running in $f(k) \cdot |R \cup L|^{O(1)}$ time. On the one hand, we show in Sections 6.3 and 6.4 that for $d \geq 3$ and $d = 2$, respectively, the problem is W[1]-hard with respect to the parameter k, which presumably means that there is no such algorithm. On the other hand, we consider several natural restrictions of the case $d = 2$ in Section 6.5 and show them to be fixed-parameter tractable with respect to the parameter k.

Note that the known reductions proving the NP-hardness of Rectangle Stabbing and d-Dimensional Rectangle Stabbing [GIK02, MSW05] do not show the W[1]-hardness of these problems.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

6.2 Basics Facts and Definitions

A graph $G = (V, E)$ is called *k-colorable* if there is a function $c : V \to \{1, \ldots, k\}$ satisfying $\forall\{u, v\} \in E : c(u) \neq c(v)$; the function c is then called a *proper vertex k-coloring* for G.

To achieve our hardness results in Sections 6.3 and 6.4, we consider d-Dimensional Rectangle Stabbing as a restriction of the problem Set Cover, which we introduced in Section 2.5:

Set Cover
Input: A binary matrix M and a positive integer k.
Question: Is there a set C' of at most k columns of M such that the submatrix M' of M that is induced by these columns has at least one 1 in every row?

To introduce our restricted versions of Set Cover, we need the following definitions.

Definition 6.1. 1. A binary matrix M has the *d-consecutive-ones property (d-C1P)* if in every row of M there are at most d blocks of 1s.

2. A binary matrix M with columns $c_1, \ldots, c_n$ has the *separated d-consecutive-ones property (d-XC1P)* if M can be split into submatrices $M_1, \ldots, M_d$ such that $M = (M_1 \mid M_2 \mid \ldots \mid M_d)$ and each M_i with $i \in \{1, \ldots, d\}$ has the strong C1P; that is, the columns of M can be partitioned into d sets of consecutive columns $C^1 = \{c_1, \ldots, c_{j_1}\}$, $C^2 = \{c_{j_1+1}, \ldots, c_{j_2}\}$, $\ldots$, $C^d = \{c_{j_{d-1}+1}, \ldots, c_n\}$ such that for every $i \in \{1, \ldots, d\}$ the submatrix of M induced by C^i has at most one block of 1s per row.

See Figure 6.2 for an illustration for the d-C1P and d-XC1P.[1]

If Set Cover is restricted by demanding that the input matrix M must have the d-C1P, then we call the resulting problem d-C1P-Set Cover; if M must have the d-XC1P, then we call the resulting problem d-XC1P-Set Cover.

Observation 6.1. *The problems d-Dimensional Rectangle Stabbing and d-XC1P-Set Cover are equivalent: There are two polynomial-time reductions, one from d-Dimensional Rectangle Stabbing to d-XC1P-Set Cover and one from d-XC1P-Set Cover to d-Dimensional Rectangle Stabbing,*

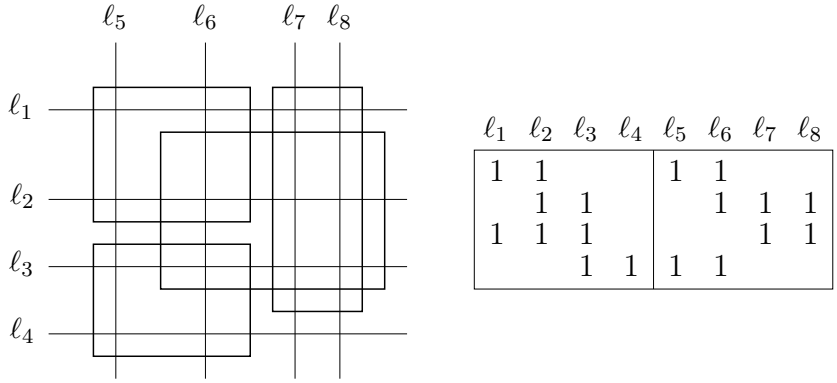
[1] To be consistent with the terms C1P, strong C1P, Circ1P, and strong Circ1P, it would be more appropriate to call the properties introduced in Definition 6.1 *strong* d-C1P and *strong* d-XC1P instead of just d-C1P and d-XC1P, respectively. However, we omit the attribute "strong" here because we assume that in this chapter the columns of every matrix are already ordered in an appropriate way. Note that it is NP-hard to find a permutation of the columns of a binary matrix that minimizes the number of blocks of 1s per row [AM96, FGS96, GGKS95, WLZ07] (see also [BGRS04, WR00]), and it is also NP-hard to find a permutation that minimizes the total number of blocks of 1s [GJ79, Had02] (see also [HL08]).

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

1	1			1	1		
		1	1	1		1	1
		1	1	1	1	1	
1	1					1	1

1	1			1	1		
	1	1	1		1	1	
		1	1	1	1	1	
1	1					1	1

Figure 6.2: Left: A matrix having the 2-C1P, but not the 2-XC1P. Right: A matrix having the 2-XC1P. In all figures of this chapter, only the 1-entries of the matrices are displayed (that is, all 0-entries are omitted).



ℓ_1	ℓ_2	ℓ_3	ℓ_4	ℓ_5	ℓ_6	ℓ_7	ℓ_8
1	1			1	1		
	1	1			1	1	1
1	1	1				1	1
		1	1	1	1		

Figure 6.3: Illustration of the equivalence between (2-Dimensional) Rectangle Stabbing and 2-XC1P-Set Cover.

such that in both mappings hyperrectangles and hyperplanes one-to-one correspond to rows and columns.

This observation is easy to see—the ith dimension in a d-Dimensional Rectangle Stabbing instance can be represented by the column set C^i in a d-XC1P-Set Cover instance and vice versa (see Figure 6.3).

For some of our fixed-parameter algorithms, we make use of the following well-known fact: Given a set of axis-parallel rectangles and a set of vertical (horizontal) lines, the task of finding a minimum-cardinality subset of these vertical (horizontal) lines that intersects all rectangles is polynomial-time solvable:[2] Order the rectangles with respect to their right (bottom) end. Then, repeatedly take the first rectangle r in this order, include the rightmost vertical (bottommost horizontal) line l that intersects r into the solution, and delete all rectangles intersected by l, until all rectangles are deleted. The solution obtained is a minimum-size set of vertical (horizontal) lines that are required to intersect all rectangles. Moreover, all rectangles r that are selected by the algorithm form a "certificate" in the sense that they cannot be intersected by a set of vertical (horizontal) lines

[2]This problem is also known under the name Clique Cover on interval graphs, and it is equivalent to Set Cover on matrices with the C1P (see Section 2.5).

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Input: R: a set of axis-parallel rectangles,
L: a set of lines that are either all vertical or all horizontal,
k: a nonnegative integer.
Output: Either $L' \subseteq L$ or $R^0 \subseteq R$.
If all rectangles from R can be stabbed with a set L' of at most k lines from L, then such a set L' is returned. Otherwise, a set R^0 of $k+1$ rectangles from R is returned that cannot be stabbed with at most k lines from L.

```
1:  function greedy(R, L, k) {
2:      R' := R;   R^0 := ∅;   L' := ∅;
3:      while R' ≠ ∅: {
4:          if L contains only vertical lines: {
5:              r := a rectangle from R' with minimum rx(r);   l := v_rx(r);   }
6:          else {   // L contains only horizontal lines
7:              r := a rectangle from R' with minimum bx(r);   l := h_bx(r);   }
8:          R^0 := R^0 ∪ {r};   L' := L' ∪ {l};
9:          delete all rectangles from R' that are intersected by l;
10:         if |R^0| = k + 1: return R^0;   }
11:     return L';   }
```

Figure 6.4: Greedy algorithm for stabbing a set R of rectangles with at most k lines chosen from a given set L of vertical lines or horizontal lines. If L consists of vertical lines, then we denote the lines in L with $v_1, \ldots, v_n$ ordered from left to right; otherwise we denote the lines in L with $h_1, \ldots, h_m$, ordered from top to bottom. For a rectangle $r \in R$, we denote with $\mathrm{lx}(r), \mathrm{rx}(r), \mathrm{tx}(r), \mathrm{bx}(r)$ the index of the leftmost, rightmost, topmost and bottommost line intersecting r.

that is smaller than the solution found by the algorithm. The pseudocode of this algorithm is displayed in Figure 6.4. The correctness of the algorithm is easy to see: Clearly, if the algorithm outputs a set $L' \subseteq L$, then each rectangle from R is intersected by at least one line from L'. Now assume, for the sake of a contradiction, that after executing line 9 it is possible to stab all rectangles in R^0 with *less* than $|L'| = |R^0|$ lines from L. Then, there must exist a line in L that intersects two rectangles $r_1, r_2 \in R^0$. Without loss of generality, assume that the algorithm considered r_1 before r_2. When r_1 was considered by the algorithm in line 5 (line 7), the line l chosen in line 5 (line 7) also intersected r_2. Hence, r_2 was deleted from R' in line 9. Therefore, r_2 cannot be chosen into R^0 in a later execution of the while loop.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

1	1			1	1				1	1	1	
	1	1	1		1	1		1	1	1		
		1	1	1	1	1					1	1
1	1					1	1		1	1	1	1

Figure 6.5: Example for an instance of 3-XC1P-Set Cover. The columns of the input matrix can be partitioned into three sets of consecutive columns, such that in each of the resulting submatrices there is at most one block of 1s per row.

6.3 W[1]-Hardness of d-Dimensional Rectangle Stabbing with $d \geq 3$

In this section, we prove that d-Dimensional Rectangle Stabbing with parameter k is W[1]-hard for every constant $d \geq 3$. To this end, we exhibit a parameterized reduction from Multicolored Clique, which is defined as follows, to 3-XC1P-Set Cover—the W[1]-hardness of d-Dimensional Rectangle Stabbing, $d \geq 3$, follows from the equivalence between 3-XC1P-Set Cover and 3-Dimensional Rectangle Stabbing.

Multicolored Clique
Input: A k-colorable graph $G = (V, E)$, a positive integer k, and a proper vertex k-coloring $c : V \to \{1, \ldots, k\}$ for G.
Question: Is there a clique with k vertices in G?

Multicolored Clique is W[1]-complete with respect to the parameter k [FHRV09]. Using the "Multicolored Clique reduction technique" designed by Fellows et al. [FHRV09], a parameterized reduction from Multicolored Clique to 3-C1P-Set Cover can be found in a rather straightforward way [Fel07], which proves the W[1]-hardness of the latter problem. However, the W[1]-hardness of 3-XC1P-Set Cover is more difficult to prove because of the more restricted nature of this problem (see Figure 6.5).

The basic scheme of the reduction. The basic scheme of our reduction follows the technique described by Fellows et al. [FHRV09]. The key idea is to use an alternative, equivalent formulation of Multicolored Clique: Given an undirected k-colorable graph $G = (V, E)$, a positive integer k, and a proper vertex k-coloring $c : V \to \{1, \ldots, k\}$ for G, find a set $E' \subseteq E$ with $|E'| = \binom{k}{2}$ and a set $V' \subseteq V$ with $|V'| = k$ that satisfy the following three constraints:

Constraint 1: For every unordered pair $\{a, b\}$ of colors from $\{1, \ldots, k\}$, the edge set E' contains an edge whose endpoints are colored with a and b.

Constraint 2: For every color from $\{1, \ldots, k\}$, the vertex set V' contains a vertex of this color.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Constraint 3: If E' contains an edge $\{u, v\}$, then V' contains the vertices u and v.

Clearly, this formulation of Multicolored Clique is equivalent to the original definition: On the one hand, if there exist sets E' and V' satisfying Constraints 1–3, then V' induces a clique of size k. On the other hand, if a vertex set V' induces a size-k clique and E' is the edge set of this clique, then V' and E' satisfy the three constraints.

Given an instance (G, k, c) of Multicolored Clique, we construct an equivalent instance (M, k') of 3-XC1P-Set Cover (see Figure 6.5) based on this alternative formulation. To this end, define the *color of an edge* $\{u, v\}$, denoted $d(\{u, v\})$, as the set of colors of its endpoints, that is, $d(\{u, v\}) := \{c(u), c(v)\}$. We assume that the edges $E = \{e_1, \ldots, e_{|E|}\}$ and vertices $V = \{v_1, \ldots, v_{|V|}\}$ of G are ordered in such a way that edges and vertices of the same color appear consecutively: For every pair $p_1, p_2 \in \{1, \ldots, |E|\}$ with $p_1 < p_2$ and $d(e_{p_1}) = d(e_{p_2})$ it holds that $\forall p_3 \in \{p_1 + 1, \ldots, p_2 - 1\} : d(e_{p_3}) = d(e_{p_1}) = d(e_{p_2})$, and for every pair $q_1, q_2 \in \{1, \ldots, |V|\}$ with $q_1 < q_2$ and $c(v_{q_1}) = c(v_{q_2})$ it holds that $\forall q_3 \in \{q_1 + 1, \ldots, q_2 - 1\} : c(v_{q_3}) = c(v_{q_1}) = c(v_{q_2})$. We define for every edge color $\{a, b\}$

$$\begin{aligned} E_{\{a,b\}} &:= \{e \in E \mid d(e) = \{a, b\}\}, \\ \text{first}(\{a, b\}) &:= \min\{p \in \{1, \ldots, |E|\} \mid d(e_p) = \{a, b\}\}, \text{ and} \\ \text{last}(\{a, b\}) &:= \max\{p \in \{1, \ldots, |E|\} \mid d(e_p) = \{a, b\}\}. \end{aligned}$$

The idea of the reduction is that every column of M corresponds to an edge or a vertex of the given graph G; the rows of M are constructed in such a way that any solution C' for 3-XC1P-Set Cover on (M, k') corresponds to a solution (E', V') as described above for Multicolored Clique on (G, k, c). That is, the rows of M shall enforce that Constraints 1–3 are satisfied. One approach for such a construction would be to create a matrix M with $|V|+|E|$ columns, one column for each vertex and one column for each edge of G, and to set $k' = k + \binom{k}{2}$. However, we do not know how to encode Constraints 1–3 into the rows of such a matrix without violating the 3-XC1P. Therefore, to obtain a matrix that has the 3-XC1P, we need not only one, but *two* columns in M for every edge e in G. Hence, an instance (G, k, c) of Multicolored Clique is mapped to an instance (M, k'), where $k' = 2 \cdot \binom{k}{2} + k$. The details of the construction of M read as follows.

The columns of M. The matrix M has $2 \cdot |E| + |V|$ columns, partitioned into three sets $C^1 = \{c^1_1, \ldots, c^1_{|E|}\}$, $C^2 = \{c^2_1, \ldots, c^2_{|E|}\}$, and $C^3 = \{c^3_1, \ldots, c^3_{|V|}\}$, ordered as follows: $c^1_1, \ldots, c^1_{|E|}, c^2_1, \ldots, c^2_{|E|}, c^3_1, \ldots, c^3_{|V|}$. Intuitively speaking, for every $p \in \{1, \ldots, |E|\}$, the columns $c^1_p \in C^1$ and $c^2_p \in C^2$ correspond to the edge $e_p \in E$, and for every $q \in \{1, \ldots, |V|\}$, the column $c^3_q \in C^3$ corresponds to the vertex $v_q \in V$.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

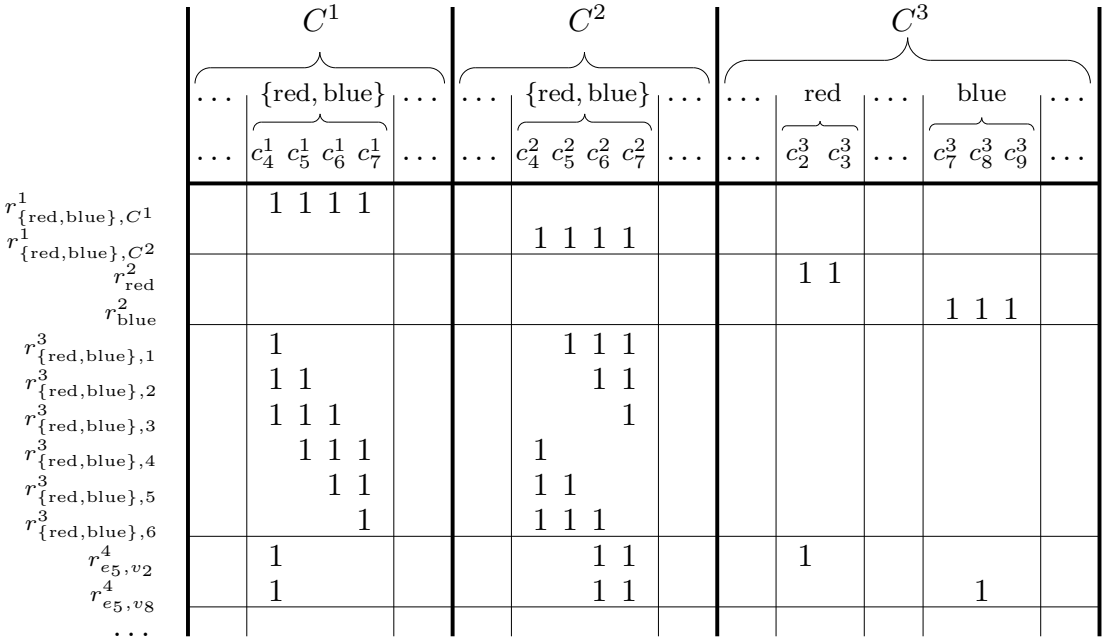


Figure 6.6: Example for the construction of M in the W[1]-hardness proof for 3-XC1P-Set Cover. We assume that in G there are exactly two red vertices v_2, v_3 and exactly three blue vertices v_7, v_8, v_9, among vertices of other colors. Moreover, the only edges between red and blue vertices are e_4, e_5, e_6, e_7 with $e_5 = \{v_2, v_8\}$.

The rows of M. The rows of M have to ensure that every solution C' for 3-XC1P-Set Cover on $(M, k' = 2 \cdot \binom{k}{2} + k)$ corresponds to a subset of edges and vertices of G satisfying Constraints 1–3. Since there are two columns in M for every edge in G, we need *four* types of rows: Rows of Type 1 and 2 ensure that any size-k' solution contains exactly $\binom{k}{2}$ columns from C^1—one of each edge color—, exactly $\binom{k}{2}$ columns from C^2—one of each edge color—, and exactly k columns from C^3—one of each vertex color. Type-3 rows ensure that the columns chosen from C^1 and C^2 are consistent: if a solution contains the column c_j^1, then it must contain c_j^2, and vice versa. Finally, Type-4 rows ensure that if a solution contains the columns c_j^1 and c_j^2 corresponding to an edge $e_j = \{u, v\}$ then it also contains the columns corresponding to the vertices u and v. See Figure 6.6 for an illustration of the following construction details.

Type-1 rows. For every edge color $\{a, b\}$, M contains two rows $r^1_{\{a,b\},C^1}$ and $r^1_{\{a,b\},C^2}$. For $x = 1, 2$, the row $r^1_{\{a,b\},C^x}$ has a 1 in every column $c_j^x \in C^x$ with $d(e_j) = \{a, b\}$, and 0s in all other columns.

Type-2 rows. For every vertex color $a \in \{1, \ldots, k\}$, M contains a row r_a^2 which has a 1 in every column $c_j^3 \in C^3$ with $c(v_j) = a$, and 0s in all other columns.

Observe that the rows of the Types 1 and 2 together with the value of k'

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

force every solution for 3-XC1P-Set Cover on (M, k') to contain *exactly one* column from C^1 for every edge color, *exactly one* column from C^2 for every edge color, and *exactly one* column from C^3 for every vertex color.

Type-3 rows. For every edge color $\{a, b\}$, M contains a set of $2 \cdot (|E_{\{a,b\}}| - 1)$ rows $r^3_{\{a,b\},i}$, where $1 \leq i \leq 2 \cdot (|E_{\{a,b\}}| - 1)$. A row $r^3_{\{a,b\},i}$ with $i \in \{1, \ldots, |E_{\{a,b\}}| - 1\}$ has a 1 in

- every column $c^1_j \in C^1$ with $d(e_j) = \{a, b\}$ and $j < \text{first}(\{a, b\}) + i$ and
- every column $c^2_j \in C^2$ with $d(e_j) = \{a, b\}$ and $j \geq \text{first}(\{a, b\}) + i$,

and 0s in all other columns. A row $r^3_{\{a,b\},i}$ with $i \in \{|E_{\{a,b\}}|, \ldots, 2 \cdot (|E_{\{a,b\}}| - 1)\}$ has a 1 in

- every column $c^1_j \in C^1$ with $d(e_j) = \{a, b\}$ and $j \geq \text{first}(\{a, b\}) + i - (|E_{\{a,b\}}| - 1)$ and
- every column $c^2_j \in C^2$ with $d(e_j) = \{a, b\}$ and $j < \text{first}(\{a, b\}) + i - (|E_{\{a,b\}}| - 1)$,

and 0s in all other columns.

To see that the columns selected from C^1 and C^2 are consistent in every solution for 3-XC1P-Set Cover on (M, k'), observe that, taken a column c^1_j that corresponds to an edge of the color $\{a, b\}$ into the solution, this column does not contain a 1 from the rows $r^3_{\{a,b\},j-\text{first}(\{a,b\})}$ and $r^3_{\{a,b\},j-\text{first}(\{a,b\})+|E_{\{a,b\}}|}$ (provided that these rows exist). Hence, the single column from C^2 that corresponds to an edge of the color $\{a, b\}$ and belongs to the solution must be c^2_j.

Type-4 rows. For every edge $e_p = \{v_{q_1}, v_{q_2}\} \in E$, M contains two rows $r^4_{e_p,v_{q_1}}$ and $r^4_{e_p,v_{q_2}}$. For $i = 1, 2$, the row $r^4_{e_p,v_{q_i}}$ has a 1 in

- every column $c^1_j \in C^1$ with $d(e_j) = d(e_p)$ and $j < p$,
- every column $c^2_j \in C^2$ with $d(e_j) = d(e_p)$ and $j > p$, and
- the column $c^3_{q_i} \in C^3$,

and 0s in all other columns.

Observation 6.2. *In every row of M, there is at most one block of 1s in the submatrix of M induced by the columns of C^1, at most one block of 1s in the submatrix of M induced by the columns of C^2, and at most one block of 1s in the submatrix of M induced by the columns of C^3. Therefore, M has the 3-XC1P.*

Lemma 6.1. *Let $(G = (V, E), k, c)$ be an instance of* Multicolored Clique *and let (M, k') be the instance of* 3-XC1P-Set Cover *obtained by the above construction. Then G contains a clique of size k if and only if there exists a set C' of $k' = 2 \cdot \binom{k}{2} + k$ columns in M that contains at least one 1 in every row.*

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Proof. "⇒": Assume that G contains a clique of size k, and let V' and E' be the k vertices and $\binom{k}{2}$ edges, respectively, of this clique. We claim that the column set

$$C' := \{c_j^1 \mid e_j \in E'\} \cup \{c_j^2 \mid e_j \in E'\} \cup \{c_j^3 \mid v_j \in V'\}$$

has the properties claimed in the lemma. Clearly, C' has cardinality $2 \cdot \binom{k}{2} + k$; it remains to show that C' contains at least one 1 from every row of M.

Since V' and E' form a clique in G, they fulfill Constraints 1 and 2. This, together with the definition of C', directly implies that C' contains a 1 from every row of Types 1 and 2.

To show that C' also contains a 1 from every row of Types 3 and 4, let $\{a, b\}$ be an arbitrary edge color. Observe that, due to Constraint 1, there is an index $j \in \{1, \ldots, |E|\}$ such that $d(e_j) = \{a, b\}$ and $e_j \in E'$. Therefore, we have $c_j^1, c_j^2 \in C'$. We first prove that for any $i \in \{1, \ldots, 2 \cdot (|E_{\{a,b\}}| - 1)\}$ the columns from C' contain a 1 from the row $r := r^3_{\{a,b\},i}$. This suffices to show that C' contains a 1 from every row of Type 3. To see that C' contains a 1 from r, we make a case distinction on the values of i and j. If $i \in \{1, \ldots, |E_{\{a,b\}}| - 1\}$ and $j < \text{first}(\{a, b\}) + i$, or if $i \in \{|E_{\{a,b\}}|, \ldots, 2 \cdot (|E_{\{a,b\}}| - 1)\}$ and $j \geq \text{first}(\{a, b\}) + i - (|E_{\{a,b\}}| - 1)$, then c_j^1 contains a 1 in row r. If $i \in \{1, \ldots, |E_{\{a,b\}}| - 1\}$ and $j \geq \text{first}(\{a, b\}) + i$, or if $i \in \{|E_{\{a,b\}}|, \ldots, 2 \cdot (|E_{\{a,b\}}| - 1)\}$ and $j < \text{first}(\{a, b\}) + i - (|E_{\{a,b\}}| - 1)$, then c_j^2 contains a 1 in row r. In both cases, C' contains a 1 from r. Next, we prove that C' contains a 1 from every Type-4 row $r := r^4_{e_p, v_q}$ with $d(e_p) = \{a, b\}$. If $j < p$, then c_j^1 contains a 1 in row r. If $j > p$, then c_j^2 contains a 1 in row r. If, however, $j = p$, then observe that, due to Constraint 3 and the definition of C', we have $c_q^3 \in C'$. Since c_q^3 has a 1 in row r, C' contains a 1 from r in all three cases.

"⇐": Assume that there exists a set C' of columns as described in the lemma. To show that G contains a clique with k vertices, it suffices to show that there is a vertex set $V' \subseteq V$ and an edge set $E' \subseteq E$ fulfilling Constraints 1–3. We construct E' and V' as follows.

$$E' := \{e_j \in E \mid c_j^1 \in C' \ \vee \ c_j^2 \in C'\}, \text{ and}$$
$$V' := \{v_j \in V \mid c_j^3 \in C'\}.$$

First observe that, due to the construction of the Type-1 rows of M, for every edge color $\{a, b\}$, the set C' must contain at least one column $c_{j_1}^1$ from C^1 and at least one column $c_{j_2}^2$ from C^2 such that e_{j_1} and e_{j_2} are edges in G with $d(e_{j_1}) = \{a, b\}$ and $d(e_{j_2}) = \{a, b\}$. Similarly, due to the construction of the Type-2 rows of M, for every vertex color a, the set C' must contain at least one column $c_{j_3}^3$ such that v_{j_3} is a vertex in G whose color is a. Together with the definition of E' and V', these observations imply that E' and V' fulfill Constraints 1 and 2. Now, since C' contains $2 \cdot \binom{k}{2} + k$ columns, it follows that C' cannot contain more than *exactly one* column from C^1 for every edge color $\{a, b\}$, *exactly one* column from C^2 for every edge color $\{a, b\}$, and *exactly one* column from C^3 for every vertex color a—otherwise, C' would consist of more than $2 \cdot \binom{k}{2} + k$ columns.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

We are now ready to prove that the columns from C^1 and C^2 are chosen "consistently", that is, for any $j \in \{1, \ldots, |E|\}$, we have $c^1_j \in C'$ iff $c^2_j \in C'$. To see this, let $\{a, b\}$ be an arbitrary edge color. We know that C' contains exactly one column $c^1_{j_1}$ from C^1 that corresponds to an edge of the color $\{a, b\}$ and exactly one column $c^2_{j_2}$ from C^2 that corresponds to an edge of the color $\{a, b\}$; in other words, $j_1, j_2 \in \{\text{first}(\{a, b\}), \ldots, \text{last}(\{a, b\})\}$. We will now prove that $j_2 = j_1$: we first show that $j_2 \geq j_1$, and then we show that $j_2 \leq j_1$.

Clearly, if $j_1 = \text{first}(\{a, b\})$, then $j_2 \geq j_1$. To see that $j_2 \geq j_1$ also holds in the case when $j_1 > \text{first}(\{a, b\})$, we make the following argument based on the construction of the Type-3 rows. If $j_1 > \text{first}(\{a, b\})$, then the column $c^1_{j_1}$ does not contain a 1 in the row $r := r^3_{\{a,b\}, j_1 - \text{first}(\{a,b\})}$. Note that all 1s of the row r lie in columns from C^1 and from C^2 that correspond to edges of the color $\{a, b\}$. Note also that $c^1_{j_1}$ is the only column from C^1 in C' that corresponds to an edge with color $\{a, b\}$. Therefore, the column $c^2_{j_2}$ (which is the only column from C^2 in C' corresponding to an edge with color $\{a, b\}$) must contain a 1 from r, which implies, due to the construction of r, that $j_2 \geq j_1$. We can similarly argue that $j_2 \leq j_1$: Clearly, if $j_1 = \text{last}(\{a, b\})$, then $j_2 \leq j_1$. If, however, $j_1 < \text{last}(\{a, b\})$, then the column $c^1_{j_1}$ does not contain a 1 in the row $r := r^3_{\{a,b\}, j_1 - \text{first}(\{a,b\}) + |E_{\{a,b\}}|}$, which again implies that we have to select the column $c^2_{j_2}$ in such a way that it contains a 1 from r. Therefore, we have $j_2 \leq j_1$ also in this case. It follows that $j_2 = j_1$, which proves the claimed consistency.

Finally, we can prove that the edges and vertices in E' and V' fulfill Constraint 3. To this end, let e_j be an edge in E', and let $d(e_j) = \{a, b\}$. Due to the definition of E' and the consistency between the columns selected from C^1 into C' and the columns selected from C^2 into C', we know that c^1_j and c^2_j belong to C'. We have also seen that, apart from these two columns, the set C' contains no other column corresponding to an edge of the color $\{a, b\}$. Now, let the vertices v_{q_1} and v_{q_2} be the endpoints of e_j. Then M contains the two rows $r^4_{e_j, v_{q_1}}$ and $r^4_{e_j, v_{q_2}}$, which both do not have a 1 in any of the columns c^1_j and c^2_j. Therefore, the set C' must contain the column $c^3_{q_1}$ because of the row $r^4_{e_j, v_{q_1}}$ and the column $c^3_{q_2}$ because of the row $r^4_{e_j, v_{q_2}}$. Due to the definition of V', the vertices v_{q_1} and v_{q_2} then belong to V', and, therefore, Constraint 3 is fulfilled. □

Observation 6.2 and Lemma 6.1 imply the following result.

Lemma 6.2. 3-XC1P-Set Cover *and* 3-Dimensional Rectangle Stabbing *are* W[1]*-hard with respect to the parameter* k.

For any $d > 3$, by adding additional "dummy" columns containing only 0s to the above construction, we can always obtain a matrix M that has at least d columns. Obviously, this matrix can then be partitioned into d sets of consecutive columns in such a way that in each of the resulting submatrices there is at most

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

one block of 1s per row; hence, the matrix has the d-XC1P. This implies the following.

Theorem 6.1. *For every* $d \geq 3$, d-XC1P-Set Cover *and* d-Dimensional Rectangle Stabbing *are* W[1]*-hard with respect to the parameter* k.

6.4 W[1]-Hardness of 2-Dimensional Rectangle Stabbing

In the previous section, we have shown that d-Dimensional Rectangle Stabbing with parameter k is W[1]-hard for $d \geq 3$. Now we describe a more complicated reduction from Multicolored Clique showing that d-Dimensional Rectangle Stabbing is W[1]-hard even in the case $d = 2$.

To reduce Multicolored Clique to (2-Dimensional) Rectangle Stabbing, we construct, given an instance of Multicolored Clique, a matrix M whose columns can be partitioned into *two* sets C^1 and C^2 of consecutive columns such that for both $i = 1$ and $i = 2$ the submatrix of M induced by C^i has at most one block of 1s per row. To this end, we use a very similar approach as in Section 6.3—the matrix M constructed to show the hardness of (2-Dimensional) Rectangle Stabbing is just a modification of the matrix M described in Section 6.3: The column set C^1 of the "new" matrix M contains the columns $c^1_1, \ldots, c^1_{|E|}$ and $c^2_1, \ldots, c^2_{|E|}$, that is, the "new" column set C^1 consists of the "old" column sets C^1 and C^2 from Section 6.3. The "new" column set C^2 contains the columns $c^3_1, \ldots, c^3_{|V|}$, that is, the columns from the "old" column set C^3 from Section 6.3. As we will see, in addition to these columns we have to add some more columns to C^2.

The first reason why the matrix M in Section 6.3 does not have the 2-XC1P is that Type-4 rows typically contain three blocks of 1s. Therefore, we have to arrange the columns in the set C^1 of our new construction in a different way. The second reason is that Type-3 rows typically contain one block of 1s in the columns $c^1_1, \ldots, c^1_{|E|}$ and one block of 1s in the columns $c^2_1, \ldots, c^2_{|E|}$. Therefore, we replace these rows, which are needed to enforce that the columns selected from $c^1_1, \ldots, c^1_{|E|}$ are consistent with the columns selected from $c^2_1, \ldots, c^2_{|E|}$, by a new construction serving the same purpose: This construction consists of a set of columns $c^4_1, \ldots, c^4_{|E|}$ added to C^2, and a set of new Type-3 rows replacing the Type-3 rows described in Section 6.3. An instance (G, k, c) of Multicolored Clique is, thus, mapped to an instance (M, k'), where $k' = 3 \cdot \binom{k}{2} + k$ and M is constructed as follows.

The columns of M**.** The matrix M has $3 \cdot |E| + |V|$ columns, partitioned into two sets C^1 and C^2. The column set C^1 consists of two subsets of columns: a subset D^1 consisting of the columns $c^1_1, \ldots, c^1_{|E|}$, and a subset D^2 consisting of the

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

columns $c^2_1, \ldots, c^2_{|E|}$. The column set C^2 also consists of two subsets of columns: the subset D^3 consisting of the columns $c^3_1, \ldots, c^3_{|V|}$, and the subset D^4 consisting of the columns $c^4_1, \ldots, c^4_{|E|}$.

These columns are ordered as follows in M. The leftmost $2 \cdot |E|$ columns of M are those from C^1, the remaining $|V| + |E|$ columns are those from C^2. The columns from C^1 are ordered in such a way that columns corresponding to edges of the same color appear consecutively. More precisely, for every edge color $\{a, b\}$, there are $2 \cdot |E_{\{a,b\}}|$ consecutive columns

$$c^2_{\text{first}(\{a,b\})}, \ldots, c^2_{\text{last}(\{a,b\})}, c^1_{\text{first}(\{a,b\})}, \ldots, c^1_{\text{last}(\{a,b\})}.$$

The columns from C^2 are ordered as follows: To the right of the columns from C^1, there are the $|V|$ columns $c^3_1, \ldots, c^3_{|V|}$. The rightmost $|E|$ columns of M, finally, are the columns $c^4_1, \ldots, c^4_{|E|}$ (which implies that columns corresponding to edges of the same color appear consecutively and that, for every edge color $\{a, b\}$, there are $|E_{\{a,b\}}|$ consecutive columns $c^4_{\text{first}(\{a,b\})}, \ldots, c^4_{\text{last}(\{a,b\})}$). Intuitively speaking, for every $p \in \{1, \ldots, |E|\}$, the columns $c^1_p \in C^1$, $c^2_p \in C^1$, and $c^4_p \in C^2$ correspond to the edge $e_p \in E$, and for every $q \in \{1, \ldots, |V|\}$, the column $c^3_q \in C^2$ corresponds to the vertex $v_q \in V$.

The rows of M. As in Section 6.3, there are four types of rows: Rows of Type 1 and 2 ensure that any solution C' for 2-XC1P-Set Cover on (M, k') contains exactly $\binom{k}{2}$ columns from D^1—one of each edge color—, $\binom{k}{2}$ columns from D^2—one of each edge color—, $\binom{k}{2}$ columns from D^4—one of each edge color—, and k columns from D^3—one of each vertex color. Type-3 rows ensure that the columns chosen from D^1, D^2, and D^4 are consistent: if a solution contains the column c^1_j, then it must contain c^4_j, and vice versa; analogously, if a solution contains the column c^2_j, then it must contain c^4_j, and vice versa. Finally, Type-4 rows ensure that if a solution contains the columns c^1_j, c^2_j, and c^4_j corresponding to an edge $e_j = \{u, v\}$ then it also contains the columns corresponding to the vertices u and v. See Figure 6.7 for an illustration of the following construction details.

Type-1 rows. For every edge color $\{a, b\}$, M contains three rows $r^1_{\{a,b\},D^1}$, $r^1_{\{a,b\},D^2}$, and $r^1_{\{a,b\},D^4}$. For $x = 1, 2, 4$, the row $r^1_{\{a,b\},D^x}$ has a 1 in every column $c^x_j \in D^x$ with $d(e_j) = \{a, b\}$, and 0s in all other columns.

Type-2 rows. These rows are identical to the Type-2 rows in Section 6.3: For every vertex color $a \in \{1, \ldots, k\}$, M contains a row r^2_a which has a 1 in every column $c^3_j \in D^3$ with $c(v_j) = a$, and 0s in all other columns.

Type-3 rows. For every edge color $\{a, b\}$, M contains a set of $2 \cdot (|E_{\{a,b\}}| - 1)$ rows $r^3_{\{a,b\},D^1,i}$ and a set of $2 \cdot (|E_{\{a,b\}}| - 1)$ rows $r^3_{\{a,b\},D^2,i}$, where in both cases $1 \leq i \leq 2 \cdot (|E_{\{a,b\}}| - 1)$. A row $r^3_{\{a,b\},D^x,i}$ with $x \in \{1, 2\}$ and $i \in \{1, \ldots, |E_{\{a,b\}}| - 1\}$ has a 1 in

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

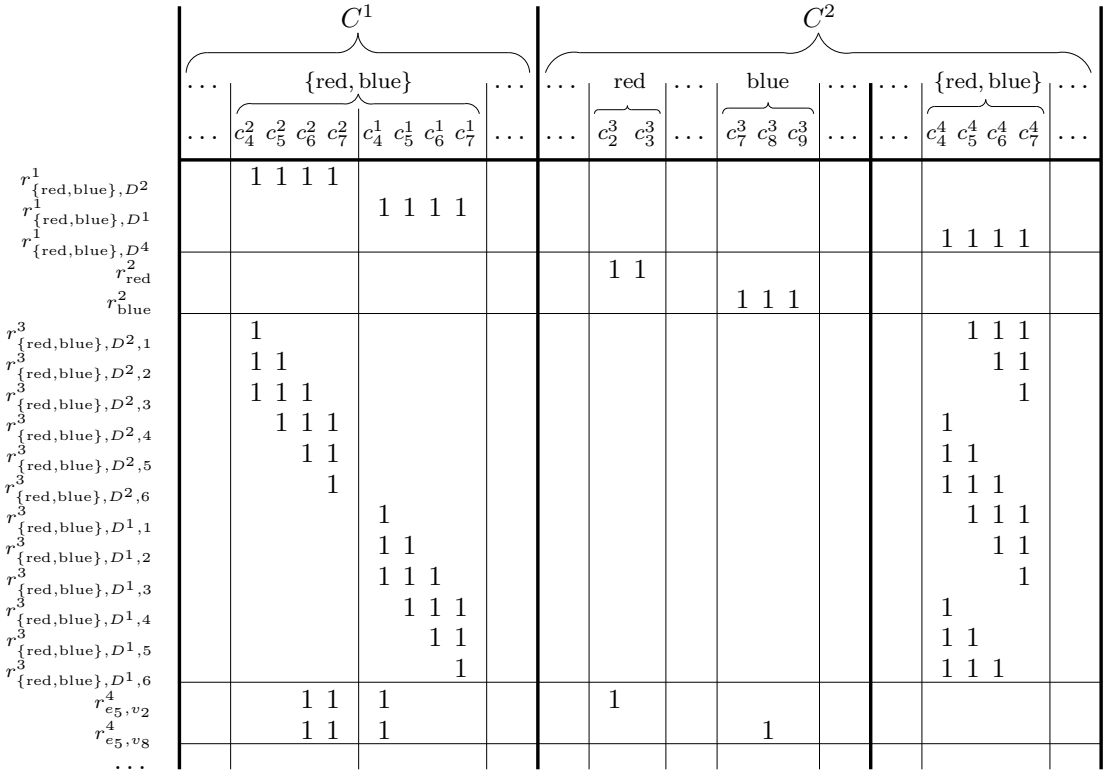


Figure 6.7: Example for the construction of M in the W[1]-hardness proof for 2-XC1P-Set Cover. We assume that in G there are exactly two red vertices v_2, v_3 and exactly three blue vertices v_7, v_8, v_9, among vertices of other colors. Moreover, the only edges between red and blue vertices are e_4, e_5, e_6, e_7 with $e_5 = \{v_2, v_8\}$.

- every column $c_j^x \in D^x$ with $d(e_j) = \{a, b\}$ and $j < \text{first}(\{a, b\}) + i$ and
- every column $c_j^4 \in D^4$ with $d(e_j) = \{a, b\}$ and $j \geq \text{first}(\{a, b\}) + i$,

and 0s in all other columns. A row $r^3_{\{a,b\},D^x,i}$ with $x \in \{1, 2\}$ and $i \in \{|E_{\{a,b\}}|, \ldots, 2 \cdot (|E_{\{a,b\}}| - 1)\}$ has a 1 in

- every column $c_j^x \in D^x$ with $d(e_j) = \{a, b\}$ and $j \geq \text{first}(\{a, b\}) + i - (|E_{\{a,b\}}| - 1)$ and
- every column $c_j^4 \in D^4$ with $d(e_j) = \{a, b\}$ and $j < \text{first}(\{a, b\}) + i - (|E_{\{a,b\}}| - 1)$,

and 0s in all other columns.

Type-4 rows. Again, these rows are identical to the Type-4 rows in Section 6.3: For every edge $e_p = \{v_{q_1}, v_{q_2}\} \in E$, M contains two rows $r^4_{e_p,v_{q_1}}$ and $r^4_{e_p,v_{q_2}}$. For $i = 1, 2$, the row $r^4_{e_p,v_{q_i}}$ has a 1 in

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

- every column $c_j^1 \in D^1$ with $d(e_j) = d(e_p)$ and $j < p$,
- every column $c_j^2 \in D^2$ with $d(e_j) = d(e_p)$ and $j > p$, and
- the column $c_{q_i}^3 \in D^3$,

and 0s in all other columns.

The description of the construction implies that M has the 2-XC1P. Moreover, the graph G in the given Multicolored Clique instance contains a clique of size k iff there exists a set of $k' = 3 \cdot \binom{k}{2} + k$ columns in M that contains a 1 in every row—this can be seen in analogy to the proof of Lemma 6.1. Therefore, we have the following result.

Theorem 6.2. 2-XC1P-Set Cover, 2-C1P-Set Cover, *and* (2-Dimensional) Rectangle Stabbing *are* W[1]*-hard with respect to the parameter* k.

By adding additional "dummy" columns to the above construction, we can obtain a matrix M where in every row the number of 1-entries in the columns of C^1 equals the number of 1-entries in the columns of C^2. Therefore, we get the W[1]-hardness of the following restricted variant of (2-Dimensional) Rectangle Stabbing (a rectangle r is called a *square* if the number of horizontal lines intersecting it is equal to the number of vertical lines intersecting it).

Theorem 6.3. *The restricted variant of* Rectangle Stabbing *where all rectangles in* R *are squares is* W[1]*-hard with respect to the parameter* k.

6.5 Fixed-Parameter Algorithms for Restricted Variants of 2-Dimensional Rectangle Stabbing

In Sections 6.3 and 6.4, we have seen that d-Dimensional Rectangle Stabbing with parameter k is W[1]-hard for each $d \geq 2$. In this section, we consider some natural restrictions of (2-Dimensional) Rectangle Stabbing and show them to be fixed-parameter tractable.

As described in Section 6.1, Rectangle Stabbing asks to stab a set R of axis-parallel rectangles with at most k lines chosen from a given set L of vertical and horizontal lines. For an instance (R, L, k) of Rectangle Stabbing, let $L = V \cup H$, where $V = \{v_1, \ldots, v_n\}$ are the vertical lines, ordered from left to right, and $H = \{h_1, \ldots, h_m\}$ are the horizontal lines, ordered from top to bottom. For a rectangle $r \in R$, let $\mathrm{lx}(r), \mathrm{rx}(r), \mathrm{tx}(r), \mathrm{bx}(r)$ be the index of the leftmost, rightmost, topmost and bottommost line intersecting r. Define the width $\mathrm{wh}(r) := \mathrm{rx}(r) - \mathrm{lx}(r) + 1$ and the height $\mathrm{ht}(r) := \mathrm{bx}(r) - \mathrm{tx}(r) + 1$ as the number of vertical and horizontal lines, respectively, intersecting r.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

We start with some well-known data reduction rules for Rectangle Stabbing, whose correctness is obvious—the rules are derived from the data reduction rules for Set Cover described in Section 2.5.

Rule 1: If there are two rectangles $r_1, r_2 \in R$ such that every line in L that intersects r_1 also intersects r_2, then delete r_2.

Rule 2: If there are two lines $l_1, l_2 \in L$ such that every rectangle in R that is intersected by l_2 is also intersected by l_1, then delete l_2.

Rule 3: If there is a line $l \in L$ that intersects no rectangle from R, then delete l from L.

Rule 4: If there is a rectangle that is intersected by exactly one line $l \in L$, then delete all rectangles that are intersected by l, delete l, and decrease k by one.

Rule 5: If $k \geq 0$ and R contains no rectangle, then the instance is a *yes*-instance.

Rule 6: If there is a rectangle in R that is intersected by no line from L, or if $k < 0$, or if $k = 0$ and R contains at least one rectangle, then the instance is a *no*-instance.

The following observation is an immediate consequence of Rule 2.

Observation 6.3. *In a reduced problem instance, for every vertical line $v_j \in V$, there exist rectangles $r, r' \in R$ with $\mathrm{lx}(r) = j$ and $\mathrm{rx}(r') = j$. For every horizontal line $h_i \in H$ there exist rectangles $r, r' \in R$ with $\mathrm{tx}(r) = i$ and $\mathrm{bx}(r') = i$.*

In particular, Observation 6.3 implies that in a reduced problem instance there exist rectangles $r_1, r_2 \in R$ such that $\mathrm{wh}(r_1) = 1$ and $\mathrm{ht}(r_2) = 1$: just let r_1 be a rectangle with $\mathrm{rx}(r_1) = 1$ and r_2 a rectangle with $\mathrm{bx}(r_2) = 1$.

6.5.1 Rectangles with Bounded Height

We first consider the restriction where the height of every rectangle in R is bounded by a number b. Even the case $b = 1$ where every rectangle is just a horizontal segment is NP-complete; Hassin and Megiddo [HM91] and Kovaleva and Spieksma [KS01, KS06] gave approximation algorithms for this case and some of its variants.

For our FPT considerations, we use a simple search-tree algorithm using Observation 6.3. At every step, apply the data reduction rules until the current instance is reduced, search for a rectangle r with $\mathrm{rx}(r) = 1$, and branch as follows: either select the single vertical line that intersects r or select one of the at most b horizontal lines that intersect r.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Theorem 6.4. *The restricted variant of* Rectangle Stabbing *where the height* $\mathrm{ht}(r)$ *of every rectangle* $r \in R$ *is bounded from above by a number* b *can be solved in* $(b+1)^k \cdot n^{O(1)}$ *time.*

The algorithm described above can be modified to solve also the weighted version of the problem, where every line l has a weight $w(l) \geq 1$ that is bounded from above by a number b', and a minimum-*weight* subset of L shall be found. To this end, we modify the data reduction rules: Instead of Rule 2, we have to use the following reduction rule.

Rule 2': If there are two lines $l_1, l_2 \in L$ such that $w(l_2) \geq w(l_1)$ and every rectangle in R that is intersected by l_2 is also intersected by l_1, then delete l_2.

Observation 6.4. *Let* $v_1, \ldots, v_{b'}$ *be the leftmost* b' *vertical lines in a reduced problem instance. If* L *contains more than* b' *vertical lines, then there is a vertical line* $v_j \in \{v_1, \ldots, v_{b'}\}$ *such that there exists a rectangle* $r \in R$ *with* $\mathrm{rx}(r) = j$.

Proof. Assume, for the sake of a contradiction, that there exists no line v_j as claimed. Then, all rectangles that are intersected by one of the lines $v_1, \ldots, v_{b'}$ are also intersected by $v_{b'+1}$. Since at least one of the b' lines $v_1, \ldots, v_{b'}$ must have the same weight as $v_{b'+1}$, Rule 2' would have deleted this line. □

Observation 6.4 says that in a reduced problem instance there exists always a rectangle $r \in R$ such that $\mathrm{wh}(r) \leq b'$: just let r be a rectangle with $\mathrm{rx}(r) \in \{1, \ldots, b'\}$.

Now a search tree algorithm can be used to solve the problem. As in the unweighted case, apply the data reduction rules in every node of the search tree until the current instance is reduced. Then search for a rectangle r with $\mathrm{rx}(r) = b'$ and branch as follows: either select one of the at most b' vertical lines that intersect r or select one of the at most b horizontal lines that intersect r. The running time of the algorithm is $(b+b')^k \cdot n^{O(1)}$.

6.5.2 Rectangles with Bounded Width or Height

Here, we consider a generalization of the previous restriction: Now, for every rectangle r in R the width $\mathrm{wh}(r)$ *or* the height $\mathrm{ht}(r)$ is bounded from above by a number b. Clearly, even the case $b = 1$, where every rectangle is either a horizontal or a vertical segment, is NP-complete; this case was already considered by Hassin and Megiddo [HM91] from the approximation point of view.

The approach outlined in Section 6.5.1 does not work anymore since in a reduced instance the height of every rectangle r with $\mathrm{rx}(r) = 1$ may be unbounded. However, there is again a search-tree algorithm. Let $R_h \subseteq R$ be the set of rectangles with bounded height and let $R_v \subseteq R$ be the set of rectangles with bounded width. Now, we try all possible ways to write k as a sum $k_h + k_v$, where k_h and k_v denote the number of horizontal and vertical lines, respectively, allowed

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

to be chosen into the solution. For every splitting of k into k_h and k_v, we run a branching algorithm, which performs in every step the following actions.

First, compute the minimum number of vertical lines required to intersect the rectangles in R_h. This is polynomial-time doable, and the simple greedy algorithm in Figure 6.4 obtains such a set of vertical lines.

If R_h cannot be stabbed with a set of at most k_v vertical lines, then the algorithm in Figure 6.4 outputs a set $R_h^0 \subseteq R_h$ of size k_v+1 such that the optimum number of vertical lines needed to intersect all rectangles in R_h^0 is exactly $k_v + 1$. Any solution for Rectangle Stabbing on (R, L, k) consisting of at most k_v vertical and at most k_h horizontal lines must intersect at least one rectangle in R_h^0 by a horizontal line. Hence, branch on the $(k_v + 1) \cdot b$ possibilities to do so.

If, however, all rectangles in R_h can be intersected with k_v vertical lines, we use the greedy algorithm to check whether the rectangles in R_v can be intersected with k_h horizontal lines. If not, we branch on $(k_h + 1) \cdot b$ possibilities in analogy to the branching for R_h^0 described above; otherwise, we return the union of the solutions returned by the two calls to the greedy algorithm. Figure 6.8 shows a pseudocode for this algorithm. The branching number is at most bk, which leads to the following theorem.

Theorem 6.5. *The restricted variant of* Rectangle Stabbing *where the width or the height of every rectangle in R is bounded from above by a number b can be solved in $(bk)^k \cdot n^{O(1)}$ time.*

6.5.3 Bounded Intersection

In this subsection, we consider the restriction of Rectangle Stabbing where every horizontal line in L intersects at most b rectangles from R; this restriction was already considered by Kovaleva and Spieksma [KS01, KS06] from the approximation point of view. For $b = 1$, this problem is clearly polynomial-time solvable since the horizontal lines can just be ignored. For $b = 2$, the problem is NP-complete because we can give a reduction from the NP-complete [GJ79] problem Vertex Cover (see Section 1.3 for the problem definition): Given a graph $G = (V, E)$ with $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_m\}$, first transform it into a graph $G' = (V', E')$ by replacing every edge $e = \{v_i, v_j\}$ of G by a path v_j–x_e–y_e–v_j. Then G' has a vertex cover of size $|E| + k$ iff G has a vertex cover of size k. To see this, note that there is always a minimum-size vertex cover for G' that contains exactly one of the two vertices x_e, y_e for every edge e. Now the Vertex Cover instance $(G', |E|+k)$ can be transformed into an instance of 2-XC1P-Set Cover as follows. Let $r = |E'| = 3|E|$ and $s = |V'| = |V| + 2|E|$, and let M be an $r \times s$ matrix whose columns represent the vertices and whose rows represent the edges of G'. The columns of M are ordered as follows:

$$v_1, v_2, \ldots, v_n, x_{e_1}, y_{e_1}, \ldots, x_{e_m}, y_{e_m}.$$

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Input: R_h: a set of axis-parallel rectangles with bounded height,
R_v: a set of axis-parallel rectangles with bounded width,
H, V: a set of horizontal lines and a set of vertical lines,
k_h, k_v: nonnegative integers.

Output: A subset of $H \cup V$ containing at most k_h lines from H and at most k_v lines from V that stabs all rectangles from $R_h \cup R_v$, or null, if no such subset exists.

1: **function** stab$(R_h, R_v, H, V, k_h, k_v)$ {
2: **if** greedy(R_h, V, k_v) returns a set $R_h^0 \subseteq R_h$ of rectangles: {
3: **if** $k_h = 0$: **return** *null*;
4: **for each** rectangle $r \in R_h^0$: **for each** line $h \in H$ that intersects r: {
5: $R'_h := R_h \setminus R_h(h)$; $R'_v := R_v \setminus R_v(h)$; $H' := H \setminus \{h\}$;
6: $A :=$ stab$(R'_h, R'_v, H', V, k_h - 1, k_v)$;
7: **if** $A \neq$ *null*: **return** $A \cup \{h\}$; }
8: **return** *null*; }
9: **if** greedy(R_v, H, k_h) returns a set $R_v^0 \subseteq R_v$ of rectangles: {
10: **if** $k_v = 0$: **return** *null*;
11: **for each** rectangle $r \in R_v^0$: **for each** line $v \in V$ that intersects r: {
12: $R'_h := R_h \setminus R_h(v)$; $R'_v := R_v \setminus R_v(v)$; $V' := V \setminus \{v\}$;
13: $A :=$ stab$(R'_h, R'_v, H, V', k_h, k_v - 1)$;
14: **if** $A \neq$ null: **return** $A \cup \{v\}$; }
15: **return** *null*; }
16: **return** the union $V' \cup H'$ of the solutions returned by the two calls (lines 2 and 9) of greedy(); }

Figure 6.8: Branching algorithm for stabbing a set $R_v \cup R_h$ of rectangles with at most k_v lines chosen from a given set V of vertical lines and at most k_h lines chosen from a given set H of horizontal lines. For a line l, we denote with $R_h(l)$ and $R_v(l)$ the set of all rectangles in R_h and R_v, respectively, that are intersected by l.

An entry in a row i and a column j of M is 1 iff the edge e_i is incident to the vertex corresponding to column j. Clearly M has the 2-XC1P with $C^1 = \{v_1, v_2, \ldots, v_n\}$ and $C^2 = \{x_{e_1}, y_{e_1}, \ldots, x_{e_m}, y_{e_m}\}$. Moreover, there are exactly two 1s in every column $x_{e_1}, y_{e_1}, \ldots, x_{e_m}, y_{e_m}$. Therefore, the matrix M can be transformed into an equivalent instance of the restricted variant of Rectangle Stabbing. Obviously, G' has a vertex cover of size $|E|+k$ iff M has a solution of size $|E|+k$. We just mention in passing that this reduction also proves Corollary 1 of [MSW05] in an easier way.

For the case $b = 2$, there is a simple $k^k \cdot n^{O(1)}$-time branching algorithm: In every node of the search tree, try to stab all rectangles while only using vertical lines—to this end, use the greedy algorithm of Figure 6.4. If the greedy algorithm

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

does not return a solution but a set of $k + 1$ rectangles that cannot be stabbed with k vertical lines, then one has to select a horizontal line that intersects two of these rectangles. We can assume that the instance is reduced with respect to the data reduction rules described above, and, therefore, there can be at most k such lines. Branch on which of these lines should be taken into the solution. However, we do not know whether this algorithm can be generalized for the case $b \geq 3$. Nevertheless, we show that the restriction of Rectangle Stabbing is fixed-parameter tractable with respect to the combined parameters k and b by developing a problem kernel.

First, in addition to the previously mentioned data reduction rules, we use the following data reduction rule:

Rule 7: If there are $bk + 2$ rectangles $r_1, \ldots, r_{bk+2} \in R$ such that for each $i \in \{1, \ldots, bk + 1\}$ it holds that every vertical line that intersects r_i also intersects r_{bk+2}, then delete r_{bk+2}.

The correctness of this data reduction rule follows from the fact that k horizontal lines cannot intersect all rectangles $r_1, \ldots, r_{bk+1}$. Hence, if the instance with r_{bk+2} deleted is a *yes*-instance, every solution must contain a vertical line stabbing some of the rectangles $r_1, \ldots, r_{bk+1}$, and this line also stabs r_{bk+2} in the original instance, which, therefore, is also a *yes*-instance.

The following two observations are immediate consequences of Rule 7.

Observation 6.5. *For every rectangle r in a reduced instance there are at most bk rectangles $r' \neq r$ with $\mathrm{lx}(r') \geq \mathrm{lx}(r)$ and $\mathrm{rx}(r') \leq \mathrm{rx}(r)$.*

Observation 6.6. *In a reduced instance, for every $j \in \{1, \ldots, n\}$ there are at most $bk + 1$ rectangles r with $\mathrm{lx}(r) = j$.*

The observations made so far lead to the following correlation between the position of a rectangle and its maximum possible width:

Lemma 6.3. *For every rectangle $r \in R$ in a reduced instance, $\mathrm{rx}(r) \leq (bk + 1) \cdot \mathrm{lx}(r)$.*

Proof. By induction on $\mathrm{lx}(r)$. For all rectangles r with $\mathrm{lx}(r) = 1$, the lemma is true: Assume, for the sake of a contradiction, that there is a rectangle r with $\mathrm{lx}(r) = 1$ and $\mathrm{rx}(r) > (bk + 1)$. Due to Observation 6.3, there must be a rectangle r' with $\mathrm{rx}(r') = p$ for every $p \in \{\mathrm{lx}(r), \ldots, \mathrm{rx}(r) - 1\}$. This means that there are at least $bk + 1$ rectangles $r' \neq r$ with $\mathrm{lx}(r') \geq \mathrm{lx}(r)$ and $\mathrm{rx}(r') \leq \mathrm{rx}(r)$, contradicting Observation 6.5.

Now let j be an integer greater than 1 and assume the lemma to be true for all rectangles r with $1 \leq \mathrm{lx}(r) \leq j$. Let r be a rectangle with $\mathrm{lx}(r) = j + 1$, and assume, for the sake of a contradiction, that $\mathrm{rx}(r) > (bk + 1) \cdot \mathrm{lx}(r)$. Observation 6.3 implies that there must be a rectangle r' with $\mathrm{rx}(r') = p$ for every $p \in \{\mathrm{lx}(r), \ldots, \mathrm{rx}(r) - 1\}$. Due to Observation 6.5, at most bk of these

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

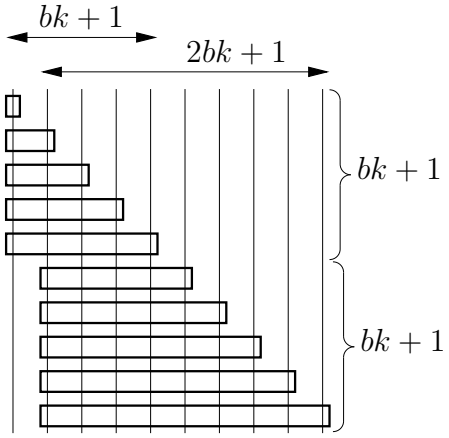


Figure 6.9: Illustration of Lemma 6.3: For every rectangle $r \in R$ in a reduced instance it holds that $\mathrm{rx}(r) \leq (bk+1) \cdot \mathrm{lx}(r)$, because there are at most $bk+1$ rectangles r' with $\mathrm{lx}(r') = \mathrm{lx}(r)$ and $\mathrm{rx}(r') \leq \mathrm{rx}(r)$. In the displayed example we assume that $bk = 4$.

rectangles can have $\mathrm{lx}(r') \geq \mathrm{lx}(r)$, and, hence, there exists $p \in \{\mathrm{rx}(r) - bk - 1, \ldots, \mathrm{rx}(r)-1\}$ such that there is a rectangle r' with $\mathrm{rx}(r') = p$ and $\mathrm{lx}(r') < \mathrm{lx}(r)$. But then, by the induction hypothesis, $\mathrm{rx}(r') \leq (bk+1) \cdot (\mathrm{lx}(r) - 1)$, which is a contradiction to $\mathrm{rx}(r') = p \geq \mathrm{rx}(r) - bk - 1$, since we assumed that $\mathrm{rx}(r) > (bk+1) \cdot \mathrm{lx}(r)$. This proves the lemma. □

See Figure 6.9 for an illustration of Lemma 6.3. Now we can prove that for every vertical line there is "short" rectangle to the right of this line:

Lemma 6.4. *In a reduced instance, for every $j \in \{1, \ldots, n-1\}$, there is a rectangle $r \in R$ with $\mathrm{lx}(r) > j$ and $\mathrm{rx}(r) \leq (bk+1) \cdot j + 1$.*

Proof. Assume for the sake of contradiction that there exists $j \in \{1, \ldots, n-1\}$ such that for every rectangle $r \in R$ with $\mathrm{lx}(r) > j$ it holds that $\mathrm{rx}(r) > (bk+1) \cdot j+1$. Consider a rectangle r' with $\mathrm{rx}(r') = (bk+1) \cdot j+1$. Such a rectangle exists by Observation 6.3. Then $\mathrm{lx}(r') \leq j$ due to our assumption. But by Lemma 6.3, this implies $\mathrm{rx}(r') \leq (bk+1) \cdot j$, a contradiction. □

Corollary 6.1. *Let $q \leq n$, and let $\{v_{j_1}, v_{j_2}, \ldots, v_{j_q}\} \subseteq V$ with $j_1 < j_2 < \ldots < j_q$ be a set of vertical lines stabbing all rectangles from R in a reduced instance. Then, for every $i \in \{1, \ldots, q\}$, it holds that $j_i \leq ((bk+1)^i - 1)/bk$.*

Proof. By induction on i. For $i = 1$, the statement holds because in any reduced instance there is a rectangle r with $\mathrm{lx}(r) = \mathrm{rx}(r) = 1$. Assume that the statement holds for $i-1$, that is, $j_{i-1} \leq ((bk+1)^{i-1} - 1)/bk$. By Lemma 6.4, there is a rectangle $r \in R$ with $\mathrm{lx}(r) > j_{i-1}$ and $\mathrm{rx}(r) \leq (bk+1) \cdot j_{i-1} + 1$. Clearly this rectangle is not stabbed by any line from $\{v_{j_1}, \ldots, v_{j_{i-1}}\}$ and therefore, we have $j_i \leq \mathrm{rx}(r) \leq (bk+1) \cdot j_{i-1} + 1 \leq ((bk+1)^i - 1)/bk$. □

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

The last ingredient for the problem kernel is the following easy observation.

Observation 6.7. *If an instance of the restricted variant of* Rectangle Stabbing *is a yes-instance, then there is a set $V' \subseteq V$ of at most bk vertical lines that intersect all rectangles in R.*

Proof. Replace every horizontal line h in an optimal solution by at most b vertical lines that intersect the rectangles intersected by h. □

Putting together Corollary 6.1 and Observation 6.7 finally yields our problem kernel.

Theorem 6.6. *The restricted variant of* Rectangle Stabbing*, where every horizontal line intersects at most b rectangles, has a problem kernel of size $O((bk+1)^{bk})$ and is, therefore, fixed-parameter tractable with respect to the combined parameters k and b.*

Proof. Given an instance of this restricted version, find in polynomial time the optimal number of vertical lines needed to intersect all rectangles. If the optimal solution size is greater than bk, report that the given instance is a *no*-instance. Otherwise, by Corollary 6.1, we know that every set of vertical lines $\{v_{j_1}, \ldots, v_{j_{bk}}\}$ that intersects all rectangles in R has $j_{bk} \leq ((bk+1)^{bk}-1)/bk$. If the given instance is a *yes*-instance, then R cannot contain any rectangle r with $\mathrm{lx}(r) > j_{bk}$. For every $j \in \{1, \ldots, j_{bk}\}$, however, there are at most $bk+1$ rectangles r with $\mathrm{lx}(r) = j$ due to Observation 6.6. Hence, if R contains more than $O((bk+1)^{bk})$ rectangles, report that the given instance is a *no*-instance. □

6.6 Conclusion

The problem d-Dimensional Rectangle Stabbing is well-studied from the approximation point of view. We have answered the thus naturally arising question whether the problem is fixed-parameter tractable with respect to the parameter $k =$ "solution size." While it is known that d-Dimensional Rectangle Stabbing can be approximated with a factor of d [GIK02, MSW05], we could prove that even for the case $d = 2$ the problem is W[1]-hard.

For some restrictions of the two-dimensional case, we could show fixed-parameter tractability; however, the corresponding algorithms are rather classification tools than being practical. It remains a future work to improve the running times obtained in this chapter.

Very recently, we have complemented the results presented here with some more fixed-parameter tractability results for Rectangle Stabbing [DFR09]:

- The restricted variant of Rectangle Stabbing where all rectangles in R are squares of the same width and the same height is W[1]-hard with respect to the parameter k.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

- Rectangle Stabbing can be solved in $(4k+1)^k \cdot n^{O(1)}$ time if all rectangles in R are squares of the same width and the same height that do not overlap. Thereby, two rectangles r_1, r_2 *overlap* if there exist a vertical line v and a horizontal line h that both intersect r_1 as well as r_2.
- 2-C1P-Set Cover, 2-XC1P-Set Cover, and Rectangle Stabbing are in W[1] with respect to the parameter k. This result can be generalized: For any constant d, d-C1P-Set Cover, d-XC1P-Set Cover, and d-Dimensional Rectangle Stabbing are in W[1] with respect to the parameter k.

The following questions, thus, remained open so far: First, we do not know the parameterized complexity of the perhaps most natural restriction of Rectangle Stabbing where no two rectangles from R overlap. Second, is there a polynomial-size problem kernel for Rectangle Stabbing when all rectangles in R are squares of the same size that do not overlap? Finally, do d-C1P-Set Cover, d-XC1P-Set Cover, and d-Dimensional Rectangle Stabbing belong to W[1] when parameterized by both k *and* d?

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Chapter 7

Conclusion

We studied a number of problems that are all closely connected to the consecutive-ones property of binary matrices. Let us briefly recapitulate the central findings presented in Chapters 3–6, the four main chapters of this work, and the methods leading to the results.

Finding small forbidden submatrices. We presented two polynomial-time algorithms for finding forbidden submatrices of almost minimum size and minimum size, respectively. The central idea of both algorithms is to reduce the matrix problem to a graph problem such that standard graph algorithms can be used as subroutines. Both algorithms are easy to implement, and, due to the small degrees of their polynomial running times, should be useful in practice.

Obtaining the C1P by row or column deletions. We exhibited a colorful landscape of different computational complexity results. Our main finding is a structural theorem, leading to approximation and fixed-parameter algorithms for the minimization problems Min-COS-R and Min-COS-C on sparse matrices.

Covering problems on input matrices with the C1P. We outlined the border between polynomial-time solvabability and NP-hardness for special cases of Minimum-Degree Hypergraph and Red-Blue Set Cover. The positive results of this chapter are polynomial-time algorithms for Minimum-Degree Hypergraph and Red-Blue Set Cover on input matrices with the C1P; these algorithms were obtained with integer linear programming and dynamic programming techniques.

Rectangle stabbing in d dimensions. Besides some fixed-parameter algorithms for special cases of 2-Dimensional Rectangle Stabbing, we proved the W[1]-hardness of d-Dimensional Rectangle Stabbing for every $d \geq 2$. To this end, we applied the new technique of reducing from the Multicolored Clique problem.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Which conclusions for upcoming problems can we draw from our findings? First, our studies in Chapters 5 and 6 for covering problems on matrices being "close" to having the C1P disclosed the hardness even in the case of very restricted problem variants. In contrast, there exist approximation algorithms and heuristics for Set Cover on matrices with few blocks of 1s per row [MSW05], and Set Cover is fixed-parameter tractable [MSW05, MW04] for the parameter "distance between the first and the last 1 in every row." Altogether, these observations confirm the "golden rule" that, in order to develop or to select an algorithm for a particular application, carefully identifying the parameters and restrictions that are specific for this application is crucial [Hüf07, Chapter 6].

Second, even for very general covering problems, the C1P leads to polynomial-time solvability: integer linear programming techniques and dynamic programming can be applied. Due to the "linear" structure of the C1P, it is not surprising that dynamic programming is one of the techniques of choice. However, integer linear programs can lead to even faster polynomial-time algorithms, when flow or shortest path algorithms can be used to solve the resulting ILPs. Surprisingly, in the literature sometimes only the unimodularity of the coefficient matrices is exploited.

What remains to do? In the conclusions of the chapters, we already pointed out several directions for future research. We will not repeat these open questions here; instead, we close with another, more general issue. In Chapter 4 we have achieved approximation and fixed-parameter algorithms for deleting rows or columns from a given matrix such that the C1P is obtained. However, we have not established a connection between the findings of that chapter and, for example, the positive results of Chapter 5 concerning the polynomial-time solvability of covering problems on matrices with the C1P. It remains to be investigated whether our positive results from Chapter 4 can be combined with efficient algorithms for problems on matrices with the C1P such that, in the end, heuristics or even approximation algorithms are obtained for these problems on matrices that do *not* have the C1P.

Overall, this thesis may hopefully serve as basis for a fruitful future research towards effective algorithms for the problems considered here and their various applications.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Bibliography

[ABH98] Jonathan E. Atkins, Erik G. Boman, and Bruce Hendrickson. A spectral algorithm for seriation and the consecutive ones problem. *SIAM Journal on Computing*, 28(1):297–310, 1998. Cited on pages 1, 27, 79.

[ACE+08] Ernst Althaus, Stefan Canzar, Mark R. Emmett, Andreas Karrenbauer, Alan G. Marshall, Anke Meyer-Baese, and Huimin 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 '08)*, pages 1273–1277. ACM Press, 2008. Cited on pages 27, 48.

[ACF+04] Faisal N. Abu-Khzam, Rebecca L. Collins, Michael R. Fellows, Michael A. Langston, W. Henry Suters, and Christof T. Symons. Kernelization algorithms for the vertex cover problem: Theory and experiments. In *Proceedings of the 6th Workshop on Algorithm Engineering and Experiments (ALENEX '04)*, pages 62–69. ACM/SIAM, 2004. Cited on page 23.

[ACG+99] Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela, and Marco Protasi. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties.* Springer, 1999. Cited on pages 20, 21.

[ADP80] Giorgio Ausiello, Alessandro D'Atri, and Marco Protasi. Structure preserving reductions among convex optimization problems. *Journal of Computer and System Sciences*, 21(1):136–153, 1980. Cited on page 119.

[AF06] Faisal N. Abu-Khzam and Henning Fernau. Kernels: Annotated, proper and induced. In *Proceedings of the 2nd International Workshop on Parameterized and Exact Computation (IWPEC '06)*, volume 4169 of *LNCS*, pages 264–275. Springer, 2006. Cited on page 23.

[AFLS07] Faisal N. Abu-Khzam, Michael R. Fellows, Michael A. Langston, and W. Henry Suters. Crown structures for vertex cover kerneliza-

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

tion. *Theory of Computing Systems*, 41(3):411–430, 2007. Cited on page 23.

[AM96] Jonathan E. Atkins and Martin Middendorf. On physical mapping and the consecutive ones property for sparse matrices. *Discrete Applied Mathematics*, 71(1–3):23–40, 1996. Cited on pages 1, 27, 44, 64, 79, 80, 142.

[AMO93] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice Hall, 1993. Cited on pages 54, 58.

[Ans99] Kurt M. Anstreicher. Linear programming in $O(\frac{n^3}{\ln n}L)$ operations. *SIAM Journal on Optimization*, 9(4):803–812, 1999. Cited on pages 49, 53, 54.

[AP06] Giorgio Ausiello and Vangelis Th. Paschos. Reductions, completeness and the hardness of approximability. *European Journal of Operational Research*, 172:719–739, 2006. Cited on page 20.

[APT79] Bengt Aspvall, Michael F. Plass, and Robert Endre Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, 8:121–123, 1979. Cited on page 123.

[AS98] Pankaj K. Agarwal and Micha Sharir. Efficient algorithms for geometric optimization. *ACM Computing Surveys*, 30(4):412–458, 1998. Cited on page 139.

[AYZ95] Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *Journal of the ACM*, 42(4):844–856, 1995. Cited on pages 93, 94, 95.

[Ben59] Seymour Benzer. On the topology of the genetic fine structure. *Proceedings of the National Academy of Sciences of the United States of America*, 45:1607–1620, 1959. Cited on page 32.

[Ber70] Claude Berge. Sur certains hypergraphes généralisant les graphes bipartites. In Paul Erdős, Alfréd Rhényi, and Vera T. Sós, editors, *Combinatorial Theory and its Applications I (Proceedings of the Colloquium on Combinatorial Theory and its Applications, 1969)*, pages 119–133. North-Holland, 1970. Cited on page 51.

[Ber72] Claude Berge. Balanced matrices. *Mathematical Programming*, 2:19–31, 1972. Cited on page 51.

[BG93] Jonathan F. Buss and Judy Goldsmith. Nondeterminism within P. *SIAM Journal on Computing*, 22(3):560–572, 1993. Cited on page 23.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

[BG94] Mihir Bellare and Shafi Goldwasser. The complexity of decision versus search. *SIAM Journal on Computing*, 23(1):97–119, 1994. Cited on page 15.

[BG02] Jørgen Bang-Jensen and Gregory Gutin. *Digraphs: Theory, Algorithms and Applications*. Springer, 2002. Cited on page 8.

[BGRS04] Vittorio Bilò, Vineet Goyal, R. Ravi, and Mohit Singh. On the crossing spanning tree problem. In *Proceedings of the 7th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX '04)*, volume 3122 of *LNCS*, pages 51–60. Springer, 2004. Cited on pages 44, 142.

[BGS02] Alexander Barvinok, Edward Kh. Gimadi, and Anatoliy I. Serdyukov. The Maximum Traveling Salesman Problem. In Gregory Gutin and Abraham P. Punnen, editors, *The Traveling Salesman Problem and its Variations*, pages 585–607. Kluwer Academic Publishers, 2002. Cited on page 91.

[BHY00] Jørgen Bang-Jensen, Jing Huang, and Anders Yeo. Convex-round and concave-round graphs. *SIAM Journal on Discrete Mathematics*, 13(2):179–193, 2000. Cited on page 32.

[BL76] Kellogg S. Booth and George S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *Journal of Computer and System Sciences*, 13:335–379, 1976. Cited on pages 33, 39, 42, 103, 104.

[BLS99] Andreas Brandstädt, Van Bang Le, and Jeremy P. Spinrad. *Graph Classes: A Survey*, volume 3 of *SIAM Monographs on Discrete Mathematics and Applications*. SIAM, 1999. Cited on pages 33, 51.

[BOR80] John J. Bartholdi, III, James B. Orlin, and H. Donald Ratliff. Cyclic scheduling via integer programs with circular ones. *Operations Research*, 28(5):1074–1085, 1980. Cited on pages 27, 58.

[BT76] I. Borosh and L. B. Treybig. Bounds on positive integral solutions of linear Diophantine equations. *Proceedings of the American Mathematical Society*, 55(2):299–304, 1976. Cited on page 49.

[CC95] Michele Conforti and Gérard Cornuéjols. Balanced $0, \pm 1$-matrices, bicoloring and total dual integrality. *Mathematical Programming*, 71:249–258, 1995. Cited on page 51.

[CCV06] Michele Conforti, Gérard Cornuéjols, and Kristina Vuskovic. Balanced matrices. *Discrete Mathematics*, 306(19–20):2411–2437, 2006. Cited on page 51.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

[CDKM00] Robert D. Carr, Srinivas Doddi, Goran Konjevod, and Madhav V. Marathe. On the red-blue set cover problem. In *Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '00)*, pages 345–353. ACM Press, 2000. Cited on pages 120, 136.

[CDKW05] Gruia Călinescu, Adrian Dumitrescu, Howard J. Karloff, and Peng-Jun Wan. Separating points by axis-parallel lines. *International Journal of Computational Geometry & Applications*, 15(6):575–590, 2005. Cited on pages 6, 140.

[Ces03] Marco Cesati. The Turing way to parameterized complexity. *Journal of Computer and System Sciences*, 67(4):654–685, 2003. Cited on page 25.

[CFJ04] Benny Chor, Michael R. Fellows, and David W. Juedes. Linear kernels in linear time, or how to save k colors in $O(n^2)$ steps. In *Proceedings of the 30th International Workshop on Graph-Theoretic Conecpts in Computer Science (WG '04)*, volume 3353 of *LNCS*, pages 257–269. Springer, 2004. Cited on page 23.

[CKJ01] Jianer Chen, Iyad A. Kanj, and Weijia Jia. Vertex Cover: Further observations and further improvements. *Journal of Algorithms*, 41(2):280–301, 2001. Cited on page 23.

[CKX06] Jianer Chen, Iyad A. Kanj, and Ge Xia. Improved parameterized upper bounds for Vertex Cover. In *Proceedings of the 31st International Symposium on Mathematical Foundations of Computer Science (MFCS '06)*, volume 4162 of *LNCS*, pages 238–249. Springer, 2006. Cited on page 22.

[Cla09] Millennium prize problems. Web page by The Clay Mathematics Institute of Cambridge, Massachusetts, USA, 2009. `http://www.claymath.org/millennium`. Cited on page 14.

[CLRS01] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 2nd edition, 2001. Cited on pages 7, 33, 56, 59, 68, 71, 73, 109, 119, 125.

[CLSZ07] Jianer Chen, Songjian Lu, Sing-Hoi Sze, and Fenghui Zhang. Improved algorithms for path, matching, and packing problems. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '07)*, pages 298–307. ACM/SIAM, 2007. Cited on page 94.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

[CM08] Jianer Chen and Jie Meng. On parameterized intractability: Hardness and completeness. *The Computer Journal*, 51(1):39–59, 2008. Cited on page 25.

[COR98] Thomas Christof, Marcus Oswald, and Gerhard Reinelt. Consecutive ones and a betweenness problem in computational biology. In *Proceedings of the 6th International Conference on Integer Programming and Combinatorial Optimization (IPCO '98)*, volume 1412 of *LNCS*, pages 213–228. Springer, 1998. Cited on page 27.

[COS98] Derek G. Corneil, Stephan Olariu, and Lorna Stewart. The ultimate interval graph recognition algorithm? (extended abstract). In *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '98)*, pages 175–180. ACM/SIAM, 1998. Cited on page 33.

[CP93] Nicos Christofides and José M. P. Paixão. Algorithms for large scale set covering problems. *Annals of Operations Research*, 43(5):259–277, 1993. Cited on pages 59, 119.

[CTF00] Alberto Caprara, Paolo Toth, and Matteo Fischetti. Algorithms for the set covering problem. *Annals of Operations Research*, 98:353–371, 2000. Cited on pages 59, 119.

[CV07] Kenneth L. Clarkson and Kasturi R. Varadarajan. Improved approximation algorithms for geometric set cover. *Discrete & Computational Geometry*, 37(1):43–58, 2007. Cited on page 139.

[DER89] Iain S. Duff, Albert M. Erisman, and John K. Reid. *Direct Methods for Sparse Matrices*. Monographs on Numerical Analysis. Oxford University Press, 1989. Cited on pages 64, 80.

[DF99] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity*. Springer, 1999. Cited on pages 21, 25, 59, 91, 120.

[DFH+06] Vida Dujmovic, Michael R. Fellows, Michael T. Hallett, Matthew Kitching, Giuseppe Liotta, Catherine McCartin, Naomi Nishimura, Prabhakar Ragde, Frances A. Rosamond, Matthew Suderman, Sue Whitesides, and David R. Wood. A fixed-parameter approach to 2-Layer Planarization. *Algorithmica*, 45(2):159–182, 2006. Cited on pages 81, 83, 113.

[DFR09] Michael Dom, Michael R. Fellows, and Frances A. Rosamond. Parameterized complexity of stabbing rectangles and squares in the plane. In *Proceedings of the 3rd International Workshop on Algorithms and Computation (WALCOM '09)*, volume 5431 of *LNCS*, pages 298–309. Springer, 2009. Cited on pages xiii, 161, 185.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

[DGH+10] Michael Dom, Jiong Guo, Falk Hüffner, Rolf Niedermeier, and Anke Truss. Fixed-parameter tractability results for feedback set problems in tournaments. *Journal of Discrete Algorithms*, 8(1):76–86, 2010. Cited on pages xi, 185.

[DGHN06] Michael Dom, Jiong Guo, Falk Hüffner, and Rolf Niedermeier. Error compensation in leaf power problems. *Algorithmica*, 44(4):363–381, 2006. Cited on pages xi, 185.

[DGHN08] Michael Dom, Jiong Guo, Falk Hüffner, and Rolf Niedermeier. Closest 4-Leaf Power is fixed-parameter tractable. *Discrete Applied Mathematics*, 156(18):3345–3361, 2008. Cited on pages xi, 185.

[DGN05] Michael Dom, Jiong Guo, and Rolf Niedermeier. Bounded Degree Closest k-Tree Power is NP-complete. In *Proceedings of the 11th Annual International Computing and Combinatorics Conference (COCOON '05)*, volume 3595 of *LNCS*, pages 757–766. Springer, 2005. Cited on pages xi, 185.

[DGN07] Michael Dom, Jiong Guo, and Rolf Niedermeier. Approximability and parameterized complexity of consecutive ones submatrix problems. In *Proceedings of the 4th Annual Conference on Theory and Applications of Models of Computation (TAMC '07)*, volume 4484 of *LNCS*, pages 680–691. Springer, 2007. Cited on pages xii, 81, 185.

[DGNW08] Michael Dom, Jiong Guo, Rolf Niedermeier, and Sebastian Wernicke. Red-blue covering problems and the consecutive ones property. *Journal of Discrete Algorithms*, 6(3):393–407, 2008. Cited on pages xii, 185.

[DHN08] Michael Dom, Falk Hüffner, and Rolf Niedermeier. Tiefensuche (Ariadne und Co.). In Berthold Vöcking, Helmut Alt, Martin Dietzfelbinger, Rüdiger Reischuk, Christian Scheideler, Heribert Vollmer, and Dorothea Wagner, editors, *Taschenbuch der Algorithmen*, eXamen.press, pages 61–73. Springer, 2008. In German language; English version to appear. Cited on page xi.

[Die05] Reinhard Diestel. *Graph Theory*, volume 173 of *Graduate Texts in Mathematics*. Springer, 3rd edition, 2005. Cited on page 8.

[DLS09] Michael Dom, Daniel Lokshtanov, and Saket Saurabh. Incompressibility through colors and IDs. In *Proceedings of the 36th International Colloquium on Automata, Languages, and Programming (ICALP '09)*, LNCS. Springer, 2009. To appear. Cited on pages xi, 185.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

[DLSV08] Michael Dom, Daniel Lokshtanov, Saket Saurabh, and Yngve Villanger. Capacitated domination and covering: A parameterized perspective. In *Proceedings of the 3rd International Workshop on Parameterized and Exact Computation (IWPEC '08)*, volume 5018 of *LNCS*, pages 78–90. Springer, 2008. Cited on pages xi, 185.

[DN07] Michael Dom and Rolf Niedermeier. The search for consecutive ones submatrices: Faster and more general. In *Proceedings of the 3rd Algorithms and Complexity in Durham (ACiD '07) Workshop*, volume 9 of *Texts in Algorithmics*, pages 43–54. College Publications, 2007. Cited on pages xii, 185.

[Dom04] Michael Dom. Fehler-Korrektur bei Leaf-Root-Problemen (Error correction in leaf root problems). Diploma thesis, Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, Germany, 2004. In German language. Cited on page 185.

[Dom07] Michael Dom. Compact routing. In Dorothea Wagner and Roger Wattenhofer, editors, *Algorithms for Sensor and Ad Hoc Networks*, volume 4621 of *LNCS*, pages 187–202. Springer, 2007. Cited on page xi.

[Dom08] Michael Dom. Set Cover with almost consecutive ones. In Ming-Yang Kao, editor, *Encyclopedia of Algorithms*, pages 832–834. Springer, 2008. Cited on page xi.

[DS05] Irit Dinur and Shmuel Safra. On the hardness of approximating minimum vertex cover. *Annals of Mathematics*, 162(1):439–485, 2005. Cited on page 91.

[DS08] Michael Dom and Somnath Sikdar. The parameterized complexity of the rectangle stabbing problem and its variants. In *Proceedings of the 2nd International Frontiers of Algorithmics Workshop (FAW '08)*, volume 5059 of *LNCS*, pages 288–299. Springer, 2008. Cited on pages xiii, 185.

[ELR+08] Guy Even, Retsef Levi, Dror Rawitz, Baruch Schieber, Shimon Shahar, and Maxim Sviridenko. Algorithms for capacitated rectangle stabbing and lot sizing with joint set-up costs. *ACM Transactions on Algorithms*, 4(3), Article 34, 2008. Cited on pages 33, 141.

[ES93] Elaine M. Eschen and Jeremy Spinrad. An $O(n^2)$ algorithm for circular-arc graph recognition. In *Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '93)*, pages 128–137. ACM/SIAM, 1993. Cited on page 33.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

[Fei98] Uriel Feige. A threshold of ln n for approximating Set Cover. *Journal of the ACM*, 45(4):634–652, 1998. Cited on pages 59, 119.

[Fel03] Michael R. Fellows. Blow-ups, win/win's, and crown rules: Some new directions in FPT. In *Proceedings of the 29th International Workshop on Graph-Theoretic Conecpts in Computer Science (WG '03)*, volume 2880 of *LNCS*, pages 1–12. Springer, 2003. Cited on page 23.

[Fel07] Michael R. Fellows, September 2007. Personal communication. Cited on page 145.

[Fer05a] Henning Fernau. Parameterized algorithmics: A graph-theoretic approach. Habilitationsschrift, Wilhelm-Schickard-Institut für Informatik, Universität Tübingen, Germany, 2005. Cited on pages 81, 83, 111, 113.

[Fer05b] Henning Fernau. Two-layer planarization: Improving on parameterized algorithmics. In *Proceedings of the 31st Conference on Current Trends in Theory and Practice of Informatics (SOFSEM '05)*, volume 3381 of *LNCS*, pages 137–146. Springer, 2005. Cited on pages 81, 83, 113.

[Fer08] Henning Fernau. Parameterized algorithmics for linear arrangement problems. *Discrete Applied Mathematics*, 156(17):3166–3177, 2008. Cited on pages 81, 83, 111.

[FG65] Delbert R. Fulkerson and O. A. Gross. Incidence matrices and interval graphs. *Pacific Journal of Mathematics*, 15(3):835–855, 1965. Cited on pages 33, 35, 39.

[FG06] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory.* Springer, 2006. Cited on pages 21, 25.

[FGS96] Michele Flammini, Giorgio Gambosi, and Sandro Salomone. Interval routing schemes. *Algorithmica*, 16(6):549–568, 1996. Cited on pages 44, 142.

[FHO74] Delbert R. Fulkerson, Alan J. Hoffman, and Rosa Oppenheim. On balanced matrices. *Mathematical Programming Study*, 1:120–132, 1974. Cited on page 51.

[FHRV09] Michael R. Fellows, Danny Hermelin, Frances A. Rosamond, and Stéphane Vialette. On the parameterized complexity of multiple-interval graph problems. *Theoretical Computer Science*, 410(1):53–61, 2009. Cited on page 145.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

[Fis85] Peter C. Fishburn. *Interval Orders and Interval Graphs*. Wiley, 1985. Cited on page 35.

[FMZ06] Tomás Feder, Rajeev Motwani, and An Zhu. k-connected spanning subgraphs of low degree. Technical Report TR06-041, Electronic Colloquium on Computational Complexity (ECCC), 2006. Cited on page 120.

[Gar07] Frédéric Gardi. The Roberts characterization of proper and unit interval graphs. *Discrete Mathematics*, 307(22):2906–2908, 2007. Cited on page 33.

[Gav74] Fanica Gavril. Algorithms on circular-arc graphs. *Networks*, 4:357–369, 1974. Cited on page 35.

[GGKS95] Paul W. Goldberg, Martin Charles Golumbic, Haim Kaplan, and Ron Shamir. Four strikes against physical mapping of DNA. *Journal of Computational Biology*, 2(1):139–152, 1995. Cited on pages 1, 27, 44, 79, 142.

[Gho62] Alain Ghouila-Houri. Caractérisation des matrices totalement unimodulaires. *Comptes Rendus de l' Académie des Sciences, Paris*, 254:1192–1194, 1962. Cited on page 53.

[GIK02] Daya Ram Gaur, Toshihide Ibaraki, and Ramesh Krishnamurti. Constant ratio approximation algorithms for the rectangle stabbing problem and the rectilinear partitioning problem. *Journal of Algorithms*, 43(1):138–152, 2002. Cited on pages 62, 140, 141, 161.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979. Cited on pages 14, 17, 44, 81, 87, 137, 142, 157.

[GK03] Fred W. Glover and Gary A. Kochenberger, editors. *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science*. Springer, 2003. Cited on page 18.

[GKW08] Panos Giannopoulos, Christian Knauer, and Sue Whitesides. Parameterized complexity of geometric problems. *The Computer Journal*, 51(3):372–384, 2008. Cited on page 139.

[GLP08] GNU Linear Programming Kit. Homepage of the GLPK software package, 2008. `http://www.gnu.org/software/glpk`. Cited on page 48.

[GN07] Jiong Guo and Rolf Niedermeier. Invitation to data reduction and problem kernelization. *ACM SIGACT News*, 38(1):31–45, 2007. Cited on page 22.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

[Gol04] Martin C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*, volume 57 of *Annals of Discrete Mathematics*. Elsevier B. V., 2nd edition, 2004. First edition Academic Press, 1980. Cited on pages 33, 51.

[GS78] Joachim von zur Gathen and Malte Sieveking. A bound on solutions of linear integer equations and inequalities. *Proceedings of the American Mathematical Society*, 72:155–158, 1978. Cited on page 49.

[Had02] Salim Haddadi. A note on the NP-hardness of the consecutive block minimization problem. *International Transactions in Operational Research*, 9(6):775–777, 2002. Cited on pages 44, 142.

[Haj57] G. Hajös. Über eine Art von Graphen. *Internationale Mathematische Nachrichten*, 11, Problem 65, 1957. In German language. Cited on page 32.

[Haj00] MohammadTaghi Hajiaghayi. Consecutive ones property. Manuscript, School of Computer Science, University of Waterloo, Canada, 2000. Cited on pages 80, 81.

[Hås99] Johan Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, 182:105–142, 1999. Cited on pages 21, 88.

[HG02] MohammadTaghi Hajiaghayi and Yashar Ganjali. A note on the consecutive ones submatrix problem. *Information Processing Letters*, 83(3):163–166, 2002. Cited on pages 34, 79, 80, 81.

[HK56] Alan J. Hoffman and Joseph B. Kruskal. Integral boundary points of convex polyhedra. In H. W. Kuhn and A. W. Tucker, editors, *Linear Inequalities and Related Systems*, pages 223–246. Princeton University Press, 1956. Cited on page 53.

[HL06] Dorit S. Hochbaum and Asaf Levin. Cyclical scheduling and multishift scheduling: Complexity and approximation algorithms. *Discrete Optimization*, 3(4):327–340, 2006. Cited on page 27.

[HL08] Salim Haddadi and Zoubir Layouni. Consecutive block minimization is 1.5-approximable. *Information Processing Letters*, 108:132–135, 2008. Cited on pages 44, 142.

[HM91] Refael Hassin and Nimrod Megiddo. Approximation algorithms for hitting objects with straight lines. *Discrete Applied Mathematics*, 30:29–42, 1991. Cited on pages 27, 140, 141, 155, 156.

[HM99] Wen-Lian Hsu and Tze-Heng Ma. Fast and simple algorithms for recognizing chordal comparability graphs and interval graphs. *SIAM Journal on Computing*, 28(3):1004–1020, 1999. Cited on page 33.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

[HM03] Wen-Lian Hsu and Ross M. McConnell. PC trees and circular-ones arrangements. *Theoretical Computer Science*, 296(1):99–116, 2003. Cited on pages 42, 99, 100.

[HMPV00] Michel Habib, Ross M. McConnell, Christophe Paul, and Laurent Viennot. Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. *Theoretical Computer Science*, 234(1–2):59–84, 2000. Cited on pages 33, 43.

[HNW08] Falk Hüffner, Rolf Niedermeier, and Sebastian Wernicke. Techniques for practical fixed-parameter algorithms. *The Computer Journal*, 51(1):7–25, 2008. Cited on page 97.

[Hoc97] Dorit S. Hochbaum, editor. *Approximation Algorithms for NP-hard Problems*. PWS Publishing Company, 1997. Cited on page 19.

[HR00] Refael Hassin and Shlomi Rubinstein. Better approximations for max TSP. *Information Processing Letters*, 75(4):181–186, 2000. Cited on page 91.

[Hsu92] Wen-Lian Hsu. A simple test for interval graphs. In *Proceedings of the 18th International Workshop on Graph-Theoretic Conecpts in Computer Science (WG '92)*, volume 657 of *LNCS*, pages 11–16. Springer, 1992. Cited on page 33.

[Hsu95] Wen-Lian Hsu. $O(M * N)$ algorithms for the recognition and isomorphism problems on circular-arc graphs. *SIAM Journal on Computing*, 24(3):411–439, 1995. Cited on page 33.

[Hsu97] Wen-Lian Hsu. On physical mapping algorithms – an error-tolerant test for the consecutive ones property. In *Proceedings of the 3rd Annual International Computing and Combinatorics Conference (COCOON '97)*, volume 1276 of *LNCS*, pages 242–250. Springer, 1997. Cited on page 44.

[Hsu02] Wen-Lian Hsu. A simple test for the consecutive ones property. *Journal of Algorithms*, 43(1):1–16, 2002. Cited on page 39.

[HT02] Dorit S. Hochbaum and Paul A. Tucker. Minimax problems with bitonic matrices. *Networks*, 40(3):113–124, 2002. Cited on pages 27, 58.

[HU79] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979. Cited on page 49.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

[Hüf07] Falk Hüffner. *Algorithms and Experiments for Parameterized Approaches to Hard Graph Problems.* PhD thesis, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2007. Cited on page 164.

[HWZ08] Falk Hüffner, Sebastian Wernicke, and Thomas Zichner. Algorithm engineering for color-coding with applications to signaling pathway detection. *Algorithmica*, 52(2):114–132, 2008. Cited on pages 93, 94, 95, 96.

[ISG08] Graphclasses in ISGCI. Online database, 2008. `http://wwwteo.informatik.uni-rostock.de/isgci/classes.cgi`. Cited on page 33.

[JM08] Klaus Jansen and Marian Margraf. *Approximative Algorithmen und Nichtapproximierbarkeit.* Gruyter, 2008. In German language. Cited on page 21.

[Joh74] David S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9(3):256–278, 1974. Cited on page 19.

[Khu02] Samir Khuller. The Vertex Cover problem. *SIGACT News*, 33(2):31–33, 2002. Cited on page 23.

[KM78] Ravindran Kannan and C. L. Monma. On the computational complexity of integer programming problems. In R. Henn, B. Korte, and W. Oettli, editors, *Optimization and Operations Research*, volume 157 of *Lecture Notes in Economics and Mathematical Systems*, pages 161–172. Springer, 1978. Cited on page 49.

[KM89] Norbert Korte and Rolf H. Möhring. An incremental linear-time algorithm for recognizing interval graphs. *SIAM Journal on Computing*, 18(1):68–81, 1989. Cited on pages 33, 40.

[KMMS06] Dieter Kratsch, Ross M. McConnell, Kurt Mehlhorn, and Jeremy Spinrad. Certifying algorithms for recognizing interval graphs and permutation graphs. *SIAM Journal on Computing*, 36(2):326–353, 2006. Cited on pages 33, 41.

[KMN05] Matthew J. Katz, Joseph S. B. Mitchell, and Yuval Nir. Orthogonal segment stabbing. *Computational Geometry*, 30(2):197–205, 2005. Cited on pages 122, 139.

[KMRR06] Joachim Kneis, Daniel Mölle, Stefan Richter, and Peter Rossmanith. Divide-and-color. In *Proceedings of the 32nd International Workshop on Graph-Theoretic Conecpts in Computer Science (WG '06)*, volume 4271 of *LNCS*, pages 58–67. Springer, 2006. Cited on page 94.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

[KMSV98] Sanjeev Khanna, Rajeev Motwani, Madhu Sudan, and Umesh V. Vazirani. On syntactic versus computational views of approximability. *SIAM Journal on Computing*, 28(1):164–191, 1998. Cited on page 20.

[KN96] Matthew J. Katz and Frank Nielsen. On piercing sets of objects. In *Proceedings of the 22nd Annual ACM Symposium on Computational Geometry (SOCG '96)*, pages 113–121. ACM Press, 1996. Cited on page 139.

[KN06] Haim Kaplan and Yahav Nussbaum. A simpler linear-time recognition of circular-arc graphs. In *Proceedings of the 10th Scandinavian Workshop on Algorithm Theory (SWAT '06)*, volume 4059 of *LNCS*, pages 41–52. Springer, 2006. Cited on page 33.

[Köh04] Ekkehard Köhler. Recognizing graphs without asteroidal triples. *Journal of Discrete Algorithms*, 2(4):439–452, 2004. Cited on page 63.

[Kou77] Lawrence T. Kou. Polynomial complete consecutive information retrieval problems. *SIAM Journal on Computing*, 6(1):67–75, 1977. Cited on page 27.

[KR02] Subhash Khot and Venkatesh Raman. Parameterized complexity of finding subgraphs with hereditary properties. *Theoretical Computer Science*, 289(2):997–1008, 2002. Cited on page 91.

[KR08] Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within $2-\epsilon$. *Journal of Computer and System Sciences*, 74(3):335–349, 2008. Cited on page 82.

[KRW+05] Fabian Kuhn, Pascal von Rickenbach, Roger Wattenhofer, Emo Welzl, and Aaron Zollinger. Interference in cellular networks: The minimum membership set cover problem. In *Proceedings of the 11th Annual International Computing and Combinatorics Conference (COCOON '05)*, volume 3595 of *LNCS*, pages 188–198. Springer, 2005. Cited on pages 2, 120, 121.

[KS96] Haim Kaplan and Ron Shamir. Physical maps and interval sandwich problems: Bounded degrees help. In *Proceedings of the 4th Israeli Symposium on Theory of Computing and Systems (ISTCS '96)*, pages 195–201, 1996. Cited on pages 64, 80.

[KS01] Sofia Kovaleva and Frits C. R. Spieksma. Approximation of a geometric set covering problem. In *Proceedings of the 12th International Symposium on Algorithms and Computation (ISAAC '01)*, volume 2223 of *LNCS*, pages 493–501. Springer, 2001. Cited on pages 27, 141, 155, 157.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

[KS06] Sofia Kovaleva and Frits C. R. Spieksma. Approximation algorithms for rectangle stabbing and interval stabbing problems. *SIAM Journal on Discrete Mathematics*, 20(3):748–768, 2006. Cited on pages 141, 155, 157.

[KSPS02] Farinaz Koushanfar, Sasha Slijepcevic, Miodrag Potkonjak, and Alberto Sangiovanni-Vincentelli. Error-tolerant multimodal sensor fusion. In *Proceedings of the IEEE CAS Workshop on Wireless Communications and Networking*. IEEE CAS, 2002. Cited on pages 4, 140.

[KW97] Josef Kallrath and John M. Wilson. *Business Optimization Using Mathematical Programming*. MacMillan Press, 1997. Cited on page 48.

[LB62] C. G. Lekkerkerker and J. Ch. Boland. Representation of a finite graph by a set of intervals on the real line. *Fundamenta Mathematicae*, 51:45–64, 1962. Cited on page 37.

[LBI+01] Giuseppe Lancia, Vineet Bafna, Sorin Istrail, Ross Lippert, and Russell Schwartz. SNPs problems, complexity, and algorithms. In *Proceedings of the 9th Annual European Symposium on Algorithms (ESA '01)*, volume 2161 of *LNCS*, pages 182–193. Springer, 2001. Cited on page 33.

[LH03] Wei-Fu Lu and Wen-Lian 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. Cited on pages 1, 27, 44, 79.

[LLS75] Richard E. Ladner, Nancy A. Lynch, and Alan L. Selman. A comparison of polynomial time reducibilities. *Theoretical Computer Science*, 1(2):103–123, 1975. Cited on page 13.

[LM05] Stefan Langerman and Pat Morin. Covering things with things. *Discrete & Computational Geometry*, 33(4):717–729, 2005. Cited on page 139.

[Lov75] Laszlo Lovàsz. On the ratio of optimal fractional and integral covers. *Discrete Mathematics*, 13:383–390, 1975. Cited on page 19.

[LY93] Carsten Lund and Mihalis Yannakakis. The approximation of maximum subgraph problems. In *Proceedings of the 20st International Colloquium on Automata, Languages, and Programming (ICALP '93)*, volume 700 of *LNCS*, pages 40–51. Springer, 1993. Cited on page 91.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

[McC03] Ross M. McConnell. Linear-time recognition of circular-arc graphs. *Algorithmica*, 37(2):93–147, 2003. Cited on page 33.

[McC04] Ross M. McConnell. A certifying algorithm for the consecutive-ones property. In *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '04)*, pages 768–777. ACM Press, 2004. Cited on pages xii, 38, 40, 41, 44, 45, 46.

[MF04] Zbigniew Michalewicz and David B. Fogel. *How to Solve it: Modern Heuristics*. Springer, 2nd edition, 2004. Cited on page 18.

[MG06] Jiří Matoušek and Bernd Gärtner. *Understanding and Using Linear Programming*. Universitext. Springer, 2006. Cited on page 48.

[MPT98] João Meidanis, Oscar Porto, and Guilherme P. Telles. On the consecutive ones property. *Discrete Applied Mathematics*, 88:325–354, 1998. Cited on page 40.

[MR95] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995. Cited on page 26.

[MSW05] Steffen Mecke, Anita Schöbel, and Dorothea Wagner. Station location – complexity and approximation. In *Proceedings of the 5th Workshop on Algorithmic Methods and Models for Optimization of Railways (ATMOS '05)*. IBFI Dagstuhl, Germany, 2005. Cited on pages 3, 27, 33, 61, 121, 122, 140, 141, 158, 161, 164.

[MT82] Nimrod Megiddo and Arie Tamir. On the complexity of locating linear facilities in the plane. *Operations Research Letters*, 1:194–197, 1982. Cited on page 139.

[MU05] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005. Cited on page 26.

[MW04] Steffen Mecke and Dorothea Wagner. Solving geometric covering problems by data reduction. In *Proceedings of the 12th Annual European Symposium on Algorithms (ESA '04)*, volume 3221 of *LNCS*, pages 760–771. Springer, 2004. Cited on pages 3, 27, 33, 61, 121, 122, 164.

[Nie06] Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006. Cited on pages 21, 25, 97.

[NT75] George L. Nemhauser and Leslie E. Trotter. Vertex packings: Structural properties and algorithms. *Mathematical Programming*, 8:232–248, 1975. Cited on page 23.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

[Nus97] Doron Nussbaum. Rectilinear p-piercing problems. In *Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation (ISSAC '97)*, pages 316–323. ACM Press, 1997. Cited on page 139.

[NW88] George L. Nemhauser and Laurence A. Wolsey. *Integer and Combinatorial Optimization.* Discrete Mathematics and Optimization. Wiley, 1988. Cited on pages 33, 53, 54, 121.

[OR00] Marcus Oswald and Gerhard Reinelt. Polyhedral aspects of the consecutive ones problem. In *Proceedings of the 6th Annual International Computing and Combinatorics Conference (COCOON '00)*, volume 1858 of *LNCS*, pages 373–382. Springer, 2000. Cited on page 27.

[OR03a] Marcus Oswald and Gerhard Reinelt. Constructing new facets of the consecutive ones polytope. In *Proceedings of the 5th International Workshop on Combinatorial Optimization—"Eureka, You Shrink!" (Aussois '01)*, volume 2570 of *LNCS*, pages 147–157, 2003. Cited on page 27.

[OR03b] Marcus Oswald and Gerhard Reinelt. The weighted consecutive ones problem for a fixed number of rows or columns. *Operations Research Letters*, 31(3):350–356, 2003. Cited on page 87.

[Pap94] Christos M. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994. Cited on pages 10, 16, 21, 26, 130.

[Pap97] Christos H. Papadimitriou. NP-completeness: A retrospective. In *Proceedings of the 24th International Colloquium on Automata, Languages, and Programming (ICALP '97)*, volume 1256 of *LNCS*, pages 2–6. Springer, 1997. Cited on pages 14, 17.

[PS98] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Dover Publications Inc., corrected edition, 1998. First edition Prentice Hall, 1982. Cited on pages 18, 48.

[PW00] Florian A. Potra and Stephen J. Wright. Interior-point methods. *Journal of Computational and Applied Mathematics*, 124:281–302, 2000. Cited on page 49.

[PY91] Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43(3):425–440, 1991. Cited on pages 20, 137.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

[Rob69] Fred. S. Roberts. Indifference graphs. In Frank Harary, editor, *Proof Techniques in Graph Theory*, pages 139–146. Academic Press, 1969. Cited on pages 33, 35.

[Ros06] Kenneth H. Rosen. *Discrete Mathematics and Its Applications.* McGraw Hill, 6th edition, 2006. Cited on page 7.

[RS04] Nikolaus Ruf and Anita Schöbel. Set covering with almost consecutive ones property. *Discrete Optimization*, 1(2):215–228, 2004. Cited on pages 3, 27, 33, 61, 121, 122.

[Sch86] Alexander Schrijver. *Theory of Linear and Integer Programming.* Wiley, 1986. Cited on pages 48, 51, 54, 124.

[Seg99] Michael Segal. On piercing sets of axis-parallel rectangles and rings. *International Journal of Computational Geometry & Applications*, 9(3):219–233, 1999. Cited on page 139.

[Ser84] A. I. Serdyukov. An algorithm with an estimate for the travelling salesman problem of the maximum. *Upravlyaemye Sistemy*, 25:80–86, 1984. In Russian language. Cited on pages 91, 92.

[Sie96] Gerard Sierksma. *Linear and Integer Programming.* Marcel Dekker, 1996. Cited on page 48.

[SKD06] Harald Sack, Uwe Krüger, and Michael Dom. A knowledge base on NP-complete decision problems and its application in bibliographic search. In *XML-Tage 2006*, September 2006. Cited on pages xi, 185.

[Sud05] Matthew Suderman. *Layered Graph Drawing.* PhD thesis, School of Computer Science, McGill University Montréal, Canada, 2005. Cited on pages 81, 83, 113.

[SW96] Micha Sharir and Emo Welzl. Rectilinear and polygonal p-piercing and p-center problems. In *Proceedings of the 22nd Annual ACM Symposium on Computational Geometry (SOCG '96)*, pages 122–132. ACM Press, 1996. Cited on page 139.

[SW05] Matthew Suderman and Sue Whitesides. Experiments with the fixed-parameter approach for two-layer planarization. *Journal of Graph Algorithms and Applications*, 9(1):149–163, 2005. Cited on pages 81, 83, 113.

[Tru78] Klaus Truemper. On balanced matrices and Tutte's characterization of regular matroids. Manuscript, 1978. Cited on page 51.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

[Tuc71] Alan C. Tucker. Matrix characterizations of circular-arc graphs. *Pacific Journal of Mathematics*, 2(39):535–545, 1971. Cited on pages 29, 30, 34, 35.

[Tuc72] Alan C. Tucker. A structure theorem for the consecutive 1's property. *Journal of Combinatorial Theory. Series B*, 12:153–162, 1972. Cited on pages 28, 29, 31, 34, 36, 37, 38, 64, 66, 67, 79, 80, 82, 85.

[Tuc80] Alan C. Tucker. An efficient test for circular-arc graphs. *SIAM Journal on Computing*, 9(1):1–24, 1980. Cited on page 33.

[TZ07] Jinsong Tan and Louxin Zhang. The consecutive ones submatrix problem for sparse matrices. *Algorithmica*, 48(3):287–299, 2007. Cited on pages 34, 64, 79, 80, 81, 82, 83, 85, 86, 88, 91, 94.

[Vaz01] Vijay V. Vazirani. *Approximation Algorithms*. Springer, 2001. Cited on page 21.

[Vel85] Marinus Veldhorst. Approximation of the consecutive ones matrix augmentation problem. *SIAM Journal on Computing*, 14(3):709–729, 1985. Cited on page 87.

[VW62] Arthur F. Veinott and H. M. Wagner. Optimal capacity scheduling. *Operations Research*, 10:518–547, 1962. Cited on pages 27, 33, 54, 121.

[Wes01] Douglas B. West. *Introduction to Graph Theory*. Prentice Hall, 2nd edition, 2001. Cited on page 8.

[WLZ07] Rui Wang, Francis C. M. Lau, and Yingchao Zhao. Hamiltonicity of regular graphs and blocks of consecutive ones in symmetric matrices. *Discrete Applied Mathematics*, 155(17):2312–2320, 2007. Cited on pages 44, 142.

[Woe03] Gerhard J. Woeginger. Exact algorithms for NP-hard problems: A survey. In *Proceedings of the 5th International Workshop on Combinatorial Optimization—"Eureka, You Shrink!" (Aussois '01)*, volume 2570 of *LNCS*, pages 185–208. Springer, 2003. Cited on page 7.

[WR00] Stephan Weis and Rüdiger Reischuk. The complexity of physical mapping with strict chimerism. In *Proceedings of the 6th Annual International Computing and Combinatorics Conference (COCOON '00)*, volume 1858 of *LNCS*, pages 383–395. Springer, 2000. Cited on pages 1, 27, 44, 64, 79, 80, 142.

[XX07] Guang Xu and Jinhui Xu. Constant approximation algorithms for rectangle stabbing and related problems. *Theory of Computing Systems*, 40(2):187–204, 2007. Cited on page 141.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Erklärung

Hiermit erkläre ich,

- dass mir die geltende Promotionsordnung der Fakultät bekannt ist;
- dass ich die Dissertation selbst angefertigt und alle von mir benutzten Hilfsmittel, persönlichen Mitteilungen und Quellen in meiner Arbeit angegeben habe;
- dass ich die Hilfe eines Promotionsberaters nicht in Anspruch genommen habe und dass Dritte weder unmittelbar noch mittelbar geldwerte Leistungen von mir für Arbeiten erhalten haben, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen;
- dass ich die Dissertation noch nicht als Prüfungsarbeit für eine staatliche oder andere wissenschaftliche Prüfung eingereicht habe;
- dass ich weder die gleiche noch eine in wesentlichen Teilen ähnliche noch eine andere Abhandlung bei einer anderen Hochschule als Dissertation eingereicht habe.

Jena, Juli 2008 *Michael Dom*

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

Lebenslauf

Michael Dom

[redacted]

[redacted] [Dom04], [redacted]

Begutachtete Publikationen (bei Veröffentlichungen mit sowohl Konferenz- als auch Zeitschriftenversion ist nur die Zeitschriftenversion aufgeführt):

[DFR09, DGHN06, DGH$^+$10, DGHN08, DGN05, DGN07, DGNW08, DLS09, DLSV08, DN07, DS08, SKD06]

Jena, Juli 2008 / März 2009 *Michael Dom*

Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property.* Revised version, 02. 06. 2010. Original publication: Dissertation, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.