INAUGURAL-DISSERTATION
zur
Erlangung der Doktorwürde
der
Naturwissenschaftlich-Mathematischen Gesamtfakultät
der
Ruprecht-Karls-Universität
Heidelberg

vorgelegt von
Diplom-Mathematiker Marcus Oswald
aus Sinsheim

Tag der mündlichen Prüfung: 28.5.2003

Weighted Consecutive Ones Problems

Gutachter: Prof. Dr. Gerhard Reinelt
Prof. Dr. Dr. h.c. Hans-Georg Bock

Im Zeichen des Löwen

Contents

Introduction

We want to start our work on weighted consecutive ones problems by giving an example from the field of developmental psychology. For more details on this application see Roberts [Rob76].

Usually the development of each child has several stages, crawling, sitting up, pulling itself up to a standing position, etc. These stages correspond to a certain interval of time and can overlap each other. One hypothesis of psychologists studying the development of children is that the pattern of stages passed through by a child is common to all children, though the time each child stays in a certain stage differs. To check this the psychologists observe a lot of children and note for each child if it has the traits of the given stages. The results can be written into a 0/1 matrix with the children as the columns and the stages as the rows. A "1" is written as entry if and only if the corresponding child shows the trait of the corresponding stage. Now, if the above hypothesis were true, then ordering the children according to their (unknown) developmental age would bring the matrix into a form, where in each row the "1"-entries form a consecutive series corresponding to the time interval of that stage.

This property of the matrix we call the *consecutive ones property for rows*. Testing if the given 0/1 matrix has this property can be done very efficiently by the *PQ-tree algorithm* of Booth and Lueker [BL76]. But usually "real-world" experiments are influenced by errors. Just imagine that a child is already in the stage of sitting up, but feels only like crawling at the time it is observed. Therefore we are interested in determining this errors, i.e. finding the entries of the matrix that have to be changed to obtain a matrix with the consecutive ones property. Or more precisely, given a changing-cost for each entry of the matrix, we are interested in the cheapest way to obtain a consecutive ones matrix. This problem is called the *weighted consecutive ones problem (WC1P)* and is known to be $\mathcal{NP}$-hard (see Booth [Boo75] and Papadimitriou [Pap76]). Tackling this problem is the purpose of this thesis.

The method of choice for solving hard combinatorial problems to optimality has become the *branch-and-cut* algorithm. The branch-and-cut approach was introduced in 1984 by Grötschel, Jünger and Reinelt [GJR84] for solving the linear ordering problem. In the meantime branch-and-cut algorithms have been implemented for many combinatorial optimization problems. To get an idea of the power of the branch-and-cut method see for example the work of Applegate, Bixby, Chvátal, and Cook [ABCC01] who were able to solve the traveling salesman problem "`d15112`" (available at Reinelt's TSPLIB [Rei91]) through 15112 cities in Germany to optimality. Note that this problem has about $7,3 \cdot 10^{56592}$

feasible solutions!

Fortunately nowadays it is not necessary to implement a branch-and-cut approach from scratch. There are software frameworks like ABACUS [Thi95] which make that branch-and-cut parts available to the user that are not dependent on the current problem. What remains to be implemented are problem-specific routines such as a feasibility test, an integer programming formulation, primal heuristics, and separation procedures. For the latter theoretical knowledge of the associated polytope, which is defined as the convex hull over all incidence vectors corresponding to feasible solutions, is very helpful.

Outline of the thesis

This work is structured as follows. In chapter 1 we review basic definitions and results from linear algebra, polyhedral theory, graph theory, and complexity theory. Furthermore the branch-and-cut algorithm for solving combinatorial optimization problems is introduced. This preliminary chapter provides the reader with those concepts and notations required in the subsequent chapters, but is not meant to be comprehensive.

Chapter 2 introduces definitions, notations, and complexity results concerning consecutive ones problems. Booth's and Lueker's PQ-tree algorithm is presented, which provides a test, whether a given 0/1-matrix has the consecutive ones property for rows or for columns. The time complexity of this algorithm is linear in the number of entries of the matrix. The weighted consecutive ones problem is defined and a proof of its $\mathcal{NP}$-hardness according to Booth [Boo75] is reproduced. Furthermore we show that this problem becomes linearly solvable if the column permutation is fixed. In chapter 3 we address ourselves to polyhedral investigations of the problem. After defining the consecutive ones polytope $P^{m,n}_{\mathrm{C1R}}$ and proving some basic properties we present different ways to obtain facet-defining inequalities for $P^{m,n}_{\mathrm{C1R}}$. An IP formulation of the WC1P is given consisting only of facets for the corresponding polytope. This chapter will finish with proving that the number of facets of $P^{m,n}_{\mathrm{C1R}}$ grows only polynomially if the number of rows m or the number of columns n is fixed.

In chapter 4 the simultaneous consecutive ones problem is introduced. Different from the standard problem feasible solution matrices must fulfill the consecutive ones property as well for columns as for rows. Definitions, complexity results, polyhedral investigations, and a strong IP formulation are given analogously to the standard problem. Proving that the weighted simultaneous consecutive ones problem (WSC1P) remains $\mathcal{NP}$-hard, even if the column-permutation and the row-permutation are fixed, will be the highlight concerning the complexity results.

Chapter 5 describes the development of a branch-and-cut algorithm for solving the WC1P as well as the WSC1P. A feasibility test, primal heuristics and several separation procedures are presented. These implementations are based on the branch-and-cut framework ABACUS and enable us to solve weighted consecutive ones problems efficiently.

We discuss possible applications of the WC1P and the WSC1P in chapter 6. They range from the physical mapping problem, a fundamental problem in computational biology, to

the problem of finding clusters of inorganic crystal structure types. For these two problems computational results are presented in chapter 7. Furthermore we investigate the behavior of our branch-and-cut code for random problems with few rows or columns and go into the question, how many entries of a random matrix have to be switched in average to obtain a matrix fulfilling the consecutive ones property.

Chapter 8 with a discussion on the theoretical and computational results as well as ideas for future research conclude this thesis.

Acknowledgement

Writing this thesis would not have been possible without the support and encouragement of several persons.

First of all, I am most grateful to my advisor Prof. Dr. Gerhard Reinelt for introducing me into the beautiful field of polyhedral combinatorics and for his unrestricted support of my work.

The research group of Prof. Dr. Reinelt provided a great environment not only for working but also for several spare-time activities which were very useful from time to time to refresh consumed energy. Thanks to Chotiros Surapholchai for the little presents from Thailand - and from the rest of the world. Thanks to Dirk Oliver Theis for providing me with tea especially in the final phase of this work. In this phase also mails from Catherine Proux-Wieland were very welcome which often stated that there are sweets in her office. Thanks to Klaus Wenger for dissuading me from becoming a teacher - he was right. I also enjoyed the talks about school in theory and in practice with Karin Tenschert. Many thanks to Dino Ahr with whom I shared the room for the last 4 years. He was always very helpful in answering questions especially with regard to LaTeX and we had many fruitful discussions about the world and his brother.

Moreover, I am thankful to Matthias Elf for precisely answering questions concerning ABACUS and Dr. Sebastian Leipert for providing his implementation of the PQ-tree algorithm.

In the very final phase of this work Dino Ahr, Dirk Oliver Theis, and Klaus Wenger helped with proofreading and Günther Alius, Sabine Dolderer, Martin Rösch, and Karin Tenschert helped with suggestions for improving the language. Thanks to all readers.

Finally I am very thankful to my house group for the moral support I received during the crucial phases of this work.

Chapter 1

Preliminaries

In this chapter we will give a very brief summary on basic definitions and notations of mathematics and computer science. They are restricted to those concepts that will be used in this thesis. If the reader is not familiar with these concepts or is looking for a more detailed survey of a certain topic we recommend the usage of textbooks. According references are given in each section.

1.1 Linear algebra

The main objects we will be dealing with are **matrices**. A matrix with m rows and n columns we will denote (m, n) matrix. If the matrix is not further specified we assume that all entries are real numbers. If $m = 1$ we call the matrix **row vector** , for $n = 1$ it is called **column vector** or only **vector**. An (m, n) matrix where the entries are restricted to be 0 or 1 is denoted **binary** or **0/1 matrix**. To specify the size of a binary matrix A we also write $A \in \{0, 1\}^{(m,n)}$ or to specify the size of a real vector b we write $b \in \mathbb{R}^m$.

A^T denotes the **transpose** of the matrix A. The entry of a matrix A in the j-th column of the i-th row is denoted by a_{ij}, the whole i-th row by $A_{i.}$ and the j-th column by $A_{.j}$. Similarly, for I being an ordered subset of the row set $\{1, 2, \ldots, m\}$ and J being an ordered subset of the column set $\{1, 2, \ldots, n\}$ of A we denote by A_{IJ} the submatrix of A belonging to the rows $i \in I$ and columns $j \in J$ in the specified order.

Let $x_1, x_2, \ldots, x_k$ be vectors $\in \mathbb{R}^d$ and $a_1, a_2, \ldots, a_k$ be coefficients $\in \mathbb{R}$. A **linear combination** of the vectors $\sum_{i=1}^{k} a_i x_i$ is called **affine combination** if $\sum_{i=1}^{k} a_i = 1$, **convex combination** if $\sum_{i=1}^{k} a_i = 1$ and $a_i \geq 0$, $1 \leq i \leq k$, and **conic combination** if $a_i \geq 0$, $1 \leq i \leq k$. For a set $S \in \mathbb{R}^d$ the **convex hull conv**(S) is defined as the set of convex combinations of finitely many vectors in S. The **linear hull lin**(S), the **affine hull aff**(S), and the **conic hull cone**(S) are defined analogously. The **affine rank** of S **arank**(S) is defined to be the smallest cardinality of a subset X of S with the property that $S \subset \text{aff}(X)$. By decreasing the affine rank by 1 we obtain the **dimension** of a set S, thus **dim** $S :=$ arank $S - 1$. According to this definition, the empty set has dimension -1.

Note that all these definitions on vectors can easily be extended to (m, n) matrices by

writing them as vectors $\in \mathbb{R}^{mn}$.

For more information about linear algebra see one of the common textbooks such as Fischer [Fis02].

1.2 Graph theory

An **undirected graph** (or **graph**) $G = (V, E)$ consists of a finite **node** set V and a finite **edge** set E. Each edge $e \in E$ is related to an unordered pair of nodes (u, v), $u, v \in V$, the so called **endnodes** of e. In a **directed graph** (or **digraph**) the endnodes are ordered, thus (u, v) and (v, u) denote two different edges. The first one is directed from u to v, the second one from v to u. An edge is called to be **incident** to its endnodes. The node-edge incidence matrix is a $(|V|, |E|)$ matrix with an entry being 1 if the corresponding edge is incident to the corresponding node and 0 otherwise. Two edges are called **adjacent**, if they have a common endnode. We consider only **simple** graphs, i.e., graphs without loops (edges of the form $e = (u, u)$) and without parallel edges (two or more edges related to the same pair of endnodes).

In a **weighted graph** $G = (V, E, c)$ a weight $c_{uv} \in \mathbb{R}$ is associated with each edge $e = (u, v)$. The weight of a subset of the edge set $F \subset E$ is defined as $c(F) = \sum_{(u,v)\in F} c_{uv}$.

A **(v_0, v_k) path** in a directed or undirected graph $G = (V, E)$ is defined as an edge set of the type $P = \{(v_0, v_1), (v_1, v_2), \ldots, (v_{k-1}, v_k)\}$, where k is the **length** of the path. A path with length $|V| - 1$ and $V = \{v_0, v_1, \ldots, v_k\}$ is called **Hamiltonian** path. Paths with $v_0 = v_k$ are called **cycles**. In an undirected graph a subset $F \subset E$ with $|F| = |V| - 1$ that contains no cycle is denoted as **spanning tree**.

Diestel [Die00] provides a good survey on graph theory.

1.3 Complexity theory

We mainly use complexity theory as a measure of the difficulty of a problem and the efficiency of an algorithm.

There are two different kinds of problems we are dealing with. The first type are **decision problems** , which are formulated as a question which can be answered by "yes" or "no" . The second type are **optimization problems**, the aims of which are to find best solutions in a sense to be specified. For example, the question whether a given graph $G = (V, E)$ contains a Hamiltonian path is a decision problem. The question which spanning tree in a weighted graph $G = (V, E, c)$ has the minimal total weight is an optimization problem. In both cases we are looking for an **algorithm** that answers the question correctly. The **running time** of an algorithm is defined as the number of elementary steps (addition, multiplication, etc.) which have to be executed to solve the problem. Normally the running time of an algorithm depends on the **input size**, which in the examples above is the size of the graph. This size is defined as the number of bits the input requires to be represented in the memory of a computer.

Taking into account that the running time of an algorithm A depends on its input size we introduce the **time complexity function** of an algorithm. This is a function $t_A : \mathbb{N} \to \mathbb{N}$ giving for each $n \in \mathbb{N}$ the maximum running time of A required to solve a problem with input size less or equal to n. We say the running time of A is $O(f(n))$, if there exists a constant $c > 0$ and an integer $n_0 > 0$ with $t_A(n) \leq cf(n)$ for all $n \geq n_0$. If the function f is a polynomial in n we say that A is a **polynomial time** algorithm.

The class of decision problems which can be solved by a polynomial time algorithm is denoted by $\mathcal{P}$. The class $\mathcal{NP}$ is defined as the set of all decision problems whose answer can be verified in polynomial time with the help of some structure whenever this answer is "yes". For example, the Hamiltonian path problem introduced above is in $\mathcal{NP}$. Whenever the algorithm says "yes, there is a Hamiltonian path", the path itself is such a structure, since it can be tested in polynomial time, if a given set of edges is indeed a Hamiltonian path.

For any decision problem in $\mathcal{P}$ the algorithm itself can also be used as verification. Therefore $\mathcal{P} \subseteq \mathcal{NP}$ and it is conjectured that $\mathcal{P} \neq \mathcal{NP}$. Whether this is true is probably the most important open question in complexity theory.

We say that a decision or an optimization problem is $\mathcal{NP}$**-hard** if any polynomial time algorithm for solving it provides polynomial time algorithms for all problems in $\mathcal{NP}$ and with it would imply $\mathcal{P} = \mathcal{NP}$.

A decision problem is called $\mathcal{NP}$**-complete** if it is $\mathcal{NP}$-hard and additionally lies in $\mathcal{NP}$. Normally, the $\mathcal{NP}$-completeness of a decision problem implies the $\mathcal{NP}$-hardness of the related optimization problem. This is also the case for the consecutive ones problem considered in this thesis (see section 2.4).

For a more detailed description of these concepts and a comprehensive survey on complexity theory we refer to Garey and Johnson [GJ79].

1.4 Combinatorial optimization problems

We now want to specify the type of optimization problems we are dealing with. Grötschel et al. [GLS93] or Korte and Vygen [KV00] are excellent books on the theory and practice of combinatorial optimization problems.

Definition 1.1 *Let E be a finite set and $\mathcal{F} \subset 2^E$ a subset of the powerset of E. Let further an* **objective function** *$c : \mathcal{F} \to \mathbb{R}$ be defined. A problem of the type*

$$\begin{array}{ll} \textit{maximize} & c(F) \\ \textit{subject to} & F \in \mathcal{F} \end{array}$$

is called **combinatorial optimization problem** *$(E, \mathcal{F}, c)$. The elements $F \in \mathcal{F}$ are called* **feasible solutions**. *If the objective function can be written as*

$$c(F) = \sum_{e \in F} c'(e)$$

with a function $c' : E \to \mathbb{R}$, we call such a problem **linear combinatorial optimization problem**. *Of course the problem can also be stated as minimization problem by multiplicating the objective function with -1.*

Example 1.2 *Let E be the set of edges in a weighted graph $G(V, E, c)$. Let further $\mathcal{F}$ be the set of spanning trees of the graph. Then the problem*

$$\begin{aligned} & \min c(F) \\ s.t. \quad & F \in \mathcal{F} \end{aligned}$$

with $c(F) = \sum_{e \in F} c_e$ is a linear combinatorial optimization problem. It is well known as the **mimimum spanning tree** *problem.*

In principle a combinatorial optimization problem can be solved by enumeration of all feasible solutions. However, in terms of the running time this is not a practicable way in general. In the previous example there are $|V|^{|V|-2}$ spanning trees. But despite this exponential number of feasible solutions there are polynomial time algorithms to solve the problem (see for example [CLR90]). Many other linear combinatorial optimization problems, for example the traveling salesman problem, the linear ordering problem as well as the weighted consecutive ones problem are known to be $\mathcal{NP}$-hard.

The following sections will show a way of how to tackle such problems efficiently.

1.5 Polyhedral theory

Applicating concepts of polyhedral theory to combinatorial optimization problems results eventually in the branch-and-cut method. Therefore we will give some basic definitions and results. For a deeper analysis see Pulleyblank [Pul89], Padberg [Pad95], or Ziegler [Zie95].

Given a vector $a \in \mathbb{R}^n$ and $a_0 \in \mathbb{R}$. Then the set $\{x \in \mathbb{R}^n \mid a^T x = a_0\}$ is called **hyperplane** and the set $\{x \in \mathbb{R}^n \mid a^T x \leq a_0\}$ is called **halfspace**.

A **polyhedron** $P \subseteq \mathbb{R}^n$ is defined as the intersection of finitely many halfspaces, or more formally

$$P = \{x \in \mathbb{R}^n \mid Ax \leq b\},$$

where $A \in \mathbb{R}^{(m,n)}$ and $b \in \mathbb{R}^m$. This description of a polyhedron is called **outer description**. A classical result in polyhedral theory (see Weyl [Wey35] and Minkowski [Min96]) is that a each polyhedron $P \subseteq \mathbb{R}^n$ can be written as

$$P = \text{conv}(X) + \text{cone}(Y),$$

where $X \subset \mathbb{R}^n$ and $Y \subset \mathbb{R}^n$ are finite sets. This description is called **inner description** of the polyhedron.

In this thesis we only consider bounded polyhedra which are called **polytopes**. A polytope $P \subset \mathbb{R}^n$ can be written as $P = \text{conv}(X)$ with a suitable finite set $X \subset \mathbb{R}^n$.

A polyhedron $P \in \mathbb{R}^n$ is said to be **full dimensional** if $\dim P = n$. An inequality $a^T x \leq a_0$ is said to be **valid** for a polyhedron $P \subseteq \mathbb{R}^n$, if P completely lies inside the halfspace $a^T x \leq a_0$.

If $a^T x \leq a_0$ is a valid inequality for a polyhedron P, then the set $F = P \cap \{x \mid a^T x = a_0\}$ determines a **face** of P. A face F with $\dim F = 0$ is called **vertex** , if $\dim F = \dim P - 1$ holds, the face is called **facet**. Let V be the set of vertices of a polytope P. Then $P = \text{conv}(V)$ holds. If $F = P \cap \{x \mid a^T x = a_0\}$ defines a facet, we call the valid inequality $a^T x \leq a_0$ **facet-defining** for P. Let finally $F = P \cap \{x \mid a^T x = a_0\}$ be a face of a polytope P. Then each vertex $x \in F$ is called **root** of the face F or root of the inequality $a^T x \leq a_0$.

1.6 Branch-and-cut

A **branch-and-cut algorithm** combines the concepts of two algorithmic approaches, namely the branch-and-bound method and the cutting plane method. This approach was first used by Grötschel, Jünger and Reinelt [GJR84] for solving the linear ordering problem. The name "branch-and-cut" was introduced by Padberg and Rinaldi [PR87, PR91].

We now want to give a brief summary on the branch-and-bound algorithm and the cutting plane method. For a more detailed presentation see Jünger et al. [JRT95].

1.6.1 Branch-and-bound

The general idea of a **branch-and-bound algorithm** is to decompose a problem into smaller problems, so-called **subproblems**. Say we are dealing with a maximization problem. Then all feasible solutions of subproblems are also feasible for the original problem and therefore are a **global lower bound** of the optimal solution. Furthermore we try to solve each subproblem to optimality or at least to compute upper bounds of the optimal solution for the current subproblem, the so-called **local upper bounds**. If a local upper bound is less or equal than the global lower bound, the subproblem has not to be considered any more and can be **fathomed**. If the subproblem neither can be solved nor be fathomed, it is split recursively into further subproblems. This can be represented by the so-called **branch-and-bound tree**. If no splitting is required at all, we say the problem is solved in the **root node** of the branch-and-bound tree.

The principle of this algorithm works as follows.

Algorithm 1.3 (Branch-and-bound)

(1) Initialize the list of active subproblems with the original problem.

(2) If the list of active subproblems is empty, STOP (the best feasible solution found so far is optimal).

(3) Choose some subproblem from the list of active problems and process it as follows:

(3.1) find an optimal solution for the subproblem, or

(3.2) prove that the optimal solution for the subproblem cannot exceed the global lower bound, or

(3.3) prove that the subproblem has no feasible solution, or

(3.4) if none of the above is possible, split the subproblem into further subproblems and add them to the list of active problems.

(4) Go to step 2.

The success of a branch-and-bound algorithm depends on the values for the lower and upper bounds. Lower bounds can be obtained by any feasible solution of a subproblem. Usually (**primal**) **heuristics** are used for this purpose. For the computation of upper bounds **relaxations** are used. A relaxation of a combinatorial optimization problem is a problem for which the set of feasible solutions contains the set of feasible solutions of the original problem. In a branch-and-cut algorithm linear programming is used as relaxation method.

1.6.2 Cutting plane method

Let $E = \{e_1, e_2, \ldots, e_k\}$ be the finite basic set of a combinatorial optimization problem and $\mathcal{F}$ the set of feasible solutions. Then any feasible solution $F \in \mathcal{F}$ of the problem can be written as a k-dimensional 0/1 **incidence vector** χ^F the i-th entry χ_i^F of which is equal to 1 if and only if $e_i \in F$, $1 \leq i \leq k$.

The polytope associated with this problem is defined as

$$P = \operatorname{conv}\left(\left\{\chi^F \mid F \in \mathcal{F}\right\}\right).$$

Each vertex of P corresponds to a feasible solution F. Now assume a linear combinatorial optimization problem $(E, \mathcal{F}, c)$

$$\begin{aligned} & \max c(F) \\ \text{s.t.} \quad & F \in \mathcal{F} \end{aligned}$$

is given with the associated polytope P. Then this problem is equivalent to solving

$$\begin{aligned} & \max c^T x \\ \text{s.t.} \quad & x \in P \\ & x \in \{0,1\}^k \end{aligned}$$

where $c = (c(e_i))$, $1 \leq i \leq k$, is the objective function vector. If we knew the outer description $Ax \leq b$ of P, it would be sufficient to solve the **linear program** (**LP**)

$$\begin{aligned} & \max c^T x \\ \text{s.t.} \quad & Ax \leq b \end{aligned}$$

since the **simplex algorithm** which usually is used for solving an LP terminates with an optimal vertex of P corresponding to a feasible solution. The vector c is called the

objective function and the inequalities in $Ax \leq b$ are called the **constraints** of the LP. For more information on linear programming see Chvátal [Chv83] and Schrijver [Sch86].

Unfortunately, the complete outer description of P is not available in general, since the number of inequalities in $Ax \leq b$ is too large. But we could proceed as follows. We start with a polytope Q which contains P and the outer description of which is small enough. For example the 0/1 cube $Q = \{x \in \mathbb{R}^k \mid 0 \leq x_i \leq 1, 1 \leq i \leq k\}$ is such a choice for initializing the constraints of the first LP. After solving this LP with respect to the objective function c, there are two possibilities for the optimal vertex x^*. If $x^* \in P$, then x^* is the incidence vector of an optimal solution of the combinatorial optimization problem $(E, \mathcal{F}, c)$. Otherwise, i.e., if $x^* \in Q$ but $x^* \notin P$, we are looking for an inequality $a^T x \leq a_0$ with the property that $P \in \{x \in \mathbb{R}^k \mid a^T x \leq a_0\}$ but $a^T x^* > a_0$. Since this inequality cuts off the LP solution x^* from the polytope Q, it is called **cutting plane**. After adding this cutting plane to the current LP and reoptimizing, we obtain a new LP solution and repeat this procedure. The whole algorithm looks as follows.

Algorithm 1.4 (Cutting plane method)

(1) Initialize $Q = \{x \in \mathbb{R}^k \mid 0 \leq x_i \leq 1, 1 \leq i \leq k\}$.

(2) Solve the linear programming problem $\max\{c^T x \mid x \in Q\}$ *and obtain an optimum solution* x^*.

(3) If x^* *is the incidence vector of some feasible solution* $F \in \mathcal{F}$*, then STOP (*I^* *is an optimum solution of* $(E, \mathcal{F}, c)$*).*

(4) Otherwise, identify an inequality $a^T x \leq a_0$ *with* $P \in \{x \in \mathbb{R}^k \mid a^T x \leq a_0\}$*, but* $a^T x^* > a_0$.

(5) Set $Q = Q \cap \{x \in \mathbb{R}^k \mid a^T x \leq a_0\}$ *and proceed with step 2.*

The most important part of this algorithm is step 4 of finding a cutting plane. This problem is called **separation** problem. Theoretical knowledge on the outer description of P usually is very helpful for efficient separation procedures.

To test in step 3 whether the LP solution x^* is the incidence vector of a feasible solution usually is done by a so-called **integer programming formulation** (**IP formulation**) of the problem $(E, \mathcal{F}, c)$. This is an inequality system $\overline{A}x \leq \overline{b}$ with the property that $P = \text{conv}(\{x \mid \overline{A}x \leq \overline{b}, x \text{ integer}\})$. Whether x^* is integer can be tested easily and the separation problem concerning the system $\overline{A}x \leq \overline{b}$ can in many cases be solved in polynomial time.

Note that the objective function values of all LP solutions occurring during the cutting plane algorithm are upper bounds of the optimal solution of the problem $(E, \mathcal{F}, c)$. This upper bounds are very useful for embedding the cutting plane method in a branch-and-bound algorithm. Furthermore, if the current LP solution x^* fulfills the constraints $\overline{A}x \leq \overline{b}$ but at least one entry x_i^* is not integer (here binary), the problem can easily be splitted into two subproblems by setting $x_i = 0$ in the first and $x_i = 1$ in the second subproblem.

Chapter 2

The Consecutive Ones Problem

This chapter will give the basic definitions concerned with the consecutive ones problem as it was introduced by Fulkerson and Gross [FG65]. The theoretical results on the complexity that are given in literature are summarized. Booth [Boo75] and Papadimitriou [Pap76] independently showed the $\mathcal{NP}$-completeness of an augmentation version of the consecutive ones problem. Furthermore Booth and Lueker [BL76] presented a very efficient algorithm for testing whether a given 0/1 matrix has the consecutive ones property, the so-called PQ-tree algorithm. The chapter will finish with a first new result, namely that the weighted consecutive ones problem is solvable in linear time if the column permutation is fixed. This result will turn out to be very useful for the construction of primal heuristics which will be done in chapter 5.

First of all we will introduce some appropriate notations.

2.1 Notations

Since in the consecutive ones problem we are actually dealing with matrices, we will introduce a new notation for a product of matrices. For two (m, n) matrices $A = (a_{ij})$ and $B = (b_{ij})$ we write $A \circ B$ for the scalar product of the two matrices written as vectors of dimension $m \cdot n$, i.e.

$$A \circ B = \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} b_{ij}.$$

Consequently inequalities of the corresponding polytope, which will be introduced in chapter 3, are denoted by "$A \circ x \leq a_0$", with x being the matrix of the variables representing the solution we are looking for. The matrix entry of row i and column j is denoted by x_{ij}. We will interpret $x = (x_{11}, \ldots, x_{1n}, \ldots, x_{m1}, \ldots, x_{mn})$ as a vector or as a matrix, whatever is more appropriate in a given context.

When writing "$A \circ x \leq a_0$" we assume that both matrices are of the same size. But we often deal with inequalities that have nonzero coefficients only for a submatrix of x. For this purpose we extend our notation. Let A be an (k, l)-matrix of coefficients and x the (m, n)-matrix of the variables $(m \geq k, n \geq l)$. For an ordered k-tuple $I = (r_1, \ldots, r_k)$ with

pairwise distinct rows $r_i \in \{1, \ldots, m\}$ and an ordered l-tuple $J = (c_1, \ldots, c_l)$ with pairwise distinct columns $c_j \in \{1, \ldots, n\}$ we define

$$A \circ x_{IJ} = \sum_{i=1}^{l} \sum_{j=1}^{k} a_{ij} x_{r_i c_j}.$$

For the sake of simplicity we will just say, for example, "$A \circ x_{IJ} \leq a_0$ for all (k,l)-tuples (I,J)", meaning that all k-tuples $I = (r_1, \ldots, r_k)$ and all l-tuples $J = (c_1, \ldots, c_l)$ are allowed for mapping A to x. Whenever we use $A \circ x$ we assume that $I = (1, \ldots, m)$ and $J = (1, \ldots, n)$.

2.2 Definition of the problem

2.2.1 The consecutive ones property

The consecutive ones property can be formulated both as a property of a subset of the power set of $\{1, \ldots, n\}$ and as a property of a binary (m,n) matrix. Both formulations can easily be converted into each other; as a matter of fact they only differ in the notation of the input. For matrices, we will define both the consecutive ones property for rows and for columns. Since these definitions equal each other except for switching the roles of rows and columns, we will only make use of the row-formulation in this thesis.

Definition 2.1 *An (m,n) 0/1-matrix M has the* **consecutive ones property for rows** *(for shorter writing we say M* **is C1PR***), if there is a permutation of the columns of M such that in each row of M the ones occur consecutively.*

Let $\pi \in S(n)$ be such a permutation of the columns. Then we say π **establishes** *the consecutive ones property for rows of M (π establishes* C1PR *of M).*

Example 2.2 *All binary matrices with less than 3 columns are* C1PR *since in any row with less than 3 elements the ones already occur consecutively.*

Any 0/1-matrix M with 2 rows consists of at most 4 different types of columns

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix} \text{ and } \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

Now consider any column permutation π that sorts the columns along the order written above. Obviously all such permutations and their reverses establish C1PR *of M. Therefore M is* C1PR*. An even easier argumentation shows that all 0/1-matrices consisting of a single row are* C1PR*.*

Thus the smallest nontrivial case is $m = n = 3$. Indeed the matrix

$$M = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

is not C1PR *which can be verified for example by testing all* 6 *possible column permutations. This is the only* $(3,3)$ *binary matrix that is not* C1PR *except for column and row permutations.*

Symmetrically, one can define the consecutive ones property for columns.

Definition 2.3 *An* (m,n) $0/1$*-matrix* M *has the* **consecutive ones property for columns** *(we say* M **is C1PC***), if there is a permutation of the rows of* M *such that in each column of* M *the ones occur consecutively.*

We say π *establishes the consecutive ones property for columns of* M *(*π *establishes* C1PC *of* M*) if* π *is a permutation in the above sense.*

As a result of this symmetric definition a $0/1$-matrix M is C1PC if and only if its transposed matrix M^T is C1PR.

The set consisting of all binary (m,n) matrices that are C1PR is different from the set of those that are C1PC. For example the following $(4,3)$-matrix is C1PR,

$$M = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

but none of the 24 row permutations establishes C1PC of M. Chapter 4 will deal with those binary matrices that are both C1PR and C1PC.

Sometimes the following **set-formulation** of the consecutive ones property is more suitable, especially for formulating the PQ-tree algorithm which we will introduce later.

Definition 2.4 *Given a finite set* U *and a subset* $\mathcal{M}$ *of the power set of* U*, i.e.* $\mathcal{M} = \{S_1, S_2, \ldots, S_k\}$*, where* $S_i \subseteq U, 1 \leq i \leq k$*, are subsets of* U*.* $\mathcal{M}$ *has the* **consecutive ones property (is C1P)***, if we can find an ordering* π *of the elements of* U *such that the elements of every* S_i*,* $1 \leq i \leq k$*, appear in* π *as a consecutive sequence. Here as well we say that an ordering* π *with this property establishes* C1P *of* $\mathcal{M}$*.*

Both the matrix-formulation and the set-formulation of the consecutive ones property might suggest that the test whether a given binary matrix is C1PR requires to check all possible column permutations. However, already Fulkerson and Gross who first introduced C1PR as a property of binary matrices had a polynomial time algorithm for this test, given in [FG65]. This algorithm is based on the following neat characterization of C1PR-matrices. The original formulation is slightly modified, since in contrast to our convention Fulkerson and Gross used C1PC instead of C1PR.

Theorem 2.5 ([FG65]) *Let* A *and* B $0/1$*-matrices satisfying*

$$AA^T = BB^T.$$

Then either both A and B are C1PR *or neither is. Moreover, if A and B have the same number of columns and A is* C1PR*, then the matrix B can be obtained by permuting the columns of A.*

The algorithm of Fulkerson and Gross first decomposes the matrix to be tested and after that uses the above theorem to check each part. The running time is $O(mn^2)$, where m is the number of rows and n the number of columns of the input matrix.

2.3 PQ-trees

An even better algorithm, as it works in linear time in the number of entries of the matrix, was introduced by Booth and Lueker in [BL76]. Their algorithm is based on so called PQ-trees. With the help of this data structure, it is even possible to represent all permutations establishing C1PR of a given 0/1-matrix by a single tree .

We will now give a short introduction into the PQ-tree data structure and the principle of the PQ-tree algorithm.

Definition 2.6 *Given a finite set U. A* **PQ-tree** *over U is a rooted ordered tree with the following properties:*

(1) The PQ-tree has two types of internal nodes, P-nodes and Q-nodes. A P-node has at least two children, a Q-node has at least three children.

(2) The leaves of the tree are exactly the elements of U.

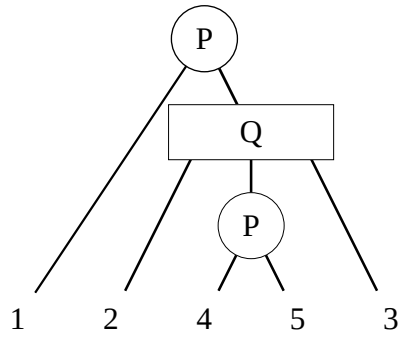


Figure 2.1: PQ-tree

Each PQ-tree over U represents a set of permutations of the elements of U. To illustrate this we need to introduce an equivalence relation on the PQ-trees of a common set U and to define the frontier of a PQ-tree.

Definition 2.7 *Two PQ-trees T_1 and T_2 over the same set U are* **equivalent** *($T_1 \equiv T_2$) if and only if T_1 can be transformed into T_2 by applying zero or more of the following transformations.*

(1) Perform an arbitrary permutation of the children of a P-node.

(2) Perform the reverse permutation of the children of a Q-node.

Definition 2.8 *Let T be a PQ-tree over U. The leaves of the tree scanned from left to right yield a permutation of the elements of U. This permutation is called the* **frontier** *of T and is denoted by* frontier(T).

Now we can define the permutations represented by a single tree.

Definition 2.9 *Let T be a PQ-tree over U. The set of all* **permutations** *of U that are* **represented** *by T is given by*

$$\text{perm}(T) = \{\text{frontier}(T') \mid T' \equiv T\}.$$

Example 2.10 *Figure 2.1 shows an illustration of a PQ-tree. P-nodes are displayed by a circle and Q-nodes by a rectangle. The frontier of this tree is* 12453 *and the set of all represented orderings is given by* $\{12453, 12543, 13452, 13542, 24531, 25431, 34521, 35421\}$.

The PQ-tree representing all possible permutations of the elements of U is called **universal tree** of U. It consists of one P-node as root which has exactly $|U|$ many leaves as children. Figure 2.2 shows the universal tree of $\{1, 2, 3, 4, 5\}$.

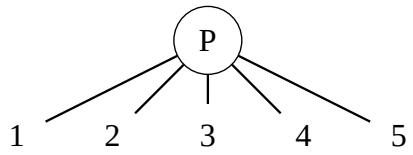


Figure 2.2: Universal tree

Another special tree is the so called **null tree** (see figure 2.3). This tree has no nodes and represents the empty set of permutations. In the strict sense it is no PQ-tree at all but its definition has turned out to be useful.

Figure 2.3: Null tree

The following theorem states the main relationship between PQ-trees and the consecutive ones property.

Theorem 2.11 ([BL76]) *Let U be a finite set. For each subset $\mathcal{M} = \{S_1, S_2, \ldots, S_k\}$, $S_i \subseteq U, 1 \leq i \leq k$, of the power set of U there exists a PQ-tree over U such that its set of represented permutations is identical to the set of all permutations π of U that establish* C1P *of $\mathcal{M}$, and vice versa.*

The **PQ-tree algorithm** related to a given set $\mathcal{M}$ works as follows. For $0 \leq i \leq k$ let $\mathcal{M}_i = \{S_1, S_2, \ldots, S_i\}$. The algorithm starts with constructing the universal tree, which in consistence with theorem 2.11 is the PQ-tree corresponding to the empty set $\mathcal{M}_0$. Now the universal tree is **reduced with respect to** S_1 to construct the PQ-tree of $\mathcal{M}_1$. This reduction step is performed k times after each other with respect to all sets S_i. After the i-th step the current PQ-tree is the one related to $\mathcal{M}_i$. Whenever the null tree arises (say after reduction step i), the algorithm can be terminated, since according to theorem 2.11 $\mathcal{M}_i$ is not C1P with the consequence that neither of $\mathcal{M}_j$, $i < j \leq k$, is. Otherwise the algorithm terminates after the k-th reduction step with the PQ-tree of $\mathcal{M}_k = \mathcal{M}$.

See the original paper of Booth and Lueker for more details on how the reduction procedure works. The outstanding fact is that reduction with respect to S_i can be performed in time $O(|S_i|)$ and therefore we obtain an overall running time which is linear in the input size.

Theorem 2.12 ([BL76]) *The algorithm for constructing the PQ-tree for a given set $\mathcal{M}$ has the time-complexity $O(|U| + k + \sum_{i=1}^{k} |S_i|)$.*

Now testing if a given binary (m, n)-matrix A is C1PR can be accomplished in linear time $O(mn)$. If the matrix has few "1"-entries and is stored in a suitable sparse-format the running time is even better. Another advantage is that the output of the PQ-tree algorithm is not only a "yes" or "no", but even provides the whole PQ-tree related to A, which allows to construct all permutations that establish C1PR of A. We will make use of this fact in section 5.2.2 by constructing a rounding heuristic for our problem. But note that creating a complete list of those permutations cannot be done in linear time, since their number might be exponential in n.

2.4 Results on the complexity

As we have just seen finding out whether a 0/1-matrix is C1PR, has a linear time complexity. An advanced problem one could tackle is "how far away" a given matrix is from being C1PR. We will see that such an augmented version of the problem will turn out to be much harder to solve.

2.4.1 Consecutive ones matrix augmentation

Of course the sense of "how far away" has to be specified. Booth [Boo75] defined the augmented version of the consecutive ones property as follows.

Definition 2.13 *A binary matrix M has the* **k-augmented consecutive ones property (is C1PR$_k$)** *if there exists a matrix M' which is* C1PR *and which arises from M by replacing at most of k "0"-entries of M by "1"-entries.*

Example 2.14

$$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

The left matrix is not C1PR *but* C1PR$_2$*, because, for example, switching the two "0"-entries in rows* 2 *and* 3 *to "1"-entries, leads to the matrix on the right-hand side which is* C1PR.

Note that the definition of C1PR$_k$ is somehow arbitrary since only changes from "0"-entries to "1"-entries are allowed. Analogously one could also have defined a problem where only the opposite changes are allowed, or even both kinds of replacements. Probably Booth did it this way, because there exists a nice transformation to an $\mathcal{NP}$-complete problem. At least this is the reason why the definition for the augmentation problem in the simultaneous case is done in a different way (see 4.2.1).

Booth [Boo75] and Papadimitriou [Pap76] independently transformed the decision version of C1PR$_k$ from the simple optimal linear arrangement problem. This problem is defined as deciding whether a graph has a numbering of the vertices such that the sum over all edges of the (positive) differences of their endpoints is less than a constant W. Or more formally,

Definition 2.15 *A graph $G = (V, E)$ has a* **simple optimal linear arrangement** *of weight W if there is an ordering π of the vertices such that*

$$\sum_{ij \in E} |\pi(i) - \pi(j)| \leq W.$$

Garey, Johnson and Stockmeyer [GJS76] showed that finding a simple optimal linear arrangement of a graph with weight W is $\mathcal{NP}$-complete. Thus to show the $\mathcal{NP}$-completeness of the decision version of C1PR$_k$ only a polynomial transformation between both problems remains to be shown. Due to the importance of this result for this thesis the proof according to [Boo75] will be given in the following.

Theorem 2.16 ([Boo75],[Pap76]) *Deciding whether a binary matrix M is* C1PR$_k$ *is $\mathcal{NP}$-complete.*

Proof. Verifying whether a given k-augmentation for M is C1PR can be done in polynomial time using the PQ-tree algorithm 2.12. It remains to show that given a polynomial time algorithm for testing C1PR$_k$ one can also solve simple optimal linear arrangement in polynomial time. The transformation works as follows. Given a graph $G = (V, E)$ and a

weight W. We construct our 0/1-matrix M simply as the node-edge incidence matrix of the graph. More precisely we define $M = (m_{ij})$ as

$$m_{ij} = \begin{cases} 1 & : \quad v_j \in e_i \\ 0 & : \quad v_j \notin e_i. \end{cases}$$

Now the claim is that G has a simple linear arrangement of weight W if and only if M is $\text{C1PR}_{(W-|E|)}$. For any ordering of the vertices the number of zeros between the two ones corresponding to an edge in each row is exactly one less than the difference of the numbers of the two incident vertices. Thus the total number of zeros to be switched is exactly $|E|$ less than the weight of the linear arrangement.

Assume we have a polynomial time algorithm which decides whether a given matrix M is C1PR_k. Since the construction above can be performed also in polynomial time there would be a polynomial time algorithm for the simple optimal linear arrangement. Thus the $\mathcal{NP}$-completeness is proven. □

2.4.2 The weighted consecutive ones problem

A step to generalize the problem further is to consider the associated optimization problem. That is, we do not only allow changes of entries both from 1 to 0 and vice versa, but we also penalize each switch with a certain cost. The aim is then to find a series of changes transferring the matrix to one that is C1PR and minimizing the total cost of the switches. This general version of the problem has to be solved in computational biology. Section 6.1 will deal with this physical mapping problem.

Definition 2.17 *Let some 0/1-matrix $B = (b_{ij}) \in \{0,1\}^{(m,n)}$ and a cost matrix $C' = (c'_{ij}) \in \mathbb{R}^{(m,n)}$ be given. For a binary matrix $A = (a_{ij}) \in \{0,1\}^{(m,n)}$ the total cost for switching entries to transform B to A is defined as*

$$c(A) = \sum_{i=1}^{m} \sum_{j=1}^{n} c'_{ij} |b_{ij} - a_{ij}|.$$

We define the **weighted consecutive ones problem (WC1P)** *as the task to find a matrix A which is* C1PR *and minimizes $c(A)$.*

Since B and C are fixed it is easy to linearize the objective function. Let the (m,n)-matrix $C = (c_{ij})$ be defined as

$$c_{ij} = \begin{cases} c'_{ij} & : \quad b_{ij} = 0 \\ -c'_{ij} & : \quad b_{ij} = 1. \end{cases}$$

Now we have $c(A) = C \circ A - C \circ B$ and since $C \circ B$ is a constant the WC1P can be written as

$$\begin{aligned} & \min C \circ A \\ s.t. \quad & A \text{ is C1PR.} \end{aligned}$$

In the following the converted matrix C will be denoted as cost matrix.

One easily can transform the augmentation problem to the weighted problem with the consequence that WC1P is $\mathcal{NP}$-hard.

Corollary 2.18 *The weighted consecutive ones problem is $\mathcal{NP}$-hard.*

Proof. Since in Booth's augmentation definition (2.13) no switches from ones to zeros are allowed, one has to penalize them with a high cost. The total number of entries mn is sufficient for this purpose. Now let A be a given binary matrix. Then we construct the cost matrix C as

$$c_{ij} = \begin{cases} 1 & : \quad a_{ij} = 0 \\ -mn & : \quad a_{ij} = 1. \end{cases}$$

Further, let l be the number of ones in A. Then A is C1PR$_k$ if and only if the solution of

$$\begin{aligned} &\min C \circ A \\ s.t. \quad &A \text{ is C1PR} \end{aligned}$$

is less or equal to $k - lmn$. The construction of the cost function is polynomial. Therefore the $\mathcal{NP}$-hardness is shown. □

Solving this $\mathcal{NP}$-hard WC1P as well as variants to optimality is the main part of this thesis. Therefore in the next chapter we will do some polyhedral investigations of the associated polytope which are the basis of the branch-and-cut approach, we will present in chapter 5. The chapters 6 and 7 will give applications and computational results on the WC1P.

First we want to investigate a special form of the WC1P, where we assume that the column permutation that should establish C1PR of the input matrix is already known.

2.4.3 Fixed column permutation

Without loss of generality we assume that the known column permutation is the identity. Otherwise one can rearrange the columns of the input matrix. According to Booth [Boo75] binary matrices where the identity establishes C1PR are in what we call the Petrie form. This denotation is due to Flinders Petrie's work on seriation in archaeology (see section 6.2).

Definition 2.19 *A 0/1-matrix which has the consecutive ones property for rows without any rearrangements of its columns is in* **Petrie form (is PET)**.

Example 2.20

$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix} \qquad \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}$$

Both matrices are C1PR *but only the left one is* PET.

Now we can define the **weighted Petrie problem (WPEP)** as a special variant of the WC1P where the column permutation is fixed to the identity.

$$\begin{aligned} & \min C \circ A \\ s.t. \quad & A \text{ is PET.} \end{aligned}$$

Since the column permutation is fixed the problem decomposes into m corresponding problems for single rows, minimizing $C_{i\cdot} \circ A_{i\cdot}$ such that $A_{i\cdot}$ is PET. But in this form the problems are equivalent to the **maximum subarray problem**. Here the aim is, for a given array of numbers, to find a consecutive subarray such that the sum of the numbers in the subarray is a maximum. This can be done in linear time in the size of the array by making use of the following scan-line algorithm.

Algorithm 2.21 (`MaximumSubarray(array, length)`)

(1) Initialize scanMax = 0, maxSum = 0

(2) For index from 0 to length-1

(2.1) if (scanMax + array[index] > 0) scanMax = scanMax + array[index]
else scanMax = 0

(2.2) if (scanMax > maxSum) maxSum = scanMax

(3) return(maxSum)

Performing this algorithm for each row and taking the sum of all return values also gives an optimal solution for WPEP in linear time.

Lemma 2.22 WPEP *is solvable in linear time.*

This directly leads to the observation that WC1P is in $\mathcal{P}$ if the number of columns is fixed.

Corollary 2.23 *For fixed number n of columns* WC1P *can be solved in linear time in the number m of rows.*

Proof. We make use of the fact that for a fixed number of columns also the number of permutations of the columns is a constant in m. For a fixed permutation π we solve a WPEP where the columns of the coefficient matrix of WC1P are permuted according to π. As seen before this can be done in linear time. Now we perform this for all $n!$ column permutations and take the minimum of all WPEP solutions. The time required for this is also linear in m, since $n!$ does not depend on m. In fact we only need half of the permutations since π and its reverse define the same WPEP. □

In section 3.6 we will see that the WC1P is also polynomially solvable in the case that the number of rows is fixed. This is a consequence of the more general result that even the number of facets of the related polytopes grows only polynomially if the number of columns or rows is fixed.

Chapter 3

The Facial Structure of the Consecutive Ones Polytope

As one of the most crucial parts of developing an effective branch-and-cut algorithm turned out to be the theoretical knowledge of the facial structure of the associated polytope.

This chapter defines the consecutive ones polytope $P^{m,n}_{\mathrm{C1R}}$ and gives a variety of classes of facet-defining inequalities as well as methods to construct them. The associated separation problems will be addressed in section 5.3. Proving that the number of facets of $P^{m,n}_{\mathrm{C1R}}$ grows only polynomially in m if n is fixed and polynomially in n if m is fixed, will conclude this chapter. An immediate consequence is that the WC1P is polynomially solvable for a fixed number of rows or columns.

3.1 Basic definitions and results

The usual way in combinatorial optimization is to define a polytope whose vertices correspond to the feasible solutions of the problem. If these feasible solutions are given as 0/1 vector we can take the convex hull of all these vectors. In our case we do not deal with binary vectors but with binary matrices. Therefore we will take the convex hull over all feasible C1PR matrices. We do not introduce polytopes as convex hulls over C1PC matrices since they are isomorphic to the previous ones after exchanging the roles of rows and columns.

3.1.1 The consecutive ones polytope $P^{m,n}_{\mathrm{C1R}}$

For given numbers of rows m and columns n the **consecutive ones polytope** is defined as

$$\boldsymbol{P^{m,n}_{\mathrm{C1R}}} = \mathrm{conv}\{M \mid M \text{ is an } (m,n)\text{-matrix with C1PR}\}.$$

It is easy to see that $P^{m,n}_{\mathrm{C1R}}$ is full dimensional.

Theorem 3.1 *$P^{m,n}_{\mathrm{C1R}}$ has dimension $m \cdot n$.*

Proof. The zero matrix is C1PR and, for every $1 \leq i \leq m$ and $1 \leq j \leq n$, the matrix consisting of zeros only except for a "1"-entry in position ij is C1PR. This gives a set of $m \cdot n + 1$ affinely independent C1PR matrices. □

A nice consequence of the fact that $P^{m,n}_{\mathrm{C1R}}$ has full dimension is that we do not need equations to describe these polytopes and therefore the descriptions of facet-defining inequalities are unique up to multiplication with a positive scalar.

Being able to lift facet-defining inequalities to polytopes of higher dimensions is a very useful property. In our case even trivial lifting is possible.

3.1.2 Lifting facet-defining inequalities

Let $A \circ x \leq a_0$ be a valid inequality for $P^{m,n}_{\mathrm{C1R}}$ and let $m' \geq m$ and $n' \geq n$. We say that the inequality $\overline{A} \circ x \leq a_0$ for $P^{m',n'}_{\mathrm{C1R}}$ is obtained from $A \circ x \leq a_0$ by **trivial lifting** if

$$\overline{a}_{ij} = \begin{cases} a_{ij} & : \quad i \leq m \text{ and } j \leq n \\ 0 & : \quad \text{otherwise.} \end{cases}$$

Theorem 3.2 *Let $A \circ x \leq a_0$ be a facet-defining inequality for $P^{m,n}_{\mathrm{C1R}}$ and let $m' \geq m$ and $n' \geq n$. If the inequality $\overline{A} \circ x \leq a_0$ for $P^{m',n'}_{\mathrm{C1R}}$ is obtained from $A \circ x \leq a_0$ by trivial lifting then it defines a facet of $P^{m',n'}_{\mathrm{C1R}}$.*

Proof. Denote the lifted inequality by $\overline{A} \circ x \leq a_0$, and let $\overline{x}$ be a matrix satisfying $A \circ \overline{x} = a_0$. First consider the case $m' > m$ and $n' = n$, where without loss of generality $m' = m + 1$.

We form the matrix $\overline{x}_0$ by adding a zero row to $\overline{x}$ and matrices $\overline{x}_j$ by adding a row of zeros except for a "1"-entry in column j, $j = 1, \ldots, n$. Obviously all generated matrices are C1PR, and we have $\overline{A} \circ \overline{x}_j = a_0$, for all $j = 0, 1, \ldots, n$.

Therefore, if we have $m \cdot n$ affinely independent matrices satisfying $A \circ x = a_0$ we can obtain by the above construction $(m + 1) \cdot n$ affinely independent matrices satisfying $\overline{A} \circ x = a_0$, thus proving that the trivially lifted inequality is also facet-defining.

The case $m' = m$ and $n' > n$, where without loss of generality $n' = n + 1$ follows along a similar line. We add one column to $\overline{x}$ and form the matrix $\overline{x}_0$ by adding a zero column and matrices $\overline{x}_i$ by adding a column of all zeros except for a 1 in row i, $i = 1, \ldots, m$. Again all generated matrices are C1PR since column $n + 1$ can be moved to an appropriate position, and we have $\overline{A} \circ \overline{x}_i = a_0$, for all $i = 0, 1, \ldots, m$. From $m \cdot n$ affinely independent matrices satisfying $A \circ x = a_0$ we can obtain this way $m \cdot (n + 1)$ affinely independent matrices satisfying $\overline{A} \circ x = a_0$.

The general result follows from these two constructions. □

3.1.3 Melting valid inequalities

If trivial lifting is possible, this means that larger polytopes inherit all facets of smaller polytopes. Conversely given a valid inequality $A \circ x = a_0$ it is also possible to derive valid

inequalities $\tilde{A} \circ \tilde{x} = a_0$ for polytopes of lower dimensions. We call this method **melting**. Unfortunately the melting process does not preserve the facet-defining property.

Theorem 3.3 *Let $A \circ x \leq a_0$ be a valid inequality for $P^{m,n}_{\mathrm{C1R}}$ and let $\tilde{A}$ be the resulting $(m-1,n)$-matrix ($(m,n-1)$-matrix) when replacing two rows (columns) of A by their sum, then $\tilde{A} \circ \tilde{x} \leq a_0$ is a valid inequality for $P^{m-1,n}_{\mathrm{C1R}}$ ($P^{m,n-1}_{\mathrm{C1R}}$).*

Proof. We will do the proof for the case of melting rows. Without loss of generalization $\tilde{A}$ is constructed from A by replacing the last two rows $A_{m-1.}$ and $A_{m.}$ by their sum $\tilde{A}_{m-1.}$. Now let $\tilde{M}$ be any $(m-1,n)$ 0/1 matrix that is C1PR. We construct an (m,n) 0/1 matrix M from $\tilde{M}$ by duplicating the last row, thus $M_{m.} = M_{m-1.} = \tilde{M}_{m-1.}$. Since the last row of M is a copy of a row of $\tilde{M}$ we can establish C1PR of M with the same column permutation as for $\tilde{M}$. Further $A \circ x \leq a_0$ is valid for $P^{m,n}_{\mathrm{C1R}}$ and with it $A \circ M \leq a_0$ holds. Because of the construction of M we have $\tilde{A} \circ \tilde{M} = A \circ M$ with the consequence that also $\tilde{A} \circ \tilde{M} \leq a_0$ holds and since $\tilde{M}$ was chosen arbitrarily from the set of all C1PR-matrices of given size $\tilde{A} \circ x \leq a_0$ is valid for $P^{m-1,n}_{\mathrm{C1R}}$. The proof in the case of melting columns follows along the same lines. □

Example 3.4 *The following inequality is facet-defining for $P^{4,4}_{\mathrm{C1R}}$ (see A.2)*

$$\begin{pmatrix} 1 & 1 & 0 & -1 \\ 1 & 0 & 1 & -1 \\ 0 & 1 & -1 & 1 \\ 0 & -1 & 1 & 1 \end{pmatrix} \circ x \leq 7.$$

Melting rows 1 *and* 2*, columns* 1 *and* 3 *resp., leads to the inequalities*

$$\begin{pmatrix} 2 & 1 & 1 & -2 \\ 0 & 1 & -1 & 1 \\ 0 & -1 & 1 & 1 \end{pmatrix} \circ x \leq 7 \qquad \begin{pmatrix} 2 & 0 & -1 \\ 1 & 1 & -1 \\ 1 & -1 & 1 \\ -1 & 1 & 1 \end{pmatrix} \circ x \leq 7.$$

The first inequality is valid for $P^{3,4}_{\mathrm{C1R}}$, the second one for $P^{4,3}_{\mathrm{C1R}}$. Performing both melting procedures after another yields

$$\begin{pmatrix} 3 & 1 & -2 \\ 1 & -1 & 1 \\ -1 & 1 & 1 \end{pmatrix} \circ x \leq 7$$

which is valid for $P^{3,3}_{\mathrm{C1R}}$ but not facet-defining since it is the sum of

$$\begin{pmatrix} 1 & 1 & -1 \\ 1 & -1 & 1 \\ -1 & 1 & 1 \end{pmatrix} \circ x \leq 5$$

and three trivial inequalities, which will be introduced in the following.

3.1.4 Facets induced by trivial inequalities

The validity of the **trivial inequalities** $x_{ij} \geq 0$ and $x_{ij} \leq 1$, for all $1 \leq i \leq m$, $1 \leq j \leq n$ is obvious due to the definition of $P^{m,n}_{\text{C1R}}$. It is easily seen that they also define facets.

Lemma 3.5 *For all $m \geq 1$, $n \geq 1$, $1 \leq i \leq m$, $1 \leq j \leq n$, the inequalities $x_{ij} \geq 0$ and $x_{ij} \leq 1$ define facets of $P^{m,n}_{\text{C1R}}$.*

Proof. We have $P^{1,1}_{\text{C1R}} = \text{conv}\{0, 1\}$ and therefore $x_{11} \geq 0$ and $x_{11} \leq 1$ are facet-defining for $P^{1,1}_{\text{C1R}}$. The general result holds due to the trivial lifting property. □

3.2 Tucker's characterization

In the previous section we presented basic properties of consecutive ones polytopes but we have still no idea how an integer programming formulation of the problem could look like. That is the outer description of a polytope whose integral interior points are exactly the feasible C1PR matrices and allows that the WC1P can be formulated as a mixed integer program. A characterization of feasible C1PR matrices could be very helpful for this purpose. By making use of a graph theoretical correspondence Tucker ([Tuc72]) has characterized these matrices by exhibiting five types of matrices M_{1_k}, M_{2_k}, M_{3_k}, M_4 and M_5 which must not occur as submatrices. To be more precise, Tucker identified the rows and columns of 0/1 matrices with the two node-sets of an undirected bipartite graph with an edge between two nodes if and only if the corresponding entry of the matrix is 1. From a set of forbidden subgraphs in the graph theoretical formulation he derived the following characterization for the matrix formulation of the consecutive ones property.

Theorem 3.6 ([Tuc72]) *The 0/1-matrix M has the consecutive ones property for rows if and only if no submatrix of M possibly after column or row permutation is one of the matrices occurring in figure 3.1.*

3.2.1 A preliminary IP formulation

Based on this result we can obtain an integer programming formulation of WC1P. All we have to do is to construct valid inequalities that cut off all forbidden matrices. For M_{1_k}, M_{2_k}, M_{3_k}, M_4 and M_5 let $\mathcal{M}_{1_k}$, $\mathcal{M}_{2_k}$, $\mathcal{M}_{3_k}$, $\mathcal{M}_4$ and $\mathcal{M}_5$ denote the corresponding matrices where all zero entries are replaced by -1. Figure 3.2 shows these matrices. Note that for convenience we write here and in the following "$-$" instead of "-1" and "$+$" instead of "$+1$".

Now look for example at the possible values of $\mathcal{M}_5 \circ x$ for a given $(4, 5)$ binary matrix x. The maximum value for this sum is 11 and will be reached if and only if x equals the Tucker matrix M_5. Therefore the inequality $\mathcal{M}_5 \circ x \leq 10$ cuts off only M_5. Note that 11 is the number of ones occurring in M_5. This principle can be generalized to all forbidden matrices. One only has to make sure that the right hand sides of the constructed inequalities are equal to the number of ones of the matrix to be cut off reduced by one.

$\overbrace{\begin{pmatrix} 1 & 1 & & & & \\ & 1 & 1 & & & \\ & & \ddots & \ddots & & \\ & & & & 1 & 1 \\ 1 & 0 & \dots & & 0 & 1 \end{pmatrix}}^{(k+2,k+2)\text{-matrix } M_{1_k}\,(k\geq 1)}$

$\overbrace{\begin{pmatrix} 1 & 1 & & & & & 0 \\ & 1 & 1 & & & & 0 \\ & & \ddots & \ddots & & & \vdots \\ & & & & 1 & 1 & 0 \\ 0 & 1 & \dots & & 1 & 1 & 1 \\ 1 & 1 & \dots & & 1 & 0 & 1 \end{pmatrix}}^{(k+3,k+3)\text{-matrix } M_{2_k}\,(k\geq 1)}$

$\overbrace{\begin{pmatrix} 1 & 1 & & & & 0 \\ & 1 & 1 & & & 0 \\ & & \ddots & \ddots & & \vdots \\ & & & 1 & 1 & 0 \\ 0 & 1 & \dots & 1 & 0 & 1 \end{pmatrix}}^{(k+2,k+3)\text{-matrix } M_{3_k}\,(k\geq 1)}$

$\overbrace{\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \end{pmatrix}}^{(4,6)\text{-matrix } M_4}$

$\overbrace{\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \end{pmatrix}}^{(4,5)\text{-matrix } M_5}$

Figure 3.1: Forbidden Tucker matrices M_{1_k}, M_{2_k}, M_{3_k}, M_4 and M_5

The complete IP formulation of the WC1P looks as follows:

$$\begin{array}{rll} & \min\ c^T x & \\ \mathcal{M}_{1_k} \circ x_{IJ} & \leq 2k+3 & \text{for all } (k+2,k+2)\text{-tuples } (I,J) \text{ and } k \geq 1, \\ \mathcal{M}_{2_k} \circ x_{IJ} & \leq 4k+5 & \text{for all } (k+3,k+3)\text{-tuples } (I,J) \text{ and } k \geq 1, \\ \mathcal{M}_{3_k} \circ x_{IJ} & \leq 3k+1 & \text{for all } (k+2,k+3)\text{-tuples } (I,J) \text{ and } k \geq 1, \\ \mathcal{M}_4 \circ x_{IJ} & \leq 8 & \text{for all } (4,6)\text{-tuples } (I,J), \\ \mathcal{M}_5 \circ x_{IJ} & \leq 10 & \text{for all } (4,5)\text{-tuples } (I,J), \\ x_{ij} & \in \{0,1\} & \text{for all } i=1,\dots,m,\ j=1,\dots,n. \end{array}$$

The validity of this formulation is clear, since for a 0/1-matrix all Tucker submatrices are cut off by this set of inequalities.

However, this is not a good formulation in the sense that most of the inequalities do not define facets of the associated polytope. The following example 3.7 shows a Tucker-based inequality that is not facet defining.

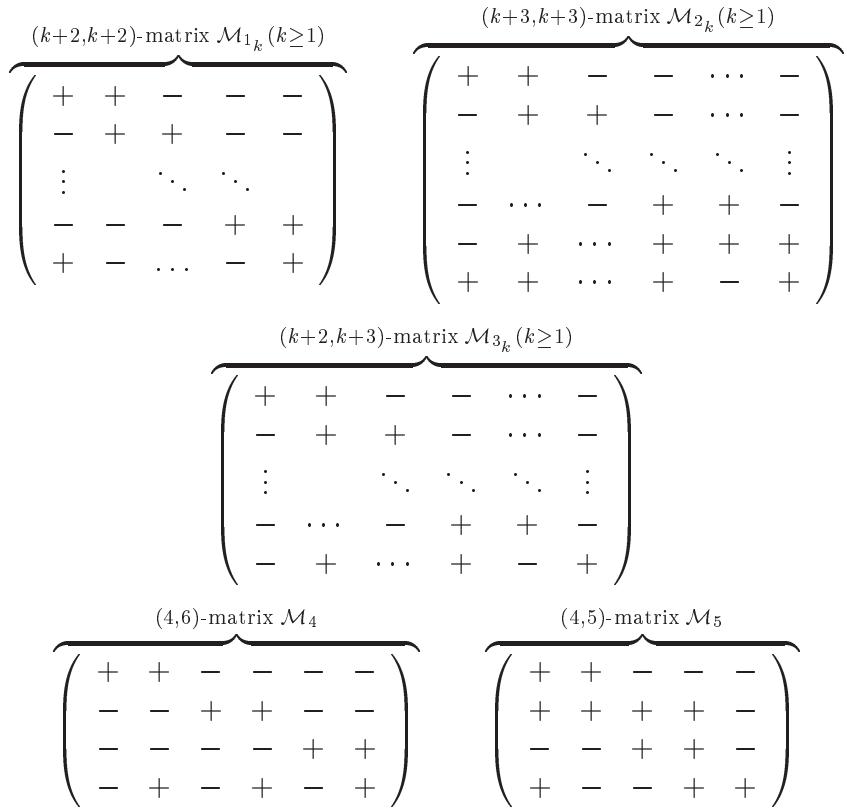


Figure 3.2: The left-hand sides of the IP formulation derived from Tucker's characterization. The right-hand sides for each inequality are equal to the number of "+"-entries reduced by 1.

Example 3.7 *Consider the inequality* $\mathcal{M}_{1_2} \circ x \leq 7$.

$$\begin{pmatrix} + & + & - & - \\ - & + & + & - \\ - & - & + & + \\ + & - & - & + \end{pmatrix} \circ x \leq 7$$

We add the inequality $\mathcal{M}_{1_1} \circ x \leq 5$ *and* 7 *trivial inequalities, all lifted to* $P^{4,4}_{\mathrm{C1R}}$.

$$\begin{pmatrix} + & + & - & 0 \\ - & + & + & 0 \\ + & - & + & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \circ x \leq 5 \qquad \begin{pmatrix} 0 & 0 & 0 & - \\ 0 & 0 & 0 & - \\ 0 & 0 & 0 & + \\ + & - & - & + \end{pmatrix} \circ x \leq 3$$

After taking the sum, dividing by 2 and round off the right hand side we get

$$\begin{pmatrix} + & + & - & - \\ - & + & + & - \\ 0 & - & + & + \\ + & - & - & + \end{pmatrix} \circ x \leq 7,$$

which is stronger than the first inequality. Therefore $\mathcal{M}_{1_2} \circ x \leq 7$ *cannot be facet-defining for* $P^{4,4}_{\text{C1R}}$.

With similar arguments one can show that $\mathcal{M}_{1_1} \circ x \leq 5$ and $\mathcal{M}_{3_1} \circ x \leq 5$ are the only inequalities of the Tucker based IP formulation that are facet-defining. Note that the strengthened inequality in example 3.7 only differs from $\mathcal{M}_{1_2} \circ x \leq 7$ by an additional "0"-entry instead of a "−"-entry. Therefore one idea for constructing facet defining inequalities cutting of the Tucker matrices could be to replace "−"-entries as long by zeros as the inequality remains valid. The next sections shows two possibilities to construct such classes of facets.

3.3 Facets induced by staircase inequalities

As said before we want to strengthen the first IP formulation given in 3.2.1 with the purpose to obtain an IP formulation consisting only of facets for the corresponding polytope.

First of all we are looking for facet defining inequalities cutting of the Tucker matrices M_{1_k}, M_{2_k} and M_{3_k}. The inequalities that will do this job are based on two classes of matrices $\mathcal{F}_{1_k}$ and $\mathcal{F}_{2_k}$, for $k \geq 1$. As mentioned before the first class of matrices $\mathcal{F}_{1_k}$ is obtained from $\mathcal{M}_{1_k}$ by replacing most of the "−"-entries by zeros. There is only one "−"-entry per row remaining. $\mathcal{F}_{2_k}$ arises from $\mathcal{M}_{3_k}$ in a similar way. There are two "−"-entries per row remaining and in addition most of the "+"-entries of the last row are replaced by zeros (the right-hand side will be decreased accordingly). The matrices are shown in figure 3.3.

$$\begin{pmatrix} + & & & & - & + \\ + & + & & & & - \\ & + & + & & & - \\ & & \ddots & \ddots & & \vdots \\ & & & + & + & - \\ - & & & & + & + \end{pmatrix} \qquad \begin{pmatrix} + & + & & & & - & - \\ & + & + & & & - & - \\ & & \ddots & \ddots & & \vdots & \vdots \\ & & & + & + & - & - \\ - & & & & + & + & - \\ - & & & & + & - & + \end{pmatrix}$$

Figure 3.3: Staircase matrices $\mathcal{F}_{1_k}$ and $\mathcal{F}_{2_k}$.

The structure of the matrices $\mathcal{F}_{1_k}$ and $\mathcal{F}_{2_k}$ as well as the Tucker matrices themselves look similar to a staircase. Therefore we will name these matrices **staircase matrices** and the derived inequalities **staircase inequalities**. Now we will show that these matrices lead to facet-defining inequalities. To be more precise, here we only show the validity of the corresponding inequalities, the facet-defining property for $P^{m,n}_{\text{C1R}}$ follows from the fact that they are even facet-defining for the simultaneous polytope $P^{m,n}_{\text{C1S}}$ which is contained in $P^{m,n}_{\text{C1R}}$. This will be shown in section 4.3.1.

Theorem 3.8 *The staircase inequalities*

$$\mathcal{F}_{1_k} \circ x_{IJ} \leq 2k+3,$$

for $k \geq 1$ and all $(k+2, k+2)$-index sets, are valid for $P^{m,n}_{\text{C1R}}$ for all $m \geq k+2$ and $n \geq k+2$.

Proof. We only need to consider the canonical ordered index sets $I = \{1, 2, \ldots, m\}$ and $J = \{1, 2, \ldots, n\}$. Let x be a 0/1-matrix such that $\mathcal{F}_{1_k} \circ x > 2k+3$. Then x must have 0 entries at the "$-$"-positions of $\mathcal{F}_{1_k}$ and 1 entries at the "+"-positions. We will show that we cannot obtain a C1PR matrix no matter how the remaining entries of x are assigned and how the columns of x are permuted. We call a row "bad" if and only if there is a forced 0 between two forced 1 entries. Note that x has exactly one bad row. Now consider any sequence of permutations of adjacent columns of x. The status of a row changes if and only if the entries of this row in the two columns are a forced 0 and a forced 1. For each pair of columns of x there is an even number (0 or 2) of rows with this property. Thus for any column permutation of x the number of bad rows remains odd and therefore at least one. But a matrix with a bad row cannot be C1PR and validity of the inequality follows. □

Theorem 3.9 *The staircase inequalities*

$$\mathcal{F}_{1_k} \circ x_{IJ} \leq 2k+3,$$

for $k \geq 1$ and all $(k+2, k+2)$-index sets, are facet-defining for $P^{m,n}_{\text{C1R}}$ for all $m \geq k+2$ and $n \geq k+2$.

Proof. The proof immediately follows from their validity, from theorem 4.33 and lemma 4.31. □

Analogously we will proceed with the second class of staircase inequalities.

Theorem 3.10 *The staircase inequalities*

$$\mathcal{F}_{2_k} \circ x_{IJ} \leq 2k+3,$$

for $k \geq 1$ and all $(k+2, k+3)$-index sets, are valid for $P^{m,n}_{\text{C1R}}$ for all $m \geq k+2$ and $n \geq k+3$.

Proof. The proof is similar to the previous one. Here rows of x are called "bad" if and only if there is a forced 0 between two forced 1 entries and a forced 1 between two forced 0 entries, i.e., the relative order of the nonzero entries of the corresponding row in $\mathcal{F}_{2_k}$ is either "$+-+-$" or "$-+-+$". Then the verification of validity follows exactly along the same lines as for $\mathcal{F}_{1_k}$. □

Theorem 3.11 *The staircase inequalities*

$$\mathcal{F}_{2_k} \circ x_{IJ} \leq 2k+3,$$

for $k \geq 1$ and all $(k+2, k+3)$-index sets, are facet-defining for $P^{m,n}_{\text{C1R}}$ for all $m \geq k+2$ and $n \geq k+3$.

Proof. The proof immediately follows from their validity, from theorem 4.34 and lemma 4.31. □

3.4 An IP formulation with facets

For the purpose of deriving an IP formulation with facets, it remains to construct facets cutting of the two single Tucker matrices M_4 and M_5. We will see that the inequalities $\mathcal{F}_3 \circ x \leq 8$ and $\mathcal{F}_4 \circ x \leq 8$ have this properties, where the matrices $\mathcal{F}_3$ and $\mathcal{F}_4$ are shown in the following picture.

$$\begin{pmatrix} + & + & - & 0 & - & 0 \\ - & 0 & + & + & - & 0 \\ - & 0 & - & 0 & + & + \\ - & + & - & + & - & + \end{pmatrix} \qquad \begin{pmatrix} + & + & 0 & + & - \\ + & 0 & 0 & + & - \\ - & 0 & + & + & - \\ + & - & - & + & + \end{pmatrix}$$

Figure 3.4: Matrices $\mathcal{F}_3$ and $\mathcal{F}_4$.

Theorem 3.12

(a) The inequalities $\mathcal{F}_3 \circ x_{IJ} \leq 8$, for all $(4,6)$-index sets, define facets of $P^{m,n}_{\mathrm{C1R}}$ for all $m \geq 4$ and $n \geq 6$.

(b) The inequalities $\mathcal{F}_4 \circ x_{IJ} \leq 8$, for all $(4,5)$-index sets, define facets of $P^{m,n}_{\mathrm{C1R}}$ for all $m \geq 4$ and $n \geq 5$.

Proof. Using the software PORTA we could verify that $\mathcal{F}_3 \circ x \leq 8$ is facet-defining for $P^{4,6}_{\mathrm{C1R}}$ and $\mathcal{F}_4 \circ x \leq 8$ is facet-defining for $P^{4,5}_{\mathrm{C1R}}$. The result then follows from trivial lifting. □

We now must make sure that all forbidden matrices are cut off at least once.

Lemma 3.13 *The facet-defining inequalities from theorems 3.9, 3.11 and 3.12 cut off all Tucker matrices occurring in figure 3.1.*

Proof. It is easily verified that

i) $\mathcal{F}_{1_k} \circ M_{1_k} = 2k + 4$,

ii) $\mathcal{F}_{1_{k+1}} \circ M_{2_k} = 2k + 6$,

iii) $\mathcal{F}_{2_k} \circ M_{3_k} = 2k + 4$,

iv) $\mathcal{F}_3 \circ M_4 = 9$,

v) $\mathcal{F}_4 \circ M_5 = 9$.

Therefore every Tucker matrix violates one of these inequalities. □

Based on the above results we obtain the following integer programming formulation of WC1P. It consists only of inequalities which are facet-defining for $P^{m,n}_{\mathrm{C1R}}$.

$$
\begin{array}{rll}
 & \min\ c^T x & \\
\mathcal{F}_{1_k} \circ x_{IJ} & \leq 2k+3 & \text{for all } (k+2,k+2)\text{-tuples } IJ \text{ and } k \geq 1, \\
\mathcal{F}_{2_k} \circ x_{IJ} & \leq 2k+3 & \text{for all } (k+2,k+3)\text{-tuples } IJ \text{ and } k \geq 1, \\
\mathcal{F}_3 \circ x_{IJ} & \leq 8 & \text{for all } (4,6)\text{-tuples } IJ, \\
\mathcal{F}_4 \circ x_{IJ} & \leq 8 & \text{for all } (4,5)\text{-tuples } IJ, \\
x_{ij} & \in \{0,1\} & \text{for all } i=1,\ldots,m,\ j=1,\ldots,n.
\end{array}
$$

For fixed k there are $\binom{m}{k+2}\binom{n}{k+2}((k+2)!)^2$ possibilities to choose the $(k+2,k+2)$-tuples IJ, where the $\mathcal{F}_{1_k}$ inequalities are defined on. Since k varies from 1 to $\min\{m-2, n-2\}$ the total number of $\mathcal{F}_{1_k}$ inequalities is exponential in the input size. The same holds for the $\mathcal{F}_{2_k}$ inequalities. Nevertheless in section 5.3.1 we will show that the separation problem associated with this IP formulation can be solved in polynomial time using a shortest path algorithm to separate the staircase inequalities.

3.5 Facets derived from betweenness polytopes

This section presents a completely different way to obtain facets for $P^{m,n}_{\text{C1R}}$. We relate the consecutive ones problem to the betweenness problem with the intention to realize connections between their associated polytopes. We will prove some results on the facet structure of the betweenness polytope and show how facets of this polytope can be used to generate facets of $P^{m,n}_{\text{C1R}}$.

3.5.1 The betweenness problem

The input of the **betweenness problem** consists of a set of n objects $1, 2, \ldots, n$ and a set $\mathcal{B}$ of betweenness conditions. Every element of $\mathcal{B}$ is a triple (i, j, k) requesting that object j should be placed between objects i and k. The task is to find an ordering of the objects such that as few betweenness conditions as possible are violated. If violations are penalized by weights, we call the problem of finding an ordering which minimizes the sum of penalties the **weighted betweenness problem (WBWP)**.

Both the WC1P and a variant of the WBWP occur as models in the physical mapping problem. For more details on this application see section 6.1.

3.5.2 The betweenness polytope P^n_{BW}

In the following we use indices $i(j)k$ **(betweenness triples)** for pairwise different objects i, j and k, indicating that we consider whether object j is between objects i and k or not. Since the indices $i(j)k$ and $k(j)i$ are equivalent, we only use $i(j)k$ such that $i < k$. In vectors, triples are ordered lexicographically, i.e., we use the order

$$1(2)3, 1(2)4, \ldots, n-1(n-2)n.$$

For each permutation π of $n \geq 3$ elements $1, \ldots, n$ and each betweenness triple $i(j)k$ we define an indicator $\chi^{\pi}_{i(j)k}$ which is 1 if and only if the element j lies between the elements

i and k in the permutation π and 0 otherwise, i.e.,

$$\chi^{\pi}_{i(j)k} = \begin{cases} 1 & : \text{ if } \pi^{-1}(i) < \pi^{-1}(j) < \pi^{-1}(k) \text{ or } \pi^{-1}(i) > \pi^{-1}(j) > \pi^{-1}(k) \ , \\ 0 & : \text{ otherwise.} \end{cases}$$

The $3\binom{n}{3}$-dimensional characteristic betweenness vector associated with a permutation π is then

$$\chi^{\pi} = (\chi^{\pi}_{1(2)3}, \chi^{\pi}_{1(2)4}, \ldots, \chi^{\pi}_{n-1(n-2)n}).$$

Now we can define the associated polytope.

Definition 3.14 *The* **betweenness polytope** *for $n \geq 3$ is defined as*

$$\boldsymbol{P^n_{\mathrm{BW}}} = \mathrm{conv}\{\chi^{\pi} \mid \pi \textit{ is a permutation of } \{1, \ldots, n\}\}.$$

In contrast to $P^{m,n}_{\mathrm{C1R}}$ the betweenness polytope is not full-dimensional as the following results show.

Lemma 3.15 *For an arbitrary point $x = (x_{1(2)3}, \ldots, x_{n-1(n-2)n}) \in P^n_{\mathrm{BW}}$ and three pairwise different i, j, k, $1 \leq i, j, k \leq n$, the equation* **(betweenness equation)**

$$x_{i(j)k} + x_{i(k)j} + x_{j(i)k} = 1$$

holds.

Proof. For each permutation π and pairwise different i, j and k exactly one of the three indicators $\chi^{\pi}_{i(j)k}$, $\chi^{\pi}_{i(k)j}$ and $\chi^{\pi}_{j(i)k}$ is 1. Thus the equations hold for any vertex of P^n_{BW} and with it for any point of P^n_{BW}. □

Up to linear combinations these equations are the only ones that are valid for P^n_{BW}.

Theorem 3.16 *P^n_{BW} has dimension $2\binom{n}{3}$.*

Proof. No variable $x_{i(j)k}$ appears twice in the betweenness equations. Therefore the coefficient matrix of these equations has full row rank $\binom{n}{3}$.

It remains to be shown that all further equations that are valid for P^n_{BW} are linear combinations of the betweenness equations implying that the dimension of P^n_{BW} is equal to $3\binom{n}{3} - \binom{n}{3}$.

We conclude a contradiction from the assumption that there is an equation $a^T x = a_0$ valid for P^n_{BW}, $n \geq 4$, which cannot be written as linear combination of betweenness equations. Then there is at least one betweenness triple $i(j)k$ with $a_{i(j)k} \neq a_{i(k)j}$. Without loss of generality we set $i = 1$, $j = 2$ and $k = 3$. Now we construct an equation $\tilde{a}^T x = \tilde{a}_0$ with $\tilde{a}_{1(2)3} \neq \tilde{a}_{1(3)2}$ and show that it is valid for P^4_{BW}, which is a contradiction, because P^4_{BW} does not contain such an equation as we have verified by using the software PORTA.

The coefficients of $\tilde{a}^T x = \tilde{a}_0$ are defined as follows (note that always $i < k$).

$$\tilde{a}_{i(j)k} = \begin{cases} a_{i(j)k} & : \text{ if } i,j,k \leq 3, \\ \dfrac{1}{n-3} \displaystyle\sum_{l=4}^{n} a_{i(j)l} & : \text{ if } k = 4, \\ \dfrac{1}{n-3} \displaystyle\sum_{l=4}^{n} a_{i(l)k} & : \text{ if } j = 4. \end{cases}$$

Now it remains to show that for each permutation $\tilde{\pi} \in S(4)$ the value $\tilde{a}^T \chi^{\tilde{\pi}}$ is a constant value not depending on $\tilde{\pi}$. We construct $(n-3)!$ permutations $\pi_1, \pi_2, \ldots, \pi_{(n-3)!} \in S(n)$ by replacing the element 4 in $\tilde{\pi}$ by all permutations of the $(n-3)$ elements $4, \ldots, n$. According to our assumption $\frac{1}{(n-3)!} \sum_{p=1}^{(n-3)!} a^T \chi^{\pi_p} = a_0$ holds. Therefore it remains to show that

$$\tilde{a}^T \chi^{\tilde{\pi}} + a_0 - \frac{1}{(n-3)!} \sum_{p=1}^{(n-3)!} a^T \chi^{\pi_p}$$

is a constant. After a short calculation we see that this value equals

$$a_0 - \frac{1}{2} \sum_{i=1}^{3} \sum_{j,k \geq 4} a_{i(j)k} - \frac{1}{3} \sum_{i,j,k \geq 4} a_{i(j)k}.$$

Obviously this term, which is the right hand side $\tilde{a}_0$ of the constructed equation, does not depend on $\chi^{\tilde{\pi}}$ anymore. □

In contrast to the consecutive ones polytope, trivial inequalities only define facets for the betweenness polytope in the smallest case.

Theorem 3.17

(a) The complete linear description of P^3_{BW} is given by the betweenness equation $x_{1(2)3} + x_{1(3)2} + x_{2(3)1} = 1$ and the three trivial inequalities $x_{1(2)3} \geq 0$, $x_{1(3)2} \geq 0$ and $x_{2(1)3} \geq 0$.

(b) For $n \geq 4$ none of the trivial inequalities $x_{i(j)k} \geq 0$ or $x_{i(j)k} \leq 1$ defines a facet.

Proof. The description of P^3_{BW} is easily verified. Now let $n \geq 4$.

For any permutation π the condition $\chi^\pi_{i(j)k} = 1$ forces $\chi^\pi_{i(k)j} = 0$ and $\chi^\pi_{j(i)k} = 0$. Thus the dimension of $\{x \mid x \in P^n_{\mathrm{BW}} \text{ and } x_{i(j)k} = 1\}$ is at most $2\binom{n}{3} - 2$. And consequently the inequality $x_{i(j)k} \leq 1$ cannot be facet-defining for P^n_{BW}.

Similarly, for all permutations of four elements i, j, k, l with $\chi^\pi_{i(j)k} = 0$ the additional equation $\chi^\pi_{i(j)l} = \chi^\pi_{k(j)l}$ holds. This can easily be seen by testing all possibilities. This equation cannot be derived from the betweenness equations. Thus again the dimension of the induced face is at most $2\binom{n}{3} - 2$. □

Using PORTA we found that the complete description of P^4_{BW} consists of the four betweenness equations and 12 additional facets, all defined by inequalities of the form

$x_{i(j)k} + x_{i(k)j} + x_{j(i)l} + x_{k(i)l} \geq 1$. The outer description of P^5_{BW} needs 10 additional classes of inequalities. Appendix A.4 shows one representative of each class.

Since we have equations, the same facet-defining inequality can be stated in various ways. Therefore we define a **normal form** with the property that two facet-defining inequalities define the same facet if and only if their normal forms coincide.

Definition 3.18 *A facet-defining inequality of P^n_{BW} is stated in normal form if the following properties hold.*

i) The inequality is written as $\sum a_{i(j)k} x_{i(j)k} \geq a_0$.

ii) All coefficients $a_{i(j)k}$ are nonnegative coprime integers.

iii) At least one of the three coefficients $a_{i(j)k}$, $a_{i(k)j}$ and $a_{j(i)k}$ is zero for pairwise different elements i, j and k, $1 \leq i, j, k \leq n$.

It is easily seen that the normal form of a facet-defining inequality is unique and that it is easy to convert an inequality to normal form.

3.5.3 A common polytope

Both the feasible solutions of the WBWP and of the WC1P are based on the permutations of n elements. In order to examine relations between the two problems we define a master problem combining their constraints. Here we seek for a permutation satisfying betweenness conditions as well as having an associated matrix where the ones appear consecutively. This would model the practical situation where a consecutive ones matrix has to be found that satisfies in addition certain betweenness conditions.

Definition 3.19 *The common betweenness and consecutive ones polytope $P^{m,n}_{\mathrm{BWC1R}}$ is defined as*

$$P^{m,n}_{\mathrm{BWC1R}} = \mathrm{conv}\{(\chi^\pi, \chi^A) \mid A \in \{0,1\}^{(m,n)} \text{ and } \pi \text{ establishes C1PR of } A\}.$$

The single polytopes can be simply obtained from $P^{m,n}_{\mathrm{BWC1R}}$.

Remark 3.20 *The projection of $P^{m,n}_{\mathrm{BWC1R}}$ on the betweenness variables $x_{i(j)k}$ is the betweenness polytope P^n_{BW} and the projection on the consecutive ones variables x_{ri} is the consecutive ones polytope $P^{m,n}_{\mathrm{C1R}}$.*

Proof. Clearly the restriction of any vertex of $P^{m,n}_{\mathrm{BWC1R}}$ on the betweenness variables, the consecutive ones variables, respectively, is a vertex of P^n_{BW}, a vertex of $P^{m,n}_{\mathrm{C1R}}$, respectively. It remains to be shown that all vertices of P^n_{BW} and $P^{m,n}_{\mathrm{C1R}}$ can be reached by projection.

If A is the zero matrix then every permutation π establishes C1PR of A. Thus for every vertex v of P^n_{BW} (which is associated to a permutation π) there is at least one vertex v' of $P^{m,n}_{\mathrm{BWC1R}}$ with $v' = (v, *)$.

Similarly, every vertex w of $P^{m,n}_{\text{C1R}}$ is associated with a matrix A that is C1PR. Therefore there is at least one permutation establishing C1PR of A. And with it there is a vertex w' of $P^{m,n}_{\text{BWC1R}}$ such that $w' = (*, w)$. □

Of course, all valid inequalities for $P^{m,n}_{\text{C1R}}$ or P^{n}_{BW} remain valid for $P^{m,n}_{\text{BWC1R}}$. We will now construct valid inequalities for $P^{m,n}_{\text{BWC1R}}$ that are formulated both on betweenness and on consecutive ones variables.

Lemma 3.21 *Let $A = (a_{ri})$ be an (m,n)-matrix ($n \geq 3$) with* C1PR*. Then for all rows r of A, all betweenness triples $i(j)k$ of columns i, j, k of A and all permutations π that establish* C1PR *of A we have*

$$\chi^{\pi}_{i(j)k} \leq 2 - a_{r\pi^{-1}(i)} + a_{r\pi^{-1}(j)} - a_{r\pi^{-1}(k)}.$$

Proof. The only possibility to violate this inequality is to set $\chi^{\pi}_{i(j)k} = 1$, $a_{r\pi^{-1}(i)} = 1$, $a_{r\pi^{-1}(j)} = 0$ and $a_{r\pi^{-1}(k)} = 1$. But according to the definition $\chi^{\pi}_{i(j)k} = 1$ means $\pi^{-1}(i) < \pi^{-1}(j) < \pi^{-1}(k)$ or $\pi^{-1}(i) > \pi^{-1}(j) > \pi^{-1}(k)$. Both cases force the 0-entry $a_{r\pi^{-1}(j)}$ to lie between the 1-entries $a_{r\pi^{-1}(i)}$ and $a_{r\pi^{-1}(k)}$. Thus the ones do not occur consecutively in row r of A in contradiction to the choice of π. □

Based on this observation we can define so-called **linking constraints**.

Theorem 3.22 *The inequalities*

$$x_{i(j)k} \leq 2 - x_{ri} + x_{rj} - x_{rk}$$

are valid for $P^{m,n}_{\text{BWC1R}}$ for all $r \in \{1, \ldots, m\}$ and all betweenness triples $i(j)k$.

Proof. According to Lemma 3.21 the inequalities are fulfilled by all vertices of $P^{m,n}_{\text{BWC1R}}$. Therefore they are valid for the polytope. □

Intuitively there is a close relationship between the consecutive ones and the betweenness problem. We now establish a connection between the facets of the two polytopes by making use of the linking constraints. These constraints are used to eliminate betweenness variables and replace them by consecutive ones variables.

Theorem 3.23 *Let $a^T x \geq a_0$ be a facet-defining inequality for P^{n}_{BW}, $n \geq 4$, in normal form. Further let m be the number of nonzero coefficients of a. We assign pairwise different numbers $r_{i(j)k} \in \{1, \ldots, m\}$ to the betweenness triples $i(j)k$ with $a_{i(j)k} > 0$. Let the inequality $B \circ x \leq b_0$ be obtained by summing up $-a^T x \leq -a_0$ and all (scaled) linking constraints $a_{i(j)k} x_{i(j)k} \leq a_{i(j)k}(2 - x_{r_{i(j)k}i} + x_{r_{i(j)k}j} - x_{r_{i(j)k}k})$. Then $B \circ x \leq b_0$ is facet-defining for $P^{m,n}_{\text{C1R}}$.*

Proof. The validity of $B \circ x \leq b_0$ for the common polytope $P^{m,n}_{\text{BWC1R}}$ is clear because it was constructed as the sum of valid inequalities. Now since the projection of $P^{m,n}_{\text{BWC1R}}$ on the consecutive ones variables is $P^{m,n}_{\text{C1R}}$ (see remark 3.20) and since $B \circ x \leq b_0$ contains no betweenness variables the inequality is also valid for $P^{m,n}_{\text{C1R}}$. It remains to show that it is facet-defining. For this purpose we use the same technique as in the proof of theorem

3.9. Again let $F = \{x \mid a^T x = a_0\} \cap P^{m,n}_{\mathrm{C1R}}$ denote the induced face of $B \circ x \leq b_0$ and let $D \circ x \leq d_0$ be any facet-defining inequality for $P^{m,n}_{\mathrm{C1R}}$ such that $F \subseteq \{x \mid D \circ x = d_0\} \cap P^{m,n}_{\mathrm{C1R}}$. We will show that both inequalities only differ by multiplication with a positive scalar.

For a given permutation π whose characteristic betweenness vector χ^π fulfills $a^T x \geq a_0$ with equality we construct a standard solution x^π as follows.

$$x^\pi_{r_{i(j)k}c} = \begin{cases} 1 & : \text{ if } \chi^\pi_{i(j)k} = 1 \text{ and } c = i, \\ 0 & : \text{ if } \chi^\pi_{i(j)k} = 1 \text{ and } c \neq i, \\ 1 & : \text{ if } \chi^\pi_{i(j)k} = 0 \text{ and } c = i, \\ 1 & : \text{ if } \chi^\pi_{i(j)k} = 0 \text{ and } c = k, \\ 1 & : \text{ if } \chi^\pi_{i(j)k} = 0 \text{ and } \chi^\pi_{i(c)k} = 1, \\ 0 & : \text{ if } \chi^\pi_{i(j)k} = 0 \text{ and } \chi^\pi_{i(c)k} = 0. \end{cases}$$

One can easily calculate that for each row $r = r_{i(j)k}$ the corresponding linking constraint $\chi^\pi_{i(j)k} \leq 2 - x^\pi_{ri} + x^\pi_{rj} - x^\pi_{rk}$ holds with equality. All inequalities which add up to $B \circ x^\pi \leq b_0$ are fulfilled with equality with the consequence that $x^\pi \in F$.

In the following we consider an arbitrary but fixed row $r = r_{i(j)k}$ of D. Since $a^T x \geq a_0$ is facet-defining for P^n_{BW} there is at least one ordering ρ with $a^T \chi^\rho = a_0$ and $\chi^\rho_{i(j)k} = 1$. Otherwise $x_{i(j)k} = 0$ would hold for all solutions of $a^T x = a_0$ contradicting the facet-defining property of $a^T x \geq a_0$ since this equation cannot be written as linear combination of the betweenness equations. Now consider the standard solution x^ρ. We construct another solution $\tilde{x}^\rho$ from x^ρ by setting $\tilde{x}^\rho_{ri} = 0$ and $\tilde{x}^\rho_{rk} = 1$. Then we have

$$0 = D \circ x^\rho - D \circ \tilde{x}^\rho = d_{ri} - d_{rk}$$

and with it $d_{ri} = d_{rk} := \delta$. Taking a column i' with $b_{ri'} = 0$ and which lies next to i in the permutation ρ we construct another solution $\bar{x}^\rho$ from x^ρ by setting $\bar{x}^\rho_{ri'} = 1$. This leads to

$$0 = D \circ x^\rho - D \circ \bar{x}^\rho = -d_{ri'}.$$

Very similar to the proof of theorem 3.9 we can extend this to a series of solutions starting from x^ρ and $\tilde{x}^\rho$ to show that $d_{rc} = 0$ for $c \notin \{i, j, k\}$.

Finally let $\hat{x}^\rho$ be the solution obtained from x^ρ by setting all entries of row r to 1. Since most of the coefficients of $D_{r\cdot}$ are already shown to be zero we get

$$0 = D \circ x^\rho - D \circ \hat{x}^\rho = -d_{rj} - d_{rk}$$

and therefore $d_{rj} = -\delta$.

The same arguments can be performed for any row. Thus we have shown that $D = \delta B$ holds. Since the solution consisting of zeros only contradicts $\delta < 0$ we are done.

□

By this construction (which in fact corresponds to projecting out some variables of a system of inequalities) we can obtain for each facet class of P^n_{BW} a facet class of $P^{m,n}_{\mathrm{C1R}}$.

Example 3.24 *Consider for example the inequality*

$$x_{1(2)3} + x_{1(3)2} + x_{2(1)4} + x_{3(1)4} \geq 1$$

which is facet-defining for P^4_{BW} *and in normal form. The four linking constraints*

$$\begin{aligned} -x_{1(2)3} &\geq -2 + x_{11} - x_{12} + x_{13} \\ -x_{1(3)2} &\geq -2 + x_{21} - x_{23} + x_{22} \\ -x_{2(1)4} &\geq -2 + x_{32} - x_{31} + x_{34} \\ -x_{3(1)4} &\geq -2 + x_{43} - x_{41} + x_{44} \end{aligned}$$

are valid for $P^{4,4}_{\mathrm{BWC1R}}$.

Summing up these five inequalities (thus eliminating the betweenness variables) and multiplying by -1 *yields the staircase inequality*

$$\begin{pmatrix} 1 & -1 & 1 & 0 \\ 1 & 1 & -1 & 0 \\ -1 & 1 & 0 & 1 \\ -1 & 0 & 1 & 1 \end{pmatrix} \circ x \leq 7.$$

According to theorem 3.23 this inequality is facet-defining for $P^{4,4}_{\mathrm{C1R}}$.

If we start with a facet $a^T x \geq a_0$ of P^n_{BW} in normal form, clearly the support of the constructed facet of $P^{m,n}_{\mathrm{C1R}}$ has at most n columns. But what about the number of rows m? According to the construction, m is the number of nonzero coefficients of a. Since at least one of three coefficients is zero, we have $m \leq 2\binom{n}{3}$. Taking into account that these facets can be trivially lifted to facets of $P^{m',n}_{\mathrm{C1R}}$ with $m' > m$ the total number of constructed facets of $P^{m,n}_{\mathrm{C1R}}$ for fixed n and arbitrary m is $O(m^{2\binom{n}{3}})$ and thus polynomial in m. Unfortunately, not every facet can be constructed in this way. But in the following section we will show the surprising result that for fixed n (fixed m) also the total number of facets of $P^{m,n}_{\mathrm{C1R}}$ is polynomial in m (in n) with the consequence that the WC1P is polynomially solvable for fixed n or m.

3.6 Number of facets for a fixed number of rows or columns

The left hand sides of the facets constructed in the last section have the common property that in each row the number of nonzeros is 3, corresponding to a betweenness triple. To take general facet defining inequalities into account one has to consider not only interactions of three but of arbitrarily many columns. Moreover, a single betweenness triple only models a special relation of columns. But we are also interested in interactions of rows. For these purposes we introduce the concept of feasible sets both of rows and of columns.

Definition 3.25 *A set $C = \{c_1, \ldots, c_k\}$ where $c_i \in \{0,1\}^{(m,1)}$, for $i = 1, \ldots, k$, of columns is called* **consecutive ones feasible for columns (C1FC)** *if the matrix consisting of all columns of C is* C1PR.

Definition 3.26 *A C1FC set C is called* **consecutive ones maximal for columns (C1MC)** *if C is maximal with respect to set inclusion.*

Example 3.27 *Let $m = 3$. The set*

$$C = \left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \right\}$$

is C1FC but not C1MC since one can add the column $(1\ 1\ 1)^T$ and the resulting set

$$\overline{C} = \left\{ \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \right\}$$

is still C1FC. But any further column $(1\ 0\ 0)^T$ or $(1\ 0\ 1)^T$ would lead to a set whose corresponding matrix is not C1PR. *Therefore $\overline{C}$ is C1MC.*

Note that C1FC and C1MC sets do not contain duplicate columns. This contrasts with matrices that are C1PR.

Lemma 3.28 *The cardinality of all C1MC sets for m rows is $2m$.*

Proof. Let $C = \{c_1, \ldots, c_k\}$ be C1MC and M a corresponding matrix such that the ones in every row of M occur consecutively and the rows are ordered lexicographically.

Then the first column of M is 0. Assume without loss of generality that the columns occur in order $c_1, \ldots, c_k$ in C. Let $|c_i - c_j|$ be the number of entries in which c_i and c_j differ. Since C is C1MC there is at least one "1"-entry in every row of M. Otherwise one could add a further column as the first column which has a "1"-entry in that row and "0"-entries otherwise.

Because the "1"-entries occur consecutively in all rows we have the equation

$$\sum_{i=1}^{k} |c_i - c_{i+1}| = 2m$$

(where c_{k+1} is identified with c_1). Since the c_i are pairwise different it follows that $\sum_{i=1}^{k} |c_i - c_{i+1}| \geq k$. On the other hand, if there were a consecutive pair of columns satisfying $|c_i - c_{i+1}| > 1$, one could add an additional different column between these two contradicting the maximality of C. Therefore we obtain that $\sum_{i=1}^{k} |c_i - c_{i+1}| = k$ which implies that $k = 2m$. □

As immediate consequence of this lemma we can give an upper bound on the number of C1MC sets.

Corollary 3.29 *The number of C1MC sets for m rows is at most $\binom{2^m}{2m}$.*

Proof. The result is clear since the number of all possible columns is 2^m and since the cardinality of all C1MC sets for m rows is $2m$. □

Analogously to the definition of feasible sets for columns we can introduce feasible sets for rows.

Definition 3.30 *A set $R = \{r_1, \ldots, r_k\}$ where $r_i \in \{0,1\}^{(1,n)}$, for $i = 1, \ldots, k$, of rows is called* **consecutive ones feasible for rows (C1FR)** *if the matrix consisting of all rows of R is C1P.*

Definition 3.31 *A C1FR set R is called* **consecutive ones maximal for rows (C1MR)** *if R is maximal with respect to set inclusion.*

Example 3.32 *Let $n = 3$. The set*

$$R = \left\{ \begin{array}{c} (0\ 0\ 0) \\ (0\ 0\ 1) \\ (0\ 1\ 0) \\ (1\ 0\ 0) \\ (0\ 1\ 1) \\ (1\ 1\ 1) \end{array} \right\}$$

is C1FR but not C1MR since one can add the row $(1\,1\,0)$ and the resulting set

$$\overline{R} = \left\{ \begin{array}{c} (0\ 0\ 0) \\ (0\ 0\ 1) \\ (0\ 1\ 0) \\ (1\ 0\ 0) \\ (0\ 1\ 1) \\ (1\ 1\ 0) \\ (1\ 1\ 1) \end{array} \right\}$$

is still C1FR. But any further row ($(1\,0\,1)$ is the only one remaining) would lead to a set which cannot be brought into C1-structure. Therefore $\overline{R}$ is C1MR. Analogously to the previous case C1FR and C1MR sets do not contain duplicate rows.

The smallest set that is not C1FR consists of the three rows $(1\,1\,0)$, $(1\,0\,1)$ and $(0\,1\,1)$. Here the relationship with the concept of betweenness triples is observable. The three rows correspond to the triples $1(3)2$, $1(2)3$ and $2(1)3$ and it is not possible to fulfill all three betweenness conditions simultaneously.

Lemma 3.33 *The cardinality of all C1MR sets for n columns is $\frac{n(n+1)}{2} + 1$. The total number of different C1MR sets for n columns is $n!/2$.*

Proof. Let R be a C1MR set of rows with n columns. Further let π be a permutation of the columns that establishes C1PR for each row. Without loss of generality we assume that π is the identity. Then all rows of R must be of the form

$$(0 \ldots 0\ 1 \ldots 1\ 0 \ldots 0).$$

There is one such row with no "1"-entry, n rows with one "1"-entry, $n-1$ rows with two "1"-entries and so on up to a single row with n "1"-entries. Because of the maximality of R none of these rows must be missing. Therefore by adding all these numbers we obtain that the total number of rows in R is equal to $\frac{n(n+1)}{2} + 1$.

For two different column permutations π_1 and π_2 the corresponding C1MR sets are equal if and only if π_2 is the reverse ordering of π_1. This can be seen by viewing the subsets of the corresponding C1MR sets consisting of the $n-1$ rows containing exactly 2 ones. These rows correspond to pairs of neighboring elements of π_1 or π_2, respectively. These 2 sets of pairs are identical if and only if π_1 and π_2 define the same permutation or one defines the reverse of the other. As a consequence the number of different C1MR sets is $n!/2$. □

Note that due to the definition of C1PR, a matrix is C1PR if and only if it consists of (an arbitrary number of) rows from a set R which is C1MR (columns from a set C which is C1MC, resp.). Further, for any matrix $M \in \{0,1\}^{(m,n)}$ which is C1PR there is at least one set R which is C1MR and contains all different rows of M as a subset and at least one set C which is C1MC and contains all different columns of M.

Remark 3.34 *All C1MC sets have the same cardinality. The same holds for the C1MR sets. Therefore both independence systems define* matroids. *Unfortunately the objective function of* WC1P *is not directly defined on rows or columns but on single entries of the matrix. Furthermore duplicated rows or columns are allowed in feasible* WC1P *solutions. Otherwise one could derive a polynomial algorithm for* WC1P *from a simple greedy algorithm on the matroid.*

See [KV00] for more details on independence systems and matroids.

Now with the help of C1MC and C1MR sets we are able to prove the surprising result that for a fixed number of rows (columns resp.) the number of facets of $P^{m,n}_{\text{C1R}}$ grows only polynomially in the number of columns (rows resp.). This contrasts with many combinatorial optimization problems where the number of facets of the corresponding polytopes grow exponentially, though the problems are polynomially solvable.

Theorem 3.35 *The number of facets of $P^{m,n}_{\text{C1R}}$ for fixed m is $O(n^{\binom{2^m}{2m}})$.*

Proof. Let m be the fixed number of rows.

Suppose $B \circ x \leq b_0$ is a nontrivial facet-defining inequality for $P^{m,n}_{\text{C1R}}$. Since the zero matrix and all matrices consisting of zeros except for one entry are feasible solutions it follows that $b_0 > 0$.

Let l be the number of C1MC sets C consisting of columns $\in \{0,1\}^{(m,1)}$ with the property that there exists a matrix $M \in \{0,1\}^{(m,n)}$ with $B \circ M = b_0$ and all columns of

M are in C. According to corollary 3.29 we have $l \leq \binom{2^m}{2m}$. Our goal is to show that the support of B has at most l columns.

For each of these chosen sets $C \in \{C_1, \ldots, C_l\}$ and every column j of B we define

$$m_j^C(B) = \max\{B_{.j}^T v \mid v \in C\}$$

as the maximum possible contribution of column j to the left hand side of the facet.

Furthermore, for all C1PR matrices M with $B \circ M = b_0$ and all C1MC sets C containing all columns of M the relation

$$B_{.j}^T M_{.j} = m_j^C(B)$$

holds for every column j. The relation $B_{.j}^T M_{.j} \leq m_j^C(B)$ is clear from the definition of $m_j^C(B)$. If we assume that $B_{.j}^T M_{.j} < m_j^C(B)$ we can construct a new C1PR matrix M' by replacing the column j of M by the maximum column v from the above definition. But then $B \circ M' > b_0$ contradicting the validity of $B \circ x \leq b_0$.

Now let k be the number of nonzero columns of B. We create the (l, k)-matrix

$$\mathcal{M}(B) = (m_{ij}(B)) = (m_j^{C_i}(B)).$$

Of course the rank of $\mathcal{M}(B)$ is at most l independently of the number k of columns.

Now assume that a facet-defining inequality $B \circ x \leq b_0$ is given with the number of nonzero columns k of B greater than l. Because of $\operatorname{rank} \mathcal{M}(B) \leq l$ at least one column j of $\mathcal{M}(B)$ can be written as a linear combination $\mathcal{M}(B)_{.j} = \sum_{j' \neq j} d_{j'} \mathcal{M}(B)_{.j'}$. And since $B_{.j}^T M_{.j} = m_j^{C_i}(B)$ holds for any C1PR matrix M with $B \circ M = b_0$ and a suitable C1MC set C_i containing the different columns of M we have

$$B_{.j}^T M_{.j} = m_j^{C_i}(B) = \sum_{j' \neq j} d_{j'} m_{j'}^{C_i}(B) = \sum_{j' \neq j} d_{j'} B_{.j'}^T M_{.j'}.$$

This equation holds for every vertex M of $\{x \in P_{\mathrm{C1R}}^{m,n} \mid B \circ x = b_0\}$. And since it contains no constant coefficient it cannot be obtained by scaling the equation $B \circ x = b_0$ with $b_0 > 0$. Therefore the inequality $B \circ x \leq b_0$ cannot be facet-defining.

Thus the support of every facet-defining inequality for $P_{\mathrm{C1R}}^{m,n}$ has at most $l \leq \binom{2^m}{2m}$ columns and therefore each of these facets can be obtained by trivial lifting from a facet of $P_{\mathrm{C1R}}^{m,\binom{2^m}{2m}}$. Since the number of facets of $P_{\mathrm{C1R}}^{m,\binom{2^m}{2m}}$ is constant in n and the number of lifting possibilities for one facet is at most $\binom{n}{\binom{2^m}{2m}}$ the total number of facet-defining inequalities for $P_{\mathrm{C1R}}^{m,n}$ is $O(n^{\binom{2^m}{2m}})$. □

Theorem 3.36 *The number of facets of $P_{\mathrm{C1R}}^{m,n}$ for fixed n is $O(m^{n!/2})$.*

Proof. The proof follows along the same lines as the proof of the previous theorem. Since for fixed n the number of C1MR sets R is equal to $n!/2$, the total number of facet-defining inequalities for $P_{\mathrm{C1R}}^{m,n}$ for fixed n is of order $O(m^{n!/2})$. □

3.6.1 Polynomial solvability

As a consequence of the results on the number of facets we obtain that WC1P is solvable in polynomial time for fixed n or fixed m. Note that for a fixed number of columns this result has already been shown in section 2.4.3.

Corollary 3.37 WC1P *is solvable in polynomial time for fixed n or fixed m.*

Proof. Consider the case that the number n of columns is fixed. According to the discussion above all facets of $P_{\mathrm{C1R}}^{m,n}$ can be obtained by trivial lifting from facets of $P_{\mathrm{C1R}}^{n!/2,n}$. Computing all of these facets takes time constant in m. And for each of these facets there are at most $\binom{m}{n!/2}$ possibilities for trivial lifting. Thus we need time $O(m^{n!/2})$ to create a complete listing of all facets of $P_{\mathrm{C1R}}^{m,n}$.

Since the number of facets as well as their encoding length is polynomial in m, we can list them all explicitly in polynomial time and use a polynomial algorithm for linear programming to minimize the objective function over the exact linear description of $P_{\mathrm{C1R}}^{m,n}$.

An analogous argumentation applies to the case where m is fixed. □

Linear programming provides one means of solving the WC1P for fixed m or n in polynomial time. However, looking more closely into the combinatorial structure of the problem, we can even derive a linear time algorithm.

Theorem 3.38 WC1P *is solvable in linear time for fixed n or fixed m.*

Proof. We only consider the case that m is fixed as the discussion for fixed n is similar. Besides, the latter case has been proven by corollary 2.23 in a different manner.

Let the WC1P be formulated as

$$\max\{B \circ x \mid x \in \{0,1\}^{(m,n)} \text{ is C1PR}\}.$$

Now for each C1MC set C and for each column c of B we compute $m_c^C(B)$. One computation takes time $O(2m^2)$. Therefore all of these calculations take time $O(2m^2\binom{2^m}{2m}n)$. Since

$$\begin{aligned} & \max\{B \circ x \mid x \in \{0,1\}^{(m,n)} \text{ is C1P}\} \\ = \; & \max\Big\{\sum_j m_j^C(B) \mid C \text{ is a C1MC set}\Big\} \end{aligned}$$

holds, we are done. The total running time of this algorithm is $O(2m^2\binom{2^m}{2m}n)$ and thus linear in n. □

When fixing the number n of columns, we obtain a running time $O(n^3 n!m)$ which can be improved to $O(n!nm)$ by making use of the scan line method described in section 2.4.3.

3.7 Outer descriptions of small polytopes

We want to conclude this chapter with some notes on the complete description of polytopes for few rows or columns.

If $m \leq 2$ or $n \leq 2$, all (m, n)-matrices are C1PR and therefore the trivial inequalities completely describe $P^{m,n}_{\mathrm{C1R}}$.

For $n = 3$ there are $n!/2 = 3$ C1MR sets. According to theorem 3.36 the support of any facet-defining inequality for $P^{m,3}_{\mathrm{C1R}}$ has at most 3 rows. Therefore the complete descriptions of $P^{m,3}_{\mathrm{C1R}}$ for $m \geq 3$ only consist of lifted facets for $P^{3,3}_{\mathrm{C1R}}$ which are the trivial inequalities and the smallest class of staircase inequalities $\mathcal{F}_{1_1}$

$$\begin{pmatrix} 1 & 1 & -1 \\ 1 & -1 & 1 \\ -1 & 1 & 1 \end{pmatrix} \circ x \leq 5.$$

For $m = 3$ we have shown by complete enumeration that there are 12 C1MC sets. Analogously, using theorem 3.36 it follows that facet-defining inequalities for $P^{3,n}_{\mathrm{C1R}}$ have at most 12 nonzero columns. We tried to compute the complete descriptions for $4 \leq n \leq 12$ by making use of the software package PORTA [CL98]. Thus the description of $P^{3,4}_{\mathrm{C1R}}$ requires one additional class of facet-defining inequalities, namely the staircase inequality class $\mathcal{F}_{2_1}$

$$\begin{pmatrix} 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \circ x \leq 5.$$

Interestingly the description of $P^{3,5}_{\mathrm{C1R}}$ requires no additional facet classes, but the description of $P^{3,6}_{\mathrm{C1R}}$ does. Appendix A.1 shows the outer description in this case including 3 further classes. Unfortunately computing the complete description of $P^{3,n}_{\mathrm{C1R}}$ for $7 \leq n \leq 12$ was not possible in a reasonable amount of time.

Finally, appendix A.2 shows all facet-defining inequalities of the polytope $P^{4,4}_{\mathrm{C1R}}$. Its complete description requires 9 classes of facet-defining inequalities besides the trivial ones. The total number of facets of $P^{4,4}_{\mathrm{C1R}}$ is 1880.

In section 5.3.2 we will see that such descriptions of small instances together with the trivial lifting property can be very useful for separation purposes.

Chapter 4

The Simultaneous Consecutive Ones Problem

Up to now we have dealt with binary matrices having the consecutive ones property only for rows. Clearly their transposes have the consecutive ones property for columns. But what about matrices that are simultaneously C1PR and C1PC? In the following this simultaneous case will be investigated. Some of the results can easily be derived from the standard case. But there are also some new outcomes. As most surprising result it will turn out that the weighted optimization problem in the simultaneous case remains $\mathcal{NP}$-hard even if the row-permutation and the column-permutation are fixed (see section 4.2.3). We will conclude this chapter with some polyhedral investigations of the simultaneous consecutive ones polytope and as in the standard case we will give an IP formulation of the weighted problem that consists only of facets of the corresponding polytope.

4.1 Basic definitions and results

Both the definition of the simultaneous consecutive ones property and the definition of the corresponding weighted optimization problem can directly be taken over from the nonsimultaneous case. However characterizing 0/1 matrices having the simultaneous consecutive ones property can be accomplished with less forbidden matrices than in the standard case.

0/1 matrices are defined to have the simultaneous consecutive ones property if they have both the consecutive ones property for rows and for columns. Therefore the definition is directly reduced to the definitions 2.1 and 2.3.

Definition 4.1 *An* (m, n) 0/1*-matrix* M *has the* **simultaneous consecutive ones property (is C1PS)** *if* M *is both* C1PR *and* C1PC*.*

Example 4.2 *The matrix*

$$M = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

is C1PS. *Permuting the first and the third column establishes* C1PR

$$M' = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

and an additional switching of the first two rows leads to

$$M'' = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

where both C1PR *and* C1PC *is established.*

Remark 4.3 *As seen in the previous example establishing* C1PR *and* C1PC *of a matrix by suitable column-permutations and row-permutations does not interact each other. Therefore we can take any column-permutation that establishes* C1PR *and any row-permutation that establishes* C1PC *and receive an arrangement of the matrix where both in each row and in each column the ones occur consecutively.*

Analogously to definition 2.17 we define the **weighted simultaneous consecutive ones problem (WSC1P)** as a problem of the form

$$\begin{aligned} & \min C \circ A \\ s.t. \quad & A \text{ is C1PS.} \end{aligned}$$

In section 4.2.2 we will show that the WSC1P is $\mathcal{NP}$-hard This is a consequence of the fact that a corresponding augmentation problem will be shown to be $\mathcal{NP}$-complete (see 4.2.1).

Of course one can characterize C1PS matrices in the same manner as C1PR ones (compare to section 3.2) by taking all forbidden matrices from figure 3.1 and adding all their transposes. But some of the matrices already contain the transposes of smaller ones. For example M_{3_4} contains $M_{2_1}^T$ as submatrix. Therefore not all of these matrices are required in the simultaneous case.

Theorem 4.4 ([Tuc72]) *The* 0/1*-matrix* M *has the simultaneous consecutive ones property if and only if no submatrix of* M*, or of the transpose of* M*, is one of the matrices occurring in figure 4.1.*

As well in this case we can derive an IP formulation for WSC1P by taking a set of valid inequalities that cut off all these matrices and their transposes. Section 4.3.2 will give a tightened version of such an IP formulation.

$\overbrace{\begin{pmatrix} 1 & 1 & & & & \\ & 1 & 1 & & & \\ & & \ddots & \ddots & & \\ & & & & 1 & 1 \\ 1 & & & & & 1 \end{pmatrix}}^{(k+2,k+2)\text{-matrix } M_{1_k}\,(k\geq 1)}$

$\overbrace{\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{pmatrix}}^{(4,4)\text{-matrix } M_{2_1}}$

$\overbrace{\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \end{pmatrix}}^{(5,5)\text{-matrix } M_{2_2}}$

$\overbrace{\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}}^{(3,4)\text{-matrix } M_{3_1}}$

$\overbrace{\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \end{pmatrix}}^{(4,5)\text{-matrix } M_{3_2}}$

$\overbrace{\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}}^{(5,6)\text{-matrix } M_{3_3}}$

Figure 4.1: Tucker matrices M_{1_k}, M_{2_1}, M_{2_2}, M_{3_1}, M_{3_2} and M_{3_3}

4.2 Complexity results

In this section we want to outline some new results on the complexity in the simultaneous case. Although the WSC1P being $\mathcal{NP}$-hard in the general case and being solvable in linear time, if we fix the number of rows or columns, shows the same behavior as in the nonsimultaneous case, an astonishing result will turn out to be the fact that WSC1P remains $\mathcal{NP}$-hard even if no column-permutation and row-permutation are allowed.

4.2.1 The augmentation problem

As mentioned in 2.4.1 our definition of the augmentation problem in the simultaneous case differs from Booth's one [Boo75] in the sense that the roles of zeros and ones are switched. This is because in the simultaneous case this definition has turned out to be more useful to show the $\mathcal{NP}$-completeness of the augmentation problem. As a corollary we get the $\mathcal{NP}$-hardness of the corresponding weighted optimization problem.

Definition 4.5 *A 0/1-matrix M has the* ***k*-augmented simultaneous consecutive ones property (is SC1P$_k$)** *if and only if there exists a matrix M' which is* C1PS *and which arises from M by replacing at most k "1"-entries of M by a "0".*

Example 4.6

$$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

The left matrix is SC1P$_2$ *since switching two entries from "1" to "0" leads to the right matrix which is* C1PS*. This can be seen by permuting the first two rows and the last two rows.*

Similarly to theorem 2.16 we will now show the $\mathcal{NP}$-completeness of the corresponding decision problem by transformation from the Hamiltonian path problem which is known to be $\mathcal{NP}$-complete [GJ79].

Theorem 4.7 *Deciding whether a* 0/1*-matrix* M *is* SC1P$_k$ *is* $\mathcal{NP}$*-complete.*

Proof. Verifying whether a given k-augmentation for M is C1PS can be done in polynomial time using the PQ-tree algorithm both for rows and for columns. It remains to specify a polynomial transformation to the Hamiltonian path problem. Although we use a different augmentation definition and a different $\mathcal{NP}$-complete problem to transform from the construction works very similarly to the one used in the proof of theorem 2.16. Again we construct the 0/1-matrix M as the node-edge incidence matrix of a given a graph $G = (V, E)$. Consequently $M = (m_{ij})$ is defined as

$$m_{ij} = \begin{cases} 1 & : \quad v_j \in e_i \\ 0 & : \quad v_j \notin e_i. \end{cases}$$

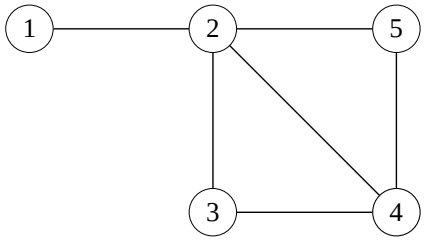
Now we claim that G contains a Hamiltonian path if and only if M is SC1P$_{(|E|-|V|+1)}$. To proof this we first assume that G contains a Hamiltonian path H. Then there are $|E| - |V| + 1$ edges not contained in H. Now in each row of M corresponding to this edges an arbitrary "1"-entry is flipped to "0". In the remaining matrix M' the columns are ordered according to the vertices appearing in H. Since the only rows containing still two "1"s correspond to edges in H the "1"s in each row occur consecutively. After arranging the rows lexicographically also the "1"s in each column must occur consecutively because the maximal difference of the number of "1"s in two rows is 1.

Conversely we assume that M is SC1P$_{(|E|-|V|+1)}$. Let M' be the SC1P matrix which arises from M by replacing $|E| - |V| + 1$ "1"s by "0"s. At most one entry per row can be replaced because otherwise there would be more than $|V|-1$ rows left with two "1"s. Since these rows correspond to edges in the graph there would be at most one cycle consisting of these edges which forces the matrix M' not to be C1PR and with it C1PS. Therefore M' has exactly $|V| - 1$ rows with two "1"-entries. Since M' is C1PS the relative order of these rows in an C1PS-establishing permutation creates a Hamiltonian path in G.

Now assume we are given a polynomial time algorithm which decides whether a given matrix M is SC1P$_k$. Also the construction above can be performed in polynomial time. Therefore we would have been created a polynomial time algorithm for finding a Hamiltonian path in a graph. Thus the $\mathcal{NP}$-completeness is proven. □

Example 4.8 *Figure 4.2 gives an example for the transformation used in the previous proof. It shows a graph containing a Hamiltonian path, the corresponding node-edge incidence matrix on the left-hand side and the modified matrix on the right hand-side which has become* C1PS *after a suitable row permutation and flipping* $2 = |E| - |V| + 1$ *entries*

from "1" to "0". An additional column permutation has not been necessary in this case because the nodes in the Hamiltonian path already occur in the order $1-2-3-4-5$.



$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Figure 4.2: Transformation used for proving theorem 4.7.

4.2.2 The weighted problem

As a corollary of the $\mathcal{NP}$-completeness of the augmentation problem we obtain the $\mathcal{NP}$-hardness of the weighted problem where the objective function is arbitrary.

Corollary 4.9 *The weighted simultaneous consecutive ones problem is $\mathcal{NP}$-hard.*

Proof. Assume we are given a polynomial time algorithm solving WSC1P to optimality. Given a 0/1-matrix A we construct the objective function $C = -A$. Let l be the number of "1"-entries of A. Then A is SC1P$_k$ if and only if the solution of

$$\begin{aligned} & \min C \circ A \\ s.t. \quad & A \text{ is C1PS} \end{aligned}$$

is less or equal to $k - l$. Since the construction of C is polynomial we have derived a polynomial time algorithm for the augmentation problem which was shown to be $\mathcal{NP}$-complete. □

4.2.3 Complexity for fixed row and column permutation

Section 2.4.3 shows that the WC1P is solvable even in linear time for fixed column permutation (the same holds for fixed row and column permutation). Therefore it seems pretty

surprising that in the simultaneous case the WSC1P remains $\mathcal{NP}$-hard even if row and column permutation are fixed. To prove this we first want to generalize definition 2.19 to the simultaneous case.

Definition 4.10 *A 0/1-matrix is in* **simultaneous Petrie form (is SPET)** *if it has the consecutive ones property both for rows and for columns without any rearrangements of rows or columns.*

Example 4.11

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Both matrices are C1PS *but only the right one is* SPET.

Also in this case we define an augmentation version. Note that in this definition both kinds of replacements are allowed.

Definition 4.12 *A 0/1-matrix M has the* **k-augmented simultaneous Petrie form (is SPET$_k$)** *if and only if there exists a matrix M' which is* SPET *and which arises from M by switching at most k entries of M from "0" to "1" or vice versa.*

Example 4.13

$$\begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \end{pmatrix} \qquad \begin{pmatrix} 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

The first matrix is SPET$_4$. *Switching 4 entries leads to the second one which is* SPET.

The proof of the $\mathcal{NP}$-completeness of the corresponding decision problem is a little bit more complicated than the the proof of theorem 4.7. Our goal is to construct a transformation from the directed Hamiltonian path problem and to do this we need some preparations.

Definition 4.14 *For a given directed graph $G = (V, A)$ we define the hyperincidencematrix $H_G = (h_{ij}), (1 \leq i, j \leq |V|)$ as*

$$h_{ij} = \begin{cases} S & : \quad i = j \\ E & : \quad i \neq j \text{ and } (ij) \in A \\ O & : \quad i \neq j \text{ and } (ij) \notin A, \end{cases}$$

where S, E and O are 0/1-matrices defined as follows

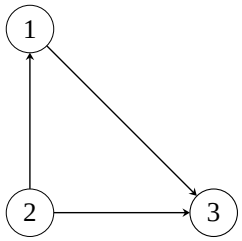
$$S = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$E = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$O = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Furthermore the binary matrix $\overline{H}_G$ exists of $|V|-1$ identical copies of H_G written horizontally.

According to this construction both H_G and $\overline{H}_G$ are 0/1-matrices with $3|V|$ rows. H_G has $7|V|$ and $\overline{H}_G$ contains $7|V|(|V|-1)$ columns. Figure 4.3 shows a directed graph with three nodes and the corresponding matrices H_G and $\overline{H}_G$.



$$H_G = \begin{pmatrix} S & E & O \\ O & S & O \\ E & E & S \end{pmatrix} =$$

$$\left(\begin{array}{ccccccc|ccccccc|ccccccc} 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 1&1&1&0&0&1&1&1&1&0&0&1&1&1&1&1&0&1&0&1&1 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ \hline 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 1&1&0&1&0&1&1&1&1&1&0&0&1&1&1&1&0&1&0&1&1 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ \hline 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \\ 1&1&0&0&1&1&1&1&1&0&0&1&1&1&1&1&1&0&0&1&1 \\ 0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0 \end{array}\right)$$

$$\overline{H}_G = \left(\begin{array}{ccc|ccc} S & E & O & S & E & O \\ O & S & O & O & S & O \\ E & E & S & E & E & S \end{array}\right)$$

Figure 4.3: A directed graph $G = (V, A)$ and the corresponding matrices H_G and $\overline{H}_G$.

Although the claims of the following two lemmata seem to be clear their proof is rather technical.

Lemma 4.15 *Given a directed graph $G = (V, A)$ with $|V| \geq 2$. Let $\overline{H}_G$ be* SPET_k *and $\overline{H}'_G$ be a corresponding* 0/1*-matrix that is* SPET *and differs from $\overline{H}_G$ exactly by k entries. If there is a column of $\overline{H}'_G$ containing at least 2 "1"-entries then $\overline{H}_G$ is* SPET_{k-1}.

Proof. We assume that $\overline{H}_G = (\overline{h}_{ij})$ and $\overline{H}'_G = (\overline{h}'_{ij})$ with $1 \leq i \leq 3|V|$ and $1 \leq j \leq 7|V|(|V|-1)$. Scanning $\overline{H}'_G$ from left to right let j be the first column that contains at least 2 "1"-entries. Furthermore let j' be the first column with the property that there exists a row i' with $i' \not\equiv 2 \pmod 3$ (that means that row i' of $\overline{H}_G$ consists of "0"-entries only) and $\overline{h}'_{i'j'} = 1$. Now we distinguish to cases.

First let $j' < j$. In this case $\overline{H}'_G$ remains SPET if the entry $\overline{h}'_{i'j'}$ changes to "0". The consecutive ones property in column j' is still fulfilled because from $j' < j$ it follows that $\overline{H}'_G$ has only a single "1"-entry in this column. Furthermore the "1"-entries of row i' remain occurring consecutively since column j' corresponds to the first "1"-entry of $\overline{H}'_G$ in this row. Therefore we have constructed a SPET matrix requiring only $k-1$ changes starting from $\overline{H}_G$ and with it $\overline{H}_G$ is SPET_{k-1}.

Now let $j' \geq j$. We have a look at the series of "1"s in column j of $\overline{H}'_G$, especially to the first and last "1"-entry in that column. Let i_1 and i_2 ($i_1 < i_2$) be the rows corresponding to these entries. If $i_1 \not\equiv 2 \pmod 3$ or $i_2 \not\equiv 2 \pmod 3$ holds we are ready, since in this case switching the entry with this property back to "0" both preserves SPET (because of $j' \geq j$) and reduces the total number of changes to $k-1$. Otherwise we have at least 4 "1"-entries in column j since both $i_1 \equiv 2 \pmod 3$ and $i_2 \equiv 2 \pmod 3$ holds. Because of $j' \geq j$ at least one of the 2 "1"-entries $\overline{h}'_{i_1 j}$ and $\overline{h}'_{i_2 j}$ is the first "1"-entry in its row. Let i denote this row. Without loss of generalization we assume $i = i_1$. Because of the appearance of $\overline{H}_G$ we get $\overline{h}_{(i+1)j} = 0$ and $\overline{h}_{(i+2)j} = 0$. And due to the definition of j' and the fact that $j' \geq j$ also $\overline{h}'_{(i+1)k} = \overline{h}'_{(i+2)k} = 0$, for all $k < j$, $\overline{h}'_{(i+1)j} = 1$ and $\overline{h}'_{(i+2)j} = 1$ follows. Consequently switching both $\overline{h}'_{ij}$ and $\overline{h}'_{(i+1)j}$ and $\overline{h}'_{(i+2)j}$ to "0" preserves SPET, increases the number of changes by 1 but decreases it by 2. Therefore $\overline{H}_G$ is SPET_{k-1}. □

Lemma 4.16 *Given a directed graph $G = (V, A)$ with $|V| \geq 2$. Let $\overline{H}_G$ be* SPET_k *and $\overline{H}'_G$ be a corresponding* 0/1*-matrix that is* SPET *and differs from $\overline{H}_G$ exactly by k entries. If there is a row $i \not\equiv 2 \pmod 3$ with the property that $\overline{H}'_G$ contains a "1"-entry in this row then $\overline{H}_G$ is* SPET_{k-1}.

Proof. If the conditions of lemma 4.15 hold we are ready. Therefore we assume that $\overline{H}'_G$ contains at most 1 nonzero entry in each column. Now let $\overline{h}'_{ij}$ the first "1"-entry in row i after scanning from the left to the right. If we switch this entry to "0" we both preserve SPET and reduce the number of changes by 1 to $k-1$. □

The previous two lemmata show how "optimal" matrices $\overline{H}'_G$ do not look like. In this sense "optimal" means, for a given graph G we are looking for a minimal k such that $\overline{H}_G$ is SPET_k. Now we prove a lower bound on this k.

Lemma 4.17 *Given a directed graph $G = (V, A)$ with $|V| \geq 2$. If $\overline{H}_G$ is* SPET_k *then $k \geq 5|V|^3 - 8|V|^2 + 3$ follows.*

Proof. Again let $\overline{H}'_G$ be a 0/1-matrix that is SPET and differing from $\overline{H}_G$ exactly by k entries. Taking the lemmata 4.15 and 4.16 into account it is sufficient to show the claim for the case that $\overline{H}'_G$ has at most 1 nonzero entry in each column and only nonzero entries in rows i with $i \equiv 2 \pmod 3$. Now we partition $\overline{H}'_G$ into the $|V|(|V|-1)$ blocks (numbered from $1 \ldots |V|(|V|-1)$) each consisting of 7 columns. Now for each block we determine a lower bound on the number of entries differing between $\overline{H}_G$ and $\overline{H}'_G$. For a given block i let r_i be the number of rows where $\overline{H}'_G$ has at least one nonzero entry in this block. If $r_i = 0$ then all $5|V|$ "1"-entries of $\overline{H}_G$ have to be switched to "0". If $r_i = 1$ then in that row at least two entries and in all the other rows 5 entries have to be switched, which makes a total of $5|V|-3$. If $r_i = 2$ then in both rows we need at least two switched entries. This works only for the case that the first row corresponds to an S-block and the second to an E-block. All the other rows need 5 entries to be switched and thus $5|V|-6$ changes altogether. For $r_i \geq 3$ we argue as follows. The total number of "1"-entries in block i of $\overline{H}'_G$ is at most 7 according to the number of columns. The number of "1"-entries of $\overline{H}_G$ is $5|V|$. Thus we need at least $5|V|-7$ changes from $\overline{H}_G$ to $\overline{H}'_G$.

Let $r_i \geq 1$ then $r_i - 1$ counts the number of transitions from one row to another. Altogether there are at most $|V|-1$ transitions since there are at most $|V|$ rows where "1"-entries can occur. Therefore we have

$$\sum_{i=1}^{|V|(|V|-1)} (r_i - 1) \leq |V| - 1.$$

Now computing the minimum number of switches needed leads to solving a simple knapsack problem. The optimum solution is to take $|V|-1$ times $r_i = 2$ and $(|V|-1)^2$ times $r_i = 1$ which leads to a minimum total number of

$$(|V|-1)(5|V|-6) + (|V|-1)^2(5|V|-3) = 5|V|^3 - 8|V|^2 + 3$$

switched entries. □

Theorem 4.18 *The directed graph $G = (V, A)$ contains a Hamiltonian path if and only if $\overline{H}_G$ is* $\mathrm{SPET}_{5|V|^3-8|V|^2+3}$.

Proof. Assume G containing a Hamiltonian path consisting of the $|V|-1$ edges $e_i = (t_i, h_i)$ $(1 \leq i \leq |V|-1)$. $\overline{H}_G$ consists of $|V|-1$ identical copies of H_G. And according to its construction H_G contains a column with an S-block in row t and an E-block in row h if and only if there is an edge from t to h in the graph G. Now we construct the matrix $\overline{H}'_G$ from the left to the right by starting with 1 entries in row $3t_1 + 2$ and making a transition from row $3t_i + 2$ to row $3h_i + 2$ in the i-th copy of H_G corresponding to a transition from an S-block to an E-block. As in the proof of the previous lemma we have exactly $|V|-1$ transitions from an S-block to an E-block and this can be managed by a total number of $5|V|^3 - 8|V|^2 + 3$ switched entries. Figure 4.4 shows an example for this construction. It models a Hamiltonian path in the graph of figure 4.3. In each of the 4 "dotted" blocks

$$\left(\begin{array}{ccc|ccccccc|ccc|ccccccc|ccc|ccc}
0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\
0 & \cdots & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & \cdots & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\
0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\
\hline
0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\
1 & \cdots & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\
0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\
\hline
0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\
0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & \cdots & 1 & 1 & \cdots & 1 \\
0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & \cdots & 0
\end{array}\right)$$

Figure 4.4: This SPET matrix $\overline{H}'_G$ arises from matrix $\overline{H}_G$ of figure 4.3 by 66 switched entries and models the Hamiltonian path from node 2 via node 1 to node 3.

this matrix differs from $\overline{H}_G$ by $5|V| - 3 = 12$ entries. In the other 2 blocks it differs by $5|V| - 6 = 9$ entries which makes a total of $66 = 5|V|^3 - 8|V|^2 + 3$.

Conversely let $\overline{H}_G$ be $\text{SPET}_{5|V|^3-8|V|^2+3}$. Then we know from the proof of the previous lemma that the corresponding solutions must contain $|V|-1$ transitions from an S to an E block of $\overline{H}_G$. Each of these transitions corresponds to an edge in the underlying directed graph G. And since in each row of $\overline{H}'_G$ the "1"s have to occur consecutively no node in the constructed path is allowed to occur twice. Therefore we have constructed a Hamiltonian path in G. □

This equivalence to the directed Hamiltonian path problem immediately leads to the $\mathcal{NP}$-completeness proof.

Theorem 4.19 *Deciding if a* 0/1*-matrix* M *is* SPET_k *is* $\mathcal{NP}$*-complete.*

Proof. Verifying if a given k-augmentation for M is SPET can be done in linear time by scanning each row and each column.

Now assume we are given a polynomial time algorithm deciding if a given 0/1-matrix is SPET_k. Because the transformation from and to the directed Hamiltonian path problem can also be performed in polynomial time we would obtain a polynomial time algorithm for finding a Hamiltonian path in a directed graph which is known to be $\mathcal{NP}$-complete ([GJ79]). □

Analogously to section 4.2.2 a direct corollary is the fact that the corresponding weighted optimization problem is $\mathcal{NP}$-hard.

Corollary 4.20 *The corresponding weighted optimization problem is* $\mathcal{NP}$*-hard.*

Proof. Given a 0/1-matrix A we construct the objective function matrix C as

$$c_{ij} = \begin{cases} 1 & : \quad a_{ij} = 0 \\ -1 & : \quad a_{ij} = 1. \end{cases}$$

Now let l be the number of "1"-entries of A. Then A is SPET_k if and only if the solution of

$$\begin{aligned} &\min C \circ A \\ s.t.\quad &A \text{ is SPET} \end{aligned}$$

is less or equal to $k-l$. The construction of the objective function is polynomial. Therefore the $\mathcal{NP}$-hardness is shown. □

4.2.4 Complexity for fixed number of rows or columns

Analogously to section 3.6 we can introduce the concept of feasible and maximal sets to the simultaneous case. For symmetrical reasons it is sufficient to define sets of columns.

Definition 4.21 *A set $C = \{c_1, \ldots, c_k\}$ where $c_i \in \{0,1\}^{(m,1)}$, for $i = 1, \ldots, k$, of columns is called* **consecutive ones simultaneous feasible (C1SF)** *if the matrix consisting of all columns of C is* C1PS.

Definition 4.22 *A C1SF set C is called* **consecutive ones simultaneous maximal (C1SM)** *if C is maximal with respect to set inclusion.*

Example 4.23 *For $m = 3$ all C1FC sets are also C1SF. This is due to the fact that non of the C1FC sets includes all the 3 columns each containing 2 "1"s. Therefore also* C1PC *can be established (compare example 3.27). Now let $m = 4$. The set*

$$C = \left\{ \begin{pmatrix} 0\\0\\0\\0 \end{pmatrix}, \begin{pmatrix} 1\\0\\0\\0 \end{pmatrix}, \begin{pmatrix} 1\\1\\1\\0 \end{pmatrix}, \begin{pmatrix} 0\\1\\1\\0 \end{pmatrix}, \begin{pmatrix} 0\\1\\1\\1 \end{pmatrix}, \begin{pmatrix} 0\\0\\0\\1 \end{pmatrix} \right\}$$

is C1SF but not C1SM since one can add the column $(1\ 1\ 0\ 0)^T$ and the resulting set

$$\overline{C} = \left\{ \begin{pmatrix} 0\\0\\0\\0 \end{pmatrix}, \begin{pmatrix} 1\\0\\0\\0 \end{pmatrix}, \begin{pmatrix} 1\\1\\0\\0 \end{pmatrix}, \begin{pmatrix} 1\\1\\1\\0 \end{pmatrix}, \begin{pmatrix} 0\\1\\1\\0 \end{pmatrix}, \begin{pmatrix} 0\\1\\1\\1 \end{pmatrix}, \begin{pmatrix} 0\\0\\0\\1 \end{pmatrix} \right\}$$

is still C1SF. But any further column would lead to a set whose corresponding matrix is not C1PS. *Therefore $\overline{C}$ is C1SM. Note that $\overline{C}$ cannot be C1MC. According to lemma 3.28 a C1MC set always contains exactly $2m = 8$ elements. Indeed, in the nonsimultaneous case (and only in this) one can add the column $(0\ 1\ 0\ 1)^T$.*

The previous example gives an idea that in contrast to C1MC and C1MR sets C1SM sets do not have the same cardinality (for given m). Therefore they do not define a matroid (compare to remark 3.34). But for the purpose of determining the complexity of WSC1P for a fixed number of columns (rows resp.) we only need the fact that the number of C1SM sets only depends on the number of rows (columns resp.). As in section 3.6 we will show

that in this case even the associated polytope has only a polynomial number of facets. First we should define this polytope.

Definition 4.24 *The* **simultaneous consecutive ones polytope** *is defined as*

$$\boldsymbol{P_{\mathrm{C1S}}^{m,n}} = \mathrm{conv}\{M \mid M \text{ is an } (m,n)\text{-matrix with C1PS}\}.$$

As in the nonsimultaneous case $P_{\mathrm{C1S}}^{m,n}$ is full-dimensional.

Theorem 4.25 *$P_{\mathrm{C1S}}^{m,n}$ has dimension $m \cdot n$.*

Proof. As in the proof of theorem 3.1 the zero matrix and the matrices consisting of zeros only except for a single "1"-entry give a set of $m \cdot n + 1$ affinely independent C1PS matrices. □

As in the standard case trivial inequalities define facets and all facet-defining inequalities can be trivially lifted to polytopes of higher dimensions. The proofs are given in section 4.3.

Proving that for a fixed number m of columns the number of facets of $P_{\mathrm{C1S}}^{m,n}$ grows only polynomially in the number n of rows follows exactly the same lines as the proof of theorem 3.35. As mentioned above we only need the fact that the number of C1SM sets only depends on m which is a constant. A very rough upper bound for this number is the number of elements of the according powerset which is $2^{(2^m)}$.

Theorem 4.26 *The number of facets of $P_{\mathrm{C1S}}^{m,n}$ is polynomial in m if n is fixed and vice versa.*

Proof. See proof of 3.35. □

And again as a consequence we obtain the polynomial solvability of WSC1P for fixed number of rows or columns.

Corollary 4.27 WSC1P *is solvable in polynomial time for fixed n or fixed m.*

Proof. See proof of 3.37. □

4.3 The facial structure of $P_{\mathrm{C1S}}^{m,n}$

We want to conclude this chapter with some polyhedral investigations of the simultaneous polytope. As already mentioned, facet-defining inequalities can be lifted trivially to higher dimensions. The proof is different to the standard case and for technical reasons we will do it only for facets of $P_{\mathrm{C1S}}^{m,n}$ with $m \geq 3$ and $n \geq 3$. This is no real restriction, since for $m \leq 2$ or $n \leq 2$ all matrices are C1PS. Therefore these polytopes only consist of trivial inequalities that will be shown to be facet-defining separately.

Theorem 4.28 *Let $A \circ x \leq a_0$ be a facet-defining inequality for $P_{\mathrm{C1S}}^{m,n}$ and let $m' \geq m \geq 3$ and $n' \geq n \geq 3$. If the inequality $\overline{A} \circ x \leq a_0$ for $P_{\mathrm{C1S}}^{m',n'}$ is obtained from $A \circ x \leq a_0$ by trivial lifting then it defines a facet of $P_{\mathrm{C1S}}^{m',n'}$.*

Proof. Because of the symmetry it is sufficient to show the case $m' > m$ and $n' = n$, where without loss of generality $m' = m + 1$.

For all matrices $\overline{x}$ satisfying $A \circ \overline{x} = a_0$ we do the following construction. We add one row to $\overline{x}$ by duplicating an arbitrary row. Let $\overline{x}^j$ denote the matrix obtained by duplicating row j of $\overline{x}$ $(1 \leq j \leq m)$. Clearly all these matrices satisfy $\overline{A} \circ \overline{x} = a_0$. Now assume $\overline{A} \circ x \leq a_0$ not to be facet-defining for $P^{m',n'}_{\text{C1S}}$. Then there must exist an equation $B \circ x = b_0$ that is fulfilled by all matrices constructed above with the property that at least one coefficient of the last row of B is nonzero. Therefore we have

$$B \circ \overline{x}^1 = \ldots = B \circ \overline{x}^m = b_0$$

and since the matrices $\overline{x}^j$ only differ in their last row we get

$$B_{m'.} \circ \overline{x}^1_{m'.} = \ldots = B_{m'.} \circ \overline{x}^m_{m'.}.$$

But due to their construction we have $\overline{x}^j_{m'.} = \overline{x}_{j.}$ and with it

$$B_{m'.} \circ \overline{x}_{1.} = \ldots = B_{m'.} \circ \overline{x}_{m.}.$$

These equations hold for all matrices $\overline{x}$ satisfying $A \circ \overline{x} = a_0$ and since $m \geq 3$ there are at least two equations. Therefore the inequality $A \circ \overline{x} \leq a_0$ cannot be facet-defining for $P^{m,n}_{\text{C1S}}$ which is a contradiction. Consequently $\overline{A} \circ x \leq a_0$ is facet-defining for $P^{m',n'}_{\text{C1S}}$. □

Very similar to the proof that $P^{m,n}_{\text{C1S}}$ has full dimension, we can show that trivial inequalities are always facet-defining.

Lemma 4.29 *For all $m \geq 1$, $n \geq 1$, $1 \leq i \leq m$, $1 \leq j \leq n$, the inequalities $x_{ij} \geq 0$ and $x_{ij} \leq 1$ define facets of $P^{m,n}_{\text{C1S}}$.*

Proof. The zero matrix and the matrices consisting of zeros only except for a single "1"-entry in all possible positions besides ij give a set of $m \cdot n$ affinely independent C1PS matrices satisfying $x_{ij} = 0$. Taking the same set but switching the role of "0"-entries and "1"-entries we get a set of $m \cdot n$ affinely independent C1PS matrices satisfying $x_{ij} = 1$. □

The next question we want to address to is, whether valid respectively facet-defining inequalities of the standard polytope keep their properties for the simultaneous polytope or vice versa. Since $P^{m,n}_{\text{C1S}}$ is contained in $P^{m,n}_{\text{C1R}}$ it is clear that valid inequalities for $P^{m,n}_{\text{C1R}}$ are also valid for $P^{m,n}_{\text{C1S}}$.

Lemma 4.30 *Let $A \circ x \leq a_0$ be a valid inequality for $P^{m,n}_{\text{C1R}}$. Then it is also valid for $P^{m,n}_{\text{C1S}}$.*

This property does not hold for facet-defining inequalities. Consider for example the inequality

$$\begin{pmatrix} 2 & -8 & -5 & 11 & 9 \\ 2 & -1 & 5 & -5 & 5 \\ -6 & 7 & 4 & 11 & -5 \\ 4 & 8 & -4 & -8 & 4 \end{pmatrix} \circ x \leq 60.$$

It is facet-defining for $P^{4,5}_{\text{C1R}}$ and has only 20 roots. If not all of these 20 C1PR-matrices are also C1PS, than the inequality defines no facet for the simultaneous case, since the number of roots of a facet must be greater or equal to the dimension of the corresponding polytope which is $4 \cdot 5 = 20$. Indeed, one of the 20 roots is the matrix

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

Its C1PR-property is already established. But this matrix is not C1PS, since the $(4,3)$ matrix consisting of columns 1, 3 and 5 is equal to the matrix $M^T_{3_1}$ which according to figure 4.1 is one of the forbidden matrices in the simultaneous case. Therefore the above inequality defines a facet for $P^{4,5}_{\text{C1R}}$ but is only valid for $P^{4,5}_{\text{C1S}}$.

And it is the smallest possible example (in terms of $m+n$) with this property. For $m \leq 3$ we have $P^{m,n}_{\text{C1R}} = P^{m,n}_{\text{C1S}}$, for $n \leq 3$ the polytopes $P^{m,n}_{\text{C1R}}$ only consist of trivial facets and facets of type $\mathcal{F}_{1_1}$ (we will see later on that all staircase inequalities remain facet-defining in the simultaneous case) and furthermore appendix A.3 displaying the complete description of $P^{4,4}_{\text{C1S}}$ shows, that all facets of $P^{4,4}_{\text{C1R}}$ appear again in this description.

Besides the transposed versions of the known facets there occur 3 completely new classes of facet-defining inequalities for $P^{4,4}_{\text{C1S}}$. They are shown in figure 4.5.

$$\begin{pmatrix} -1 & -1 & 1 & 1 \\ -1 & 2 & 1 & 1 \\ 2 & -2 & 2 & -1 \\ 2 & 2 & -1 & -1 \end{pmatrix} \circ x \leq 12$$

$$\begin{pmatrix} 2 & 2 & 1 & -1 \\ -2 & -2 & 2 & 1 \\ -2 & 2 & -2 & 2 \\ 2 & -2 & -2 & 2 \end{pmatrix} \circ x \leq 13$$

$$\begin{pmatrix} 2 & 3 & -1 & 2 \\ 3 & -3 & 3 & -2 \\ -2 & 3 & 3 & -1 \\ 1 & -1 & -2 & 2 \end{pmatrix} \circ x \leq 19$$

Figure 4.5: Inequalities that are facet-defining for $P^{4,4}_{\text{C1S}}$ but not even valid for $P^{4,4}_{\text{C1R}}$.

None of them is valid for $P^{4,4}_{\text{C1R}}$. This observation leads to the following lemma.

Lemma 4.31 *Let $A \circ x \leq a_0$ be facet-defining for $P^{m,n}_{\text{C1S}}$. Then it is either nonvalid for $P^{m,n}_{\text{C1R}}$ or facet-defining for $P^{m,n}_{\text{C1R}}$.*

Proof. Assume that $A \circ x \leq a_0$ is valid for $P^{m,n}_{\text{C1R}}$. Then we have to show that it is even facet-defining for $P^{m,n}_{\text{C1R}}$. Since it is facet-defining for $P^{m,n}_{\text{C1S}}$ there is a set of $m \cdot n$ affinely independent C1PS matrices satisfying $A \circ x = a_0$. And since each of these matrices is also C1PR we are ready. □

Example 4.32 *We have a look at the inequality*

$$\begin{pmatrix} 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \circ x \leq 5.$$

According to appendices A.2 and A.3 it is both facet-defining for $P^{3,4}_{\text{C1S}}$ and $P^{3,4}_{\text{C1R}}$. If we take the transposed version of this inequality

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \end{pmatrix} \circ x \leq 5.$$

it remains facet-defining for $P^{4,3}_{\text{C1S}}$ but it is not valid for $P^{4,3}_{\text{C1R}}$ since the feasible C1PR *matrix*

$$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

is cut off by this inequality.

Note that the proofs of the previous two lemmata only require the property that $P^{m,n}_{\text{C1S}}$ is contained in $P^{m,n}_{\text{C1R}}$. The tightened version, that a given facet-defining inequality of $P^{m,n}_{\text{C1R}}$ defines also a facet of $P^{m,n}_{\text{C1S}}$ is more difficult to show and, as we have seen above, not true in any case. We will perform this proof in the following for the large and important classes of staircase inequalities.

4.3.1 Staircase inequalities

Analogously to the standard case we are interested in investigating the staircase inequalities for the purpose of finding an IP formulation for WSC1P consisting of facets only.

In the simultaneous case the staircase inequalities $\mathcal{F}_{1_k} \circ x \leq 2k+3$ $(k \geq 1)$, $\mathcal{F}_{2_k} \circ x \leq 2k+3$ $(1 \leq k \leq 3)$ and $\mathcal{F}^T_{2_k} \circ x \leq 2k+3$ $(1 \leq k \leq 3)$ are even sufficient to cut off all forbidden Tucker matrices. This can be verified easily with the help of figures 3.3 and 4.1. We will now show that all of them and their transposes are facet-defining for $P^{m,n}_{\text{C1S}}$.

Theorem 4.33 *The staircase inequalities*

$$\mathcal{F}_{1_k} \circ x_{IJ} \leq 2k+3$$

and their transposes

$$\mathcal{F}_{1_k}^T \circ x_{IJ} \leq 2k+3,$$

for $k \geq 1$ and all $(k+2, k+2)$-index sets, are facet-defining for $P_{\mathrm{C1S}}^{m,n}$ for all $m \geq k+2$ and $n \geq k+2$.

Proof. Due to the symmetry of $P_{\mathrm{C1S}}^{m,n}$ we do not have to consider the transposed versions of the staircase inequalities separately and due to lifting theorem 4.28, we only need to show that this class of inequalities is facet-defining for $P_{\mathrm{C1S}}^{m,n}$ with $m = n = k+2$. Moreover, we only need to consider the canonical ordered index sets $I = \{1, 2, \ldots, m\}$ and $J = \{1, 2, \ldots, n\}$. Let $a^T x \leq a_0$ denote this inequality and let $F = \{x \mid a^T x = a_0\} \cap P_{\mathrm{C1S}}^{m,n}$ denote the induced face.

The validity of the inequality follows from its validity for the nonsimultaneous case (see theorem 3.8) and lemma 4.30.

Since $P_{\mathrm{C1S}}^{m,n}$ is full-dimensional, facet-defining inequalities defining the same facet only differ by multiplication with a positive scalar. Now let $b^T x \leq b_0$ be a facet-defining inequality for $P_{\mathrm{C1S}}^{m,n}$ such that $F \subseteq \{x \mid b^T x = b_0\} \cap P_{\mathrm{C1S}}^{m,n}$. If we can show that $b = \lambda a$, for some $\lambda > 0$, then it is proven that $a^T x \leq a_0$ is facet-defining. We will show this in three steps. Let $\beta = b_{11}$. We call C1PS matrices x that satisfy $a^T x = a_0$ *solutions*.

Every 0/1-matrix with $2k+3$ 1's in the "+" positions of $\mathcal{F}_{1_k}$ and 0's otherwise is a solution. We call such matrices *standard solutions* in the following. Let x^1 and x^2 be two standard solutions where $x^1_{11} = 0$ and $x^2_{21} = 0$. Then we have

$$0 = b^T x^1 - b^T x^2 = b_{21} - b_{11}$$

and therefore $b_{21} = b_{11} := \beta$. By using appropriate matrices we can thus show that $b_{ij} = \beta$ for all "+"-positions ij.

Consider the standard solution x^1 with $x_{1,k+2} = 0$. Let x^2 be a matrix which is identical to x^1 except for an additional 1 in a "0"-position next to an "+"-position in any row i and next to an "+"-position in any column $j \leq k+1$. Then x^2 is a solution and we obtain

$$0 = b^T x^1 - b^T x^2 = -b_{ij},$$

i.e., $b_{ij} = 0$. Extending the chain of 1's to the next "0"-positions eventually shows that $b_{ij} = 0$ for all "0"-positions ij of $\mathcal{F}_{1_k}$.

Finally let x^1 be any standard solution, say the one with $x_{11} = 0$. Further, after exchanging the last two columns of $\mathcal{F}_{1_k}$ and inserting row 1 before row $k+1$ we obtain the following matrix.

$$\left(\begin{array}{cccccc}
+ & + & & & & - \\
 & \ddots & \ddots & & & \vdots \\
 & & + & + & & - \\
+ & & & & + & - \\
 & & & + & - & + \\
- & & & & + & +
\end{array}\right)$$

Now let x^2 be the C1PS matrix constructed by setting all entries inside the boundary to 1 and to 0 outside. Since there is only one "$-$"-entry inside and no "$+$"-entry outside the boundary, x^2 is a solution. Note that the "$-$"-entry inside originally belongs to row $k+1$ and column $k+2$. Since the coefficients corresponding to all "0"-entries are already proven to be 0 we receive

$$0 = b^T x^1 - b^T x^2 = -b_{11} - b_{k+1,k+2},$$

and therefore $b_{k+1,k+2} = -\beta$. Using similar arguments we eventually obtain $b_{ij} = -\beta$ for all "$-$"-positions ij.

Thus we have shown that $b = \beta a$. It is clear that $\beta > 0$ since if we change a "1"-entry in a standard solution to "0" then we obtain another C1PS matrix which would violate the inequality if $\beta < 0$. □

Theorem 4.34 *The staircase inequalities*

$$\mathcal{F}_{2_k} \circ x_{IJ} \leq 2k + 3$$

respectively their transposes

$$\mathcal{F}_{2_k}^T \circ x_{IJ} \leq 2k + 3,$$

for $k \geq 1$ and all $(k+2, k+3)$-index sets, resp. all $(k+3, k+2)$-index sets, are facet-defining for $P_{\mathrm{C1S}}^{m,n}$, resp. for $P_{\mathrm{C1S}}^{n,m}$, for all $m \geq k+2$ and $n \geq k+3$.

Proof. The proof is very similar to the previous one. It is sufficient to consider the standard form of this inequality and to set $m = k+2$, $n = k+3$, $I = \{1, 2, \ldots, m\}$ and $J = \{1, 2, \ldots, n\}$. Furthermore the validity follows from theorem 3.10 and lemma 4.30.

Now let $b^T x \leq b_0$ defined as before. Again we want to compute the coefficients of b to show that b is a multiple of $\mathcal{F}_{2_k}$. Starting with the standard solution (defined as before) where $x_{k+2,k+3} = 0$ we can construct a chain of C1PS matrices to the matrix where in addition $x_{ij} = 1$ for all "0"-entries ij. As in the previous proof we can construct this chain in such a way that two consecutive matrices only differ in one of these entries showing that $b_{ij} = 0$ for all "0"-entries ij.

Now we have to show that for all "$+$"- and all "$-$"-positions ij of $\mathcal{F}_{2_k}$ there is a solution with 1's at the "$+$"-positions and 0's at the "$-$"-positions and x_{ij} being the only exception (we call these solutions (ij)-solutions in the following). For this purpose the following picture is helpful. It shows the matrix $\mathcal{F}_{2_k}$ where row $k+2$ is inserted after row 1 and column $k+3$ after column 1.

$$\begin{pmatrix}
+ & - & + & & & & - \\
- & + & & & & + & - \\
 & - & + & + & & & - \\
 & \vdots & & \ddots & \ddots & & \vdots \\
 & - & & & + & + & - \\
- & - & & & & + & +
\end{pmatrix}$$

The matrix consisting of 1's inside and 0's outside the boundary is a (11)-solution. Extending the boundary in the picture to the entries 11 and 12 we get a $(1, k{+}3)$-solution (note that column $k{+}3$ was mapped to column 2). Changing the matrix according to $x_{11} = 1$ and $x_{1j} = 0$ for $2 \leq j \leq k+3$ gives a (12)-solution (column 2 was mapped to column 3).

Furthermore if we take $\mathcal{F}_{2_k}$ and insert row $k{+}1$ after row 1 and column $k{+}2$ after column 1 we get the same matrix as shown above. Therefore we can construct a $(1, k{+}2)$-solution with the same procedure. Thus we have constructed $(1j)$-solutions for all nonzero entries of the first row of $\mathcal{F}_{2_k}$.

After permuting $\mathcal{F}_{2_k}$ in different ways, for example inserting row $k{+}2$ ($k{+}1$ resp.) after row i and column $k{+}3$ ($k{+}2$ resp.) after column i we eventually can construct (ij)-solutions for all nonzero entries ij of $\mathcal{F}_{2_k}$.

The remainder of the proof follows along the same lines as for $\mathcal{F}_{1_k}$. □

The fact that all these staircase inequalities define facets for $P^{n,m}_{\text{C1S}}$ and the nontransposed versions also for $P^{n,m}_{\text{C1R}}$ as well as the close relationship to the forbidden Tucker matrices make them a good choice for separating procedures. Indeed, we will see in 5.3.1 that both kinds of staircase inequalities can be separated in polynomial time.

Now we are able to formulate an IP formulation with facets.

4.3.2 IP formulation with facets

As in the standard case we can obtain an integer programming formulation for the WSC1P by cutting all forbidden matrices occurring in figure 4.1. Since the staircase inequalities define facets also in the simultaneous case we can derive an IP formulation for the problem that consists only of facets of $P^{m,n}_{\text{C1S}}$. The inequalities $\mathcal{F}_{1_k} \circ x \leq 2k+3$ cut off the Tucker matrices $M_{1_k}(k \geq 1)$, M_{2_1},$M^T_{2_1}$,M_{2_2} and $M^T_{2_2}$. Note that the matrices M_{1_k} are symmetric. Therefore no transposed versions of the inequalities are required. Furthermore the inequalities $\mathcal{F}_{2_k} \circ x \leq 2k+3$ cut off M_{3_k} and $\mathcal{F}^T_{2_k} \circ x \leq 2k+3$ cut off $M^T_{3_k}$ for $1 \leq k \leq 3$. Summarizing these facts we obtain the following IP formulation of WSC1P.

$$
\begin{array}{rll}
 & \min\ c^T x & \\
\mathcal{F}_{1_k} \circ x_{IJ} & \leq 2k+3 & \text{for all } (k+2, k+2)\text{-tuples } (I, J),\ k \geq 1, \\
\mathcal{F}_{2_k} \circ x_{IJ} & \leq 2k+3 & \text{for all } (k+2, k+3)\text{-tuples } I, J),\ 1 \leq k \leq 3, \\
\mathcal{F}^T_{2_k} \circ x_{IJ} & \leq 2k+3 & \text{for all } (k+3, k+2)\text{-tuples } I, J),\ 1 \leq k \leq 3, \\
x_{ij} & \in \{0,1\} & \text{for all } i = 1, \ldots, m,\ j = 1, \ldots, n.
\end{array}
$$

As in the standard case (see section 3.4) the number of inequalities is exponential but they can be separated in polynomial time in the number of rows and columns of the input matrix.

Chapter 5

A Branch-and-Cut Approach

This chapter shows how to use the theoretical knowledge developed in the previous chapters to construct an efficient branch-and-cut code for the WC1P and the WSC1P. Note that, if not stated otherwise, all algorithms described in the following can be used both in the standard and in the simultaneous case. Computational results of our branch-and-cut implementation will be given in chapter 7.

5.1 Feasibility test

Usually, in the first phases of the cutting plane procedure the solution x^* of the LP relaxation is integral but infeasible. As already introduced in section 2.3, feasibility can be tested by the PQ-tree algorithm in linear time. In the simultaneous case the algorithm has to be performed both for the actual integral matrix and for its transposes.

If x^* is feasible, the PQ-tree algorithm also generates all permutations π^* that establish the consecutive ones property of x^*. Therefore with finding all optimal matrices one also obtains all optimal permutations.

The other way around, if x^* is not feasible (but binary), we would be interested in constructing a cutting plane which cuts off the point x^* but is satisfied by all 0/1-vectors different from x^*. Given the efficient PQ-tree algorithm we would not even need an IP formulation of the problem to implement a branch-and-cut approach.

5.1.1 Integer vector separation

We only have to make sure that all integral LP but infeasible solutions are cut off by a valid cutting plane. Let $P = \{i \mid x_i^* = 1\}$ and $Z = \{i \mid x_i^* = 0\}$, then

$$\sum_{i \in P} x_i - \sum_{i \in Z} x_i \leq |P| - 1$$

obviously is a cutting plane with the desired properties.

Usually this cutting plane will be a very weak one, but it can be strengthened by removing rows and columns of the LP relaxation as long as the remaining matrix remains infeasible. This can be done in the time $O((m+n)mn)$.

A heuristic version of this separation idea can even be performed if the LP relaxation x^* is fractional. First we have to make it integral. One possibility is to round entry x^*_{ij} to 1 with probability x^*_{ij} and to 0 otherwise. Now for each row and for each column the total sum of differences between the LP values and the rounded values is calculated. Rows and columns with high value of total rounding are chosen earlier to be removed from the matrix. Again this is repeated as long as the remainder stays infeasible. And if the total amount of rounding in the remaining matrix is less than 1 we can construct a cutting plane.

The same method of shrinking the LP relaxation is also used in subsection 5.3.4 where it is explained in more detail.

5.2 Heuristics

Section 2.4.3 shows how we can solve WC1P in linear time if the column permutation is fixed (actually we have to solve a WPEP). Therefore it is sufficient to have heuristics for a good choice of the column permutation. There are different ideas for this purpose.

5.2.1 Hamming distance heuristic

One idea is to compute a Hamming distance Hamiltonian path directly on the input matrix B. The **Hamming distance** of two columns c_1 and c_2 of B is defined as

$$hd(c_1, c_2) = \sum_{i=1}^{m} |a_{ic_1} - a_{ic_2}|.$$

Now we are looking for an ordering $c_{\pi_1}, c_{\pi_2}, \ldots, c_{\pi_n}$ of the columns such that

$$\sum_{j=1}^{n-1} hd(c_{\pi_j}, c_{\pi_{j+1}})$$

is minimized. Since in a C1PR matrix usually two consecutive columns do not differ much, we can hope to obtain an ordering in which the permuted matrix is nearly C1PR and therefore not many entries have to be switched. Obviously this problem is equivalent to finding a Hamiltonian path of minimal weight in a complete undirected graph where the columns correspond to the nodes and where the edge connecting columns c_1 and c_2 has weight $hd(c_1, c_2)$. By adding one additional node this problem can be transformed to a TSP instance. After fixing the column permutation of the cost matrix C according to the Hamiltonian path found, the obtained WPEP can be solved by algorithm 2.21.

Algorithm 5.1 (`hammingDistanceHeuristic(input matrix` B`, cost matrix` C`)`)

(1) For each pair of columns c_1 *and* c_2 *of* B *compute the Hamming distance* $hd(c_1, c_2)$

(2) Compute a minimum Hamiltonian path by solving a TSP

(3) Permute the columns of C *according to the optimal ordering*

(4) Use algorithm 2.21 for each row of C *to solve the* WPEP

(5) Return the WPEP *solution*

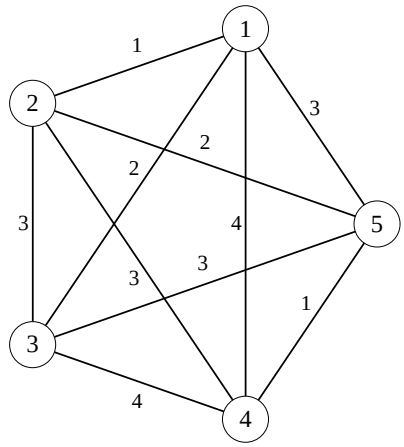
The following example shows a run of this heuristic.

Example 5.2 *Let the input matrix* B *and the cost matrix be* C *given as follows:*

$$B = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix} \qquad C = \begin{pmatrix} 2 & 3 & 1 & -2 & -4 \\ -3 & -1 & 4 & 2 & 2 \\ 2 & 4 & -2 & 1 & 1 \\ -1 & -3 & -1 & 3 & -2 \\ -1 & 1 & -3 & 1 & 4 \end{pmatrix}$$

Note that, according to 2.4.2, if $b_{ij} = 1$*, then* c_{ij} *denotes the penalty for switching entry* b_{ij}*, otherwise the penalty is* $-c_{ij}$*. Therefore all costs are positive in this example.*

After computing the Hamming distances between the columns of B*, we obtain the following weighted graph.*



It is easy to verify that (31254) *and its reverse are the Hamiltonian paths with minimum total weight* 6*. If we permute* C *due to this column permutation we obtain*

$$C' = \begin{pmatrix} 1 & 2 & 3 & -4 & -2 \\ 4 & -3 & -1 & 2 & 2 \\ -2 & 2 & 4 & 1 & 1 \\ -1 & -1 & -3 & -2 & 3 \\ -3 & -1 & 1 & 4 & 1 \end{pmatrix}.$$

Performing the scan-line algorithm 2.21 for all rows of C' we get the permuted solution

$$A' = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

And after retransforming we see that the heuristic solution A differs from B only in 2 entries and has a total switching cost $C \circ A - C \circ B$ of 4.

5.2.2 Rounding heuristic

The Hamming distance heuristic described above does not use any information of the current LP solution. But because we expect the LP relaxation somehow to reflect the structure of the problem, we are very interested in using the relaxation to obtain better column permutations. It seems reasonable to use the PQ-tree algorithm for this purpose. Firstly this algorithm is very efficient and secondly it terminates with a set of permutations.

The only problem is that the PQ-tree algorithm needs a binary matrix as input. As already mentioned before, we need to round the entries of the LP solution. Again it turned out to be a good choice to set an entry x^*_{ij} with probability x^*_{ij} to 1 and to 0 otherwise.

Now we can perform the PQ-tree algorithm on the rounded LP matrix $\overline{x}$ by adding the rows of $\overline{x}$ in an arbitrary way. The reduction procedure is performed as long as either all rows have been added or the algorithm terminates with the null tree, which corresponds to the empty set of permutations. In the first case we take permutations compatible to the last PQ-tree in the second case we take them from the last but one.

Note that we are free in the choice of the order of the added rows. There is a heuristic argument that says that rows where few rounding has been performed should be added earlier than rows with much rounding. The reason is that we assume that the less rounding is performed for a row the better the structure of an optimal solution is reflected by that row.

The complete algorithm looks as follows:

Algorithm 5.3 (`roundingHeuristic(lpSolution` x^*`, cost matrix` C`)`)

(1) Round the matrix x^* to a matrix $\overline{x}$ by setting an entry $\overline{x}_{ij}$ to 1 with probability x^*_{ij} and to 0 otherwise

(2) For each row i compute a measure of rounding $m(i) = \sum_j |\overline{x}_{ij} - x^*_{ij}|$

(3) Create a PQ-tree and reduce it by the rows of $\overline{x}$ in increasing order of $m(i)$

(4) Take the last PQ-tree with a nonempty set S of column permutations

(5) For each $\pi \in S$ (or $\pi \in S' \subset S$, if $|S|$ is too large) do:

(5.1) Permute the columns of C according to π

(5.2) Use algorithm 2.21 for each row of C to solve the WPEP

(6) Return the best WPEP *solution found*

Also this algorithm is illustrated by an example.

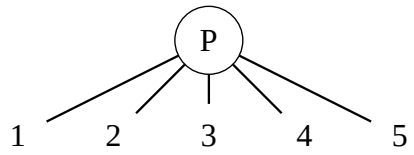
Example 5.4 *The cost matrix is chosen as in example 5.2, the LP relaxation looks as follows:*

$$x^* = \begin{pmatrix} 0.7 & 0.4 & 1.0 & 0.0 & 0.2 \\ 0.0 & 0.0 & 1.0 & 1.0 & 1.0 \\ 0.8 & 1.0 & 0.5 & 1.0 & 1.0 \\ 0.0 & 0.6 & 0.0 & 1.0 & 0.7 \\ 0.0 & 0.9 & 0.0 & 1.0 & 1.0 \end{pmatrix}$$

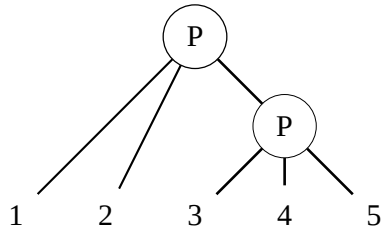
After rounding we get

$$\overline{x} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

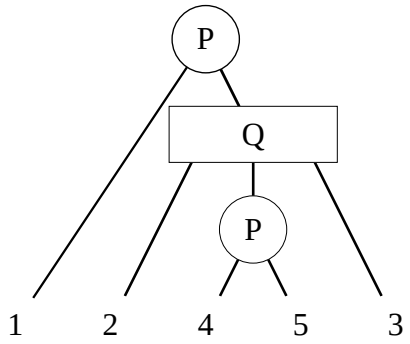
The computed measures of rounding are $m(1) = 1.1$, $m(2) = 0$, $m(3) = 0.7$, $m(4) = 0.9$ and $m(5) = 0.1$ for the 5 rows. Now we perform the PQ-tree algorithm by adding the rows of $\overline{x}$ in increasing order of $m(.)$, that is $2, 5, 3, 4, 1$. We start with the universal tree.



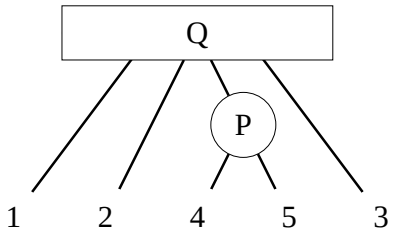
After reducing with row 2, which corresponds to the column set $S_1 = \{3, 4, 5\}$ we obtain the following tree.



The next reducing column set $S_2 = \{2, 4, 5\}$, coming from row 5, gives



Now we have to take row 3 *and reduce with respect to* $S_3 = \{1, 2, 4, 5\}$.



Reducing with respect to $S_4 = \{4, 5\}$*, the* 1*-entries of row* 4*, does not change the current PQ-tree, since the columns* 4 *and* 5 *already are forced to occur consecutively.*

The last reducing step with respect to $S_5 = \{1, 2, 3\}$ *results in the null tree. Therefore we have to extract all represented permutations from the previous tree. These are* $\pi_1 = (12453)$*,* $\pi_2 = (12543)$ *and their reverses. With the two permutations we proceed as in example 5.2. Taking the reverse permutations gives no additional solution, since the scan-line algorithm has the same output for a permutation and its reverse. Both orderings* π_1 *and* π_2 *lead to the following solution:*

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

The switching cost from B *to this* C1PR *matrix is* 1*. For this solution value it is easy to prove optimality, since* B *is not* C1PR *and all switching penalties are at least* 1*.*

This rounding heuristic works very well. We will see in sections 7.1 and 7.2 that in most times the optimal solution is found long before the branch-and-cut procedure proves its optimality.

However, both heuristic ideas described before are not a good choice for the simultaneous problem. This is due to the fact that in the simultaneous case the weighted problem is

already $\mathcal{NP}$-hard for fixed row and column permutation (see section 4.2.3). We modified the rounding heuristic in such a way that after obtaining a C1PR solution the matrix is scanned row by row and made C1PS by switching certain entries. But in section 7.4 we will see that the result were not as good as in the standard case. In addition we implemented a simulated annealing procedure. It starts with any feasible solution, for example the matrix consisting of ones only. After that it goes from one feasible solution to another by changing only one entry. If the new matrix is also feasible and the objective function has at least the value of the previous solution, the change is accepted. If the new matrix is feasible but its objective function is worse, then it is accepted only with a certain probability. This probability decreases with an increasing difference of the values and an increasing current running time of the algorithm. This is the usual procedure of a simulated annealing algorithm. A disadvantage of this algorithm is its running time for big instances, say more than 10000 entries of the matrix, since in each step two PQ-tree tests have to be performed, both for C1PR and for C1PC.

5.3 Separation procedures

Chapter 3 and section 4.3 provided us with more insight into the structure of $P^{m,n}_{\text{C1R}}$ and $P^{m,n}_{\text{C1S}}$. Now we are interested in using these informations in a practical way, namely to improve our branch-and-cut approach by constructing associated separation procedures. We will not go into using the relation to the betweenness problem for separation purposes, since the concept of patterns which will be introduced in section 5.3.3 is somehow a generalization. But note that the results of section 3.5 can be used to construct a separation procedure for the WC1P from any separation procedure for the WBWP.

5.3.1 Separation of staircase inequalities

First we consider the separation procedure of the staircase inequalities introduced in section 3.3. These two classes of inequalities F_{1_k} and F_{2_k} are very important for both the standard and the simultaneous problem, since they do not only define facets of the associated polytopes but also cut off most of the forbidden Tucker matrices. In the simultaneous case the staircase inequalities are even sufficient for the IP formulation.

Actually, in the case of the staircase inequalities F_{1_k}, we will separate a more general class of inequalities. These inequalities can be obtained by observing that the "$-$" entry in the last row of the coefficient matrix F_{1_k} can be moved to any position changing the first and last column in an appropriate way (see figure 3.3).

The corresponding $\overline{F}_{1_k}$-inequalities can also be shown to be facet-defining for $P^{m,n}_{\text{C1R}}$ and $P^{m,n}_{\text{C1S}}$. The proofs follow along the same line as those of theorems 3.8 and 4.33.

The new left hand side matrix $\overline{\mathcal{F}}_{1_k}$ is visualized in figure 5.1. As in the original form the size of the matrix is $(k+2, k+2)$ and the right hand side of the inequality is $2k+3$.

We obtain the original F_{1_k}-inequalities, if there are no columns $c_2, \ldots, c_d$, i.e., if $d = 1$.

$$
\begin{array}{c}
 \\
 \\
 \\
 \\
 \\
r_1 \\
\vdots \\
r_{d-1} \\
r_d \\
l
\end{array}
\begin{array}{c}
\begin{array}{ccccccccc} i & & & & j & c_2 & \dots & c_d & h \end{array} \\
\left(\begin{array}{ccccccccc}
+ & + & & & & & & & - \\
0 & + & + & & & & & & - \\
\vdots & & \ddots & \ddots & & & & & \vdots \\
0 & & & + & + & & & & - \\
- & & & & + & + & & & 0 \\
\vdots & & & & & \ddots & \ddots & & \vdots \\
- & & & & & & + & + & 0 \\
- & & & & & & & + & + \\
+ & 0 & \dots & 0 & - & 0 & \dots & 0 & +
\end{array}\right)
\end{array}
\circ x \leq 2k+3
$$

Figure 5.1: $\overline{F}_{1_k}$-inequality

The main task of the separation algorithm is to identify the row l and the columns i, j, and h and to sum appropriate coefficients for rows and columns in between.

We proceed as follows. For every column $i = 1, \dots, n$ we create the complete undirected bipartite graph G^i with the n columns and m rows representing the two node sets. With every edge cr we associate the weight $w^i_{cr} = 1 - x^*_{rc} + \frac{1}{2}x^*_{ri}$, where x^* is the given LP solution to be cut off.

In every weighted graph G^i we now compute for every pair $j, j \neq i$ and $h, h \neq i, h \neq j$ of columns a shortest path between j and h with respect to the assigned edge weights. This way we obtain shortest lengths p^i_{jh}.

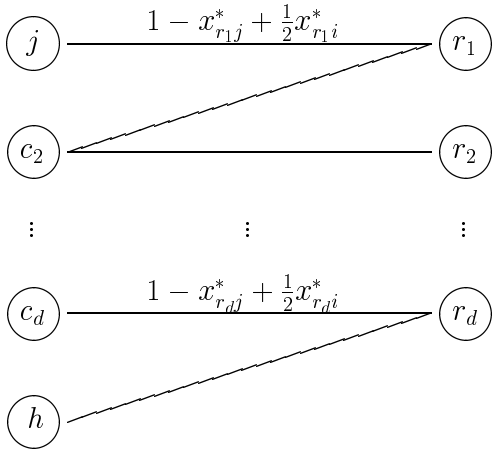


Figure 5.2: Path between columns j and h

For every quadruple i, j, h, l of columns i, j, h and rows l we evaluate

$$p^i_{jh} + p^h_{ji} - x^*_{li} + x^*_{lj} - x^*_{lh} = 2k + 2 - \overline{\mathcal{F}}_{1_k} \circ x^*.$$

For every expression that has value less than -1 we can construct a violated $\overline{F}_{1_k}$-inequality using the shortest paths computed above to include columns $c_2, \ldots, c_d$ and rows $r_1, \ldots, r_d$. If none of these values is less than -1, then no violated $\overline{F}_{1_k}$-inequality and thus no violated F_{1_k}-inequality exists.

In the following this separation algorithm is summarized.

Algorithm 5.5 (`separateStaircase1(lpSolution` x^*`)`)

(1) For each column i of x^* do:

(1.1) Create the weighted bipartite graph G^i with the columns and rows of x^* as node set and with edge weights $w^i_{cr} = 1 - x^*_{rc} + \frac{1}{2}x^*_{ri}$.

(1.2) Compute shortest paths for all pairs of columns $j \neq i$ and $h \neq i$. The corresponding lengths p^i_{jh} are stored in an array.

(2) For each triple i,j,h of columns and each row l do:

(2.1) If $p^i_{jh} + p^h_{ji} - x^*_{li} + x^*_{lj} - x^*_{lh} < -1$ then construct the corresponding violated inequality.

The all-pairs shortest path computations in step (1) take the time $O(n^4)$. Step (2) takes the time $O(n^3 m)$. Summarizing we obtain the time complexity $O(n^3(n+m))$. Therefore the F_{1_k}-inequalities can be separated in polynomial time.

The separation procedure for the F_{2_k}-inequalities (see figure 3.3) works very similarly. The corresponding algorithm looks as follows .

Algorithm 5.6 (`separateStaircase2(lpSolution` x^*`)`)

(1) For each pair of columns i and j of x^* do:

(1.1) Create the weighted bipartite graph G^{ij} with the columns and rows of x^* as node set and with edge weights $w^{ij}_{cr} = 1 - x^*_{rc} + \frac{1}{2}(x^*_{ri} + x^*_{rj})$.

(1.2) Compute shortest paths for all pairs of columns $g \notin \{i,j\}$ and $h \notin \{i,j\}$. The corresponding lengths p^{ij}_{gh} are stored in an array.

(2) For each quadruple i,j,g and h of columns and each pair of rows k and l do:

(2.1) If $p^{ij}_{gh} - x^*_{kg} + x^*_{kh} + x^*_{ki} - x^*_{lj} - x^*_{lg} + x^*_{lh} - x^*_{li} + x^*_{lj} < -1$ then construct the corresponding violated inequality.

Step (1) takes the time $O(n^5)$, step (2) takes $O(n^4 m^2)$. Usually we have at least as many rows as columns. Therefore the overall running time is dominated by the second term $O(n^4 m^2)$.

Taking into account that the F_3- and F_4-inequalities (see figure 3.12) have finite size and therefore can be separated in polynomial time just by enumerating all possible mappings, we have shown that all inequalities constituting the LP relaxations of our IP formulations

3.4 and 4.3.2 can be separated in polynomial time. Nevertheless for practical reasons enumeration of the F_3- and F_4-inequalities is not performed at all and since the running times of the algorithms 5.5 and 5.6 highly depend on the number of columns, we use submatrices of the LP relaxation x^* with at most 90 columns. Section 5.3.4 will show one possibility for a good choice of these submatrices.

5.3.2 Separation of SIR-cuts

As proven in sections 3.1.2 and 4.3 all facet-defining inequalities of $P^{m,n}_{\mathrm{C1R}}$ and $P^{m,n}_{\mathrm{C1S}}$ can be trivially lifted to facets of polytopes with higher dimensions. We assume that we are given a facet class of a "small" polytope, say $C \circ x \leq c_0$ is a facet class of $P^{4,4}_{\mathrm{C1R}}$. Now we want to know whether there is any inequality of this type which is violated by the current LP solution x^*. Let m be the number of rows and n be the number of columns of x^*. Speaking of the combinatorial structure of the consecutive ones problems, all rows are indistinguishable and so are all columns. Therefore $C \circ x \leq c_0$ is facet-defining for $P^{m,n}_{\mathrm{C1R}}$ if x is any $(4,4)$ submatrix of x^*. There are $\binom{m}{4}$ possibilities to choose the rows of x and $\binom{n}{4}$ possibilities to choose the columns. For all these mapping possibilities we are interested in the value of $C \circ x$. If it exceeds c_0 we have found a violated inequality. Finding the maximum of all these left hand sides directly leads to the problem

$$\max_{\substack{I \subset \{1,\ldots,m\}, |I|=4 \\ J \subset \{1,\ldots,n\}, |J|=4}} C \circ X^*_{IJ}.$$

This problem is a variant of the $\mathcal{NP}$-hard quadratic assignment problem, which can be tackled by different heuristics [PRW94]. Having computed the complete description of low-dimensional polytopes, we solve this quadratic assignment problem for all facet classes using the so-called **Grasp** (Greedy Randomized Adaptive Search Procedure) heuristic [LPR94]. Because of the running time the computations of the complete outer descriptions is only practicable for polytopes according to small instances. Therefore we call the generated cutting planes SIR-cuts (**small instance relaxation cuts**). See Christof [Chr97] for details on the theory and practice of SIR-cuts.

Appendices A.2 and A.3 show the complete outer description of $P^{4,4}_{\mathrm{C1R}}$ and $P^{4,4}_{\mathrm{C1S}}$. Furthermore we have computed partial descriptions of $P^{4,5}_{\mathrm{C1R}}$ with 225 classes of facets and $P^{5,4}_{\mathrm{C1R}}$ with 42 classes. All these facet classes are used with the above approach.

Similarly to the separation of the staircase inequalities, it turned out to be much more efficient not to work on the whole LP solution x^*, but only on a submatrix of x^* with the property that the probability to find a violated inequality inside this submatrix is reasonably high. Section 5.3.4 describes a heuristic for finding such submatrices.

5.3.3 Separation by patterns

The separation idea described in this section is based on the principle of maximal feasible sets of columns and rows (see sections 3.6 and 4.2.4).

We assume the LP solution contains a set of integer column-vectors or row-vectors which is not feasible. Then, just as in section 5.1.1, one can construct a cutting plane by cutting off the submatrix building up this infeasible set. But usually the LP solution contains fractional entries whereas the columns or rows of an infeasible set only have binary entries. Therefore we are interested in a kind of indicator, that specifies not only if a binary column or row (in the following we will call them **patterns**) is contained in an LP solution, but that also measures how well this pattern fits to the LP matrix x^*. In the case of an integral LP relaxation this indicator would be defined to be 1 if the pattern is contained anywhere in x^* and 0 otherwise. In the fractional case this behavior can be modeled by making use of the **Manhattan distances** (the Manhattan distance between two vectors v and w of the same size is $\sum_i |v_i - w_i|$) between a pattern and the columns or rows of x^*. To be more exact, let $p = (p_1, \ldots, p_n)$ be a row-pattern and x^* a submatrix of the LP matrix consisting of n columns. Then we compute the indicator

$$y_p^* = 1 - \min \left\{ \sum_{j=1}^{n} |p_i - x_{ij}^*| \;\mid\; 1 \leq i \leq m \right\}.$$

This indicator is computed for all patterns (in the case of column-patterns the roles of column indices and row indices have to be switched) and we try to derive a violated inequality from their values . How this may be performed will be shown in the following.

Example 5.7 *We consider the* 4 *column-patterns*

$$p_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix},\ p_2 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix},\ p_3 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \text{ and } p_4 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}.$$

The submatrix of the LP solution looks as follows:

$$x^* = \begin{pmatrix} 1.0 & 0.1 & 0.8 & 1.0 & 0.3 & 0.0 & 0.0 \\ 1.0 & 0.2 & 0.0 & 0.9 & 0.4 & 0.6 & 1.0 \\ 0.4 & 1.0 & 0.0 & 1.0 & 0.1 & 0.2 & 0.1 \end{pmatrix}$$

According to the formula above we compute for each of the 4 *patterns the best fitting columns. These are column* 4 *for pattern* p_1*, column* 2 *for pattern* p_2*, column* 7 *for pattern* p_3*, and column* 3 *for pattern* p_4*. The corresponding values of the indicators are* $y_{p_1}^* = 0.9$*,* $y_{p_2}^* = 0.7$*,* $y_{p_3}^* = 0.9$*, and* $y_{p_4}^* = 0.8$

Since the 0/1 *matrix built up by the* 4 *patterns is not* C1PR*, we know that at most* 3 *out of the* 4 *patterns are included in an C1MR set. Therefore*

$$y_{p_1} + y_{p_2} + y_{p_3} + y_{p_4} \leq 3$$

is a valid inequality on the indicator variables. This inequality is violated by y^**. Moreover according to our computation of* y^* *we have*

$$y^*_{p_1} = \quad x^*_{14} + x^*_{24} + x^*_{34} - 2, \quad y^*_{p_2} = -x^*_{12} - x^*_{22} + x^*_{32},$$
$$y^*_{p_3} = -x^*_{17} + x^*_{27} - x^*_{37} \text{ , and } y^*_{p_4} = \quad x^*_{13} - x^*_{23} - x^*_{33}.$$

Besides, by testing all integer possibilities the validity of the linking constraints

$$y_{p_1} \geq \quad x_{14} + x_{24} + x_{34} - 2, \quad y_{p_2} \geq -x_{12} - x_{22} + x_{32},$$

$$y_{p_3} \geq -x_{17} + x_{27} - x_{37} \text{ , and } y_{p_4} \geq \quad x_{13} - x_{23} - x_{33}$$

can be verified. After inserting these inequalities into the violated indicator-based inequality we get

$$\begin{pmatrix} 0 & -1 & 1 & 1 & 0 & 0 & -1 \\ 0 & -1 & -1 & 1 & 0 & 0 & 1 \\ 0 & 1 & -1 & 1 & 0 & 0 & -1 \end{pmatrix} \circ x \leq 5.$$

Due to our construction and being consistent with appendix A.2 this inequality is valid for WC1P*, but violated by* x^**.*

The example shows that we can reduce the problem of finding a cutting plane for the WC1P LP solution x^* to find an inequality violating the pattern-based "virtual LP solution" y^*. Now, in terms of patterns with fixed length $3 \leq m \leq 6$, we are able to compute all feasible pattern combinations. Each of these combinations corresponds to a C1FC or C1FR set and for each of these sets we consider the vector of the pattern indicators (y_p). Now a violated inequality can be constructed if and only if y^* lies outside the convex hull of these binary indicator vectors. This method was used by Applegate et al. [ABCC01] for the TSP based on the so-called delayed column generation, introduced by Ford and Fulkerson [FJF58] and Jewel [Jew58]. Since the virtual LP solution y^* is projected on a facet of the convex hull of all feasible indicator vectors, we call this method "projection-cut method". Essentially it needs to solve one linear program.

In the case of column-patterns the complete algorithm looks like follows. Of course the row-pattern separation procedure follows along the same line.

Algorithm 5.8 (`separateColumnPatterns(lpSolution` x^*`)`)

(1) Specify a subset I *of the rows of* x^* *with* $3 \leq |I| = m \leq 5$*.*

(2) For each of the 2^m *different column-patterns* $p = (p_i)$ *scan the best fitting column of* x^* *by setting:*

$$y^*_p = 1.0 - \min \left\{ \sum_{i \in I} |p_i - x^*_{ij}| \;\mid\; 1 \leq j \leq n \right\}$$

(3) Having generated a complete list of all C1FC sets, try to find a valid inequality on the pattern variables violated by y^* *by making use of the projection-cut method.*

(4) If a cutting plane is found, use this inequality to construct a valid inequality for the WC1P *which is violated by the current LP solution* x^*.

Similar to the SIR-approach, the pattern-based separation procedure needs a submatrix of x^* with few rows or columns as input to work efficiently. The question, how such a submatrix can be determined will be discussed in the next section.

Computational comparisons of the SIR and the pattern-based separation will be made in section 7.2. We will see that the separation by patterns is much more effective in the sense of the total running time and generated cuts in the branch-and-cut approach. Note that in our implementation we restricted the length m of the patterns to be $3, 4$ and 5 for column-patterns and 3-6 for row-patterns. Besides we do not only use the 2^m patterns consisting of "0"s or "1"s only, but also consider patterns with "$*$" entries, which means that both zeros and ones are allowed in these positions.

5.3.4 Heuristics for finding violated submatrices

The complexity of the separation procedures described in the previous sections highly depends on the size of the LP matrix. To an extremely high degree this holds for the separation procedure of patterns. As described before it can only be performed for matrices with at most 5 rows or at most 6 columns in an acceptable amount of time. Therefore we are interested in a heuristic that finds a submatrix of the LP relaxation with a high probability that this submatrix contains a violated inequality. Similar to the rounding heuristic in section 5.2.2 we are interested in using the PQ-tree algorithm. Again we need to make the current LP solution integer to use PQ-trees. For this purpose we apply the same probabilistic method as described in 5.2.2.

Since the different separation procedures described in the previous sections need different sizes of the input matrix, we present an algorithm trying to compute a violated (m', n') submatrix of x^*, where m' and n' are part of the input. The main idea is that columns and rows are deleted as long as the remainder stays infeasible. If a deletion leads to a feasible matrix, this step is undone. Rows and columns with a high Manhattan distance between LP matrix and rounded matrix are chosen to be deleted earlier. Note that feasibility of the rounded matrix means it is C1PR or C1PS depending on the considered problem.

Algorithm 5.9 (`findViolatedSubmatrix(lpSolution` x^*`,` m'`,` n'`)`

(1) Round the LP matrix x^* *to an integer matrix* $\overline{x}$ *by setting* $\overline{x}_{ij}$ *to* 1 *with probability* x^*_{ij} *and to* 0 *otherwise*

(2) For each row i *calculate* $r_i = \sum_{j=1}^{n} |\overline{x}_{ij} - x^*_{ij}|$
For each column j *calculate* $c_j = \sum_{i=1}^{m} |\overline{x}_{ij} - x^*_{ij}|$

(3) All rows and columns of $\overline{x}$ *are labeled to be deletable*

(4) While there are deletable rows or columns of $\overline{x}$ *and* $m + n > m' + n'$ *do:*

(4.1) If $n > n'$

(4.1.1) Delete the deletable column of $\overline{x}$ *with the maximum value of* c_i

(4.1.2) Check feasibility of the remaining matrix of $\overline{x}$ *by the PQ-tree algorithm*

(4.1.3) If the submatrix is not feasible

(4.1.3.1) $n = n - 1$

(4.1.3.2) Update the values of r_i

(4.1.4) else undelete the previous deleted column and label it to be undeletable

(4.2) If $m > m'$

(4.2.1) Delete the deletable row of $\overline{x}$ *with the maximum value of* r_i

(4.2.2) Check feasibility of the remaining matrix of $\overline{x}$ *by the PQ-tree algorithm*

(4.2.3) If the submatrix is not feasible

(4.2.3.1) $m = m - 1$

(4.2.3.2) Update the values of c_i

(4.2.4) else undelete the previous deleted row and label it to be undeletable

Below an example of the algorithm for the standard problem is given with $m' = n' = 3$. These are the smallest values that make sense, since all matrices with less than 3 rows or columns are both C1PR and C1PS.

Example 5.10 *The LP matrix before and after the probabilistic rounding looks as follows:*

$$x^* = \begin{pmatrix} 1.0 & 1.0 & 1.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 1.0 & 1.0 & 1.0 \\ 0.0 & 1.0 & 0.5 & 1.0 & 1.0 \\ 0.0 & 0.0 & 0.2 & 1.0 & 0.5 \\ 0.0 & 0.0 & 0.0 & 0.8 & 0.4 \end{pmatrix} \qquad \overline{x} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Computing the Manhattan distances of corresponding rows and columns we get:

$$\begin{array}{lllll} r_1 = 0 & r_2 = 0 & r_3 = 0.5 & r_4 = 0.7 & r_5 = 0.8 \\ c_1 = 0 & c_2 = 0 & c_3 = 0.7 & c_4 = 0.2 & c_5 = 1.1 \end{array}$$

Thus we start with deleting column 5 *of* $\overline{x}$ *and receive:*

$$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

This matrix is not C1PR. *Therefore we proceed with updating the row distances:*

$$\begin{array}{lllll} r_1 = 0 & r_2 = 0 & r_3 = 0.5 & r_4 = 0.2 & r_5 = 0.2 \end{array}$$

Now row 3 *has to be deleted:*

$$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Obviously this matrix is feasible, therefore we undelete and mark row 3 *to be undeletable. The same holds after deleting column* 3*. But after deleting row* 4 *we get the matrix*

$$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

which is still not feasible. Deleting column 4 *leads to a feasible matrix. But the deletions of row* 5 *and column* 1 *are possible and the algorithm terminates with a submatrix of* $\overline{x}$*, which is not feasible, namely the submatrix consisting of the column set* $\{2, 3, 4\}$ *and the row set* $\{1, 2, 3\}$*.*

Now a separation procedure can be processed on the corresponding submatrix of x^**, which is*

$$\begin{pmatrix} 1.0 & 1.0 & 0.0 \\ 0.0 & 1.0 & 1.0 \\ 1.0 & 0.5 & 1.0 \end{pmatrix}.$$

In this case any of the separation procedures described in the previous sections would find the violated inequality

$$\begin{pmatrix} 1 & 1 & -1 \\ -1 & 1 & 1 \\ 1 & -1 & 1 \end{pmatrix} \circ x \leq 5.$$

This cutting plane can be generated and added to the current LP.

As already mentioned in 5.1.1, a cutting plane can even directly be constructed from the created submatrix of the previous example without any additional separation procedure. This is because the Manhattan distance of this submatrix and its rounded counterpart is less than 1. Therefore the idea of section 5.1.1 to cut off an integer vector would also be suitable in this case. Consequently this algorithm can also be used as a heuristic standalone separation procedure. Its running time is very efficient. Since $O(m + n)$ PQ-trees have to be built, we get $O((m + n)mn)$ as total running time.

Chapter 6

Applications

In this chapter we present some applications of the WC1P and the WSC1P. The following sections will show that there is a huge variety of fields where the concept of "consecutive ones" occurs. We will start with an application in computational biology.

6.1 The physical mapping problem

6.1.1 Biological background

The main goal of the Human Genome Project is to reconstruct the linear sequence of the human chromosome consisting of roughly 10^9 bases. One step leading to that goal is the construction of so-called **physical maps** that enable us to localize important features along the chromosome, such as **clones** which are small DNA fragments. Each clone corresponds to an interval of the chromosome. Now a library of clones is given, all of them being subsets of the same larger piece of the DNA. The goal is to reconstruct the relative order of these clones occurring in the DNA piece. For this purpose the concept of **probes** is introduced. The probes also are subintervals of the DNA. Each probe is tested against the clone library. Testing means that for each clone-probe pair a **hybridization experiment** is performed. A positive result (the probe hybridizes to the clone) is an indicator that the probe and the clone overlap, a negative result indicates that probe and clone do not have a common substring. Let m denote the number of clones and n the number of probes then all these results can be written down in an (m, n) binary matrix A with

$$a_{ij} = \begin{cases} 1 & : \text{ if probe } j \text{ hybridizes to clone } i \\ 0 & : \text{ otherwise.} \end{cases}$$

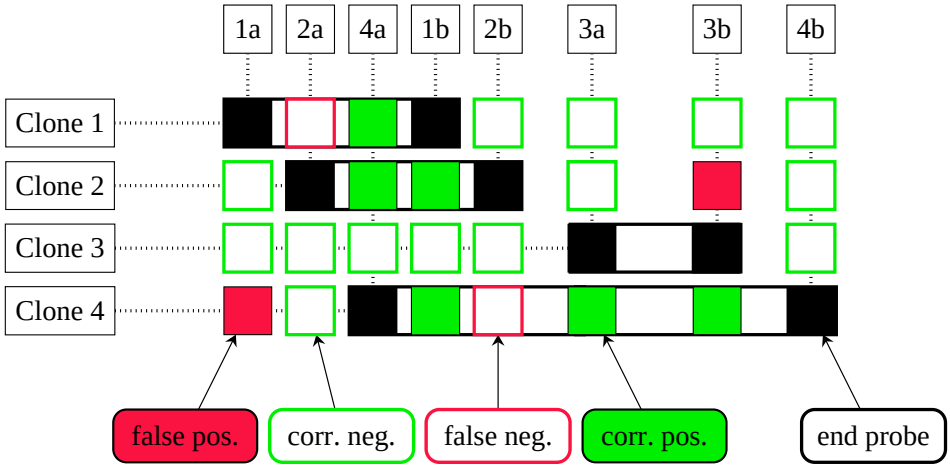
This matrix is called **hybridization matrix**.

The **physical mapping problem** is to reconstruct the order of the probes (corresponding to the columns of the matrix) in which they occur in the chromosome. From this probe order one can easily derive the order of the clones. Now assume that the columns of A are ordered in the correct way and there were no errors in the hybridization experiments.

Then for each pair of probes overlapping with the same clone all probes lying between this pair also must hybridize to this clone. In other words the "1"s in the hybridization matrix would occur consecutively in each row and therefore A would be C1PR. The PQ-tree algorithm could be used to find all admissible probe orders.

Unfortunately, hybridization experiments are highly influenced by errors. The error that a probe hybridizes to a clone though they do not overlap is called **false positive** and the other way round, an error is called **false negative** if a probe-clone pair has a common substring of the DNA but they do not hybridize to each other. A third type of error is the so called **chimerism**. A chimeric clone does not represent a single interval of the DNA but contains two or more unrelated substrings. In the following we want to tackle the physical mapping problem in the presence of false positive and false negative errors but we do not consider chimerism.

In practice there are different ways to select the set of probes. One possibility is to extract the probes from both ends of the clones. This procedure is known as **physical mapping with end probes** and leads to a weighted betweenness problem (see [COR98] and section 3.5). Note that for each probe it is known from which clone it has been extracted. Figure 6.1 gives an example of the hybridization experiments for the physical mapping problem with end probes.



hybridization matrix

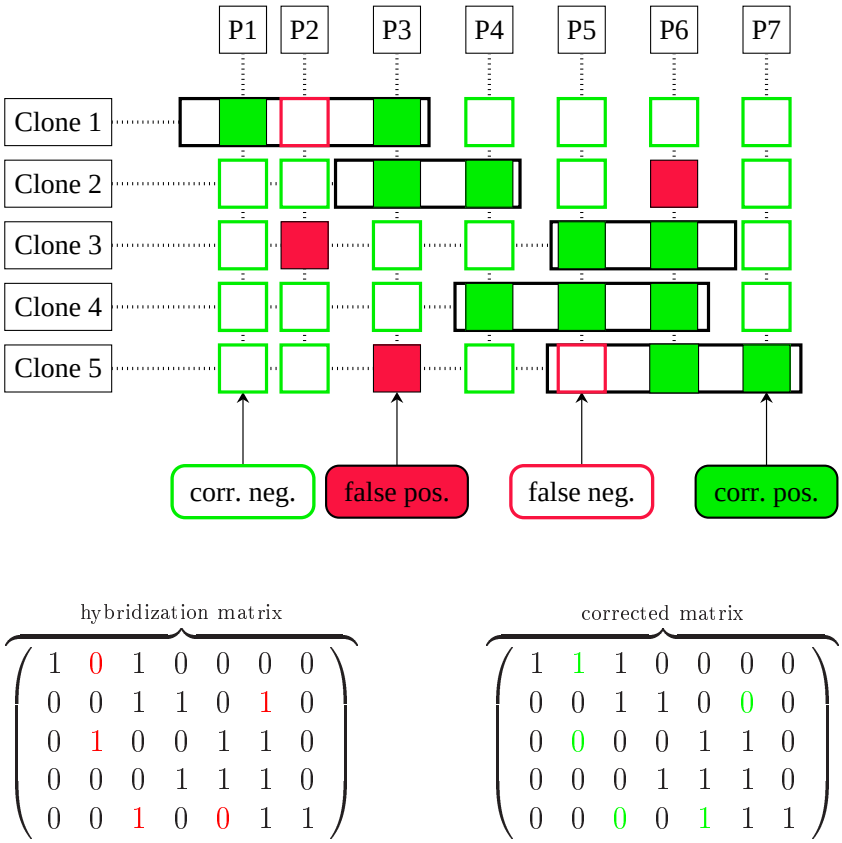
$$\begin{pmatrix} * & 0 & 1 & * & 0 & 0 & 0 & 0 \\ 0 & * & 1 & 1 & * & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & * & * & 0 \\ 1 & 0 & * & 1 & 0 & 1 & 1 & * \end{pmatrix}$$

corrected matrix

$$\begin{pmatrix} * & 1 & 1 & * & 0 & 0 & 0 & 0 \\ 0 & * & 1 & 1 & * & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & * & * & 0 \\ 0 & 0 & * & 1 & 1 & 1 & 1 & * \end{pmatrix}$$

Figure 6.1: Hybridization experiments for the physical mapping problem with end probes, errors are marked "red"

In the following we deal with another way for obtaining the probes, namely they are chosen as a certain subset of the clone library. Therefore no end probe information is known. Figure 6.2 shows an example how the results of the hybridization experiments could look like in this case.



$$\begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{pmatrix} \qquad \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

Figure 6.2: Physical mapping without end probe information

Our goal is now to reconstruct the corrected matrix on the right-hand side from the hybridization matrix on the left-hand side. In this case the two matrices differ by 5 entries each corresponding to an error. This problem of constructing a C1PR matrix by switching some entries of an input matrix reminds us of the weighted consecutive ones problem. Indeed, we will see in the following that making use of the maximum-likelihood idea leads to a WC1P with the objective function depending on the error rates.

6.1.2 Modeling as WC1P

Alizadeh et al. [AKNW95] were the first to introduce a **maximum-likelihood** approach to the physical mapping problem. The idea is to find the C1PR matrix B that maximizes $p(B|A)$, where $p(B|A)$ is the probability that B is the correct matrix if A is the observed

hybridization matrix. Using Bayes' Theorem and assuming that A is the hybridization matrix ($p(A) = 1$) we get

$$p(B|A) = p(A|B)p(B),$$

where $p(A|B)$ is the probability that A is observed if B is the correct matrix and $p(B)$ is the probability that B is the true matrix. Without any information of hybridization experiments we assume B to be uniformly distributed over all C1PR matrices. Therefore $p(B)$ has the same value for all C1PR matrices B. Thus it remains to maximize $p(A|B)$. Now let p_{fp} (p_{fn}) be the probability of a false positive (false negative) error and $n_{fp}(B)$ ($n_{fn}(B)$) be the number of 1's (of 0's) in positions of A where the correct matrix B has a 0 (a 1). Further let n be the total number of entries in the matrices A and B. Assuming that all hybridization experiments are independent of each other we have

$$p(A|B) = \left(\frac{p_{fp}}{1-p_{fp}}\right)^{n_{fp}(B)} \left(\frac{p_{fn}}{1-p_{fn}}\right)^{n_{fn}(B)} (1-p_{fp})^n (1-p_{fn})^n.$$

After logarithmizing and taking into account that the two terms on the right-hand side do not depend on B we want to maximize

$$n_{fp}(B) \log\left(\frac{p_{fp}}{1-p_{fp}}\right) + n_{fn}(B) \log\left(\frac{p_{fn}}{1-p_{fn}}\right)$$

over all C1PR matrices B. Due to the definition of $n_{fp}(B)$ and $n_{fn}(B)$ this is equivalent to the WC1P

$$\begin{aligned} &\max C \circ B \\ s.t.\quad &B \text{ is C1PR,} \end{aligned}$$

with

$$c_{ij} = \begin{cases} \log\left(\frac{p_{fn}}{1-p_{fn}}\right) & : \quad a_{ij} = 0 \\ -\log\left(\frac{p_{fp}}{1-p_{fp}}\right) & : \quad a_{ij} = 1. \end{cases}$$

If the false positive and false negative rates are not known, they can be estimated by the following procedure. We first assume $p_{fp} = p_{fn}$ and solve the resulting WC1P (in this case we have $|c_{ij}| = 1$, $\forall i, j$). From the difference between the optimal solution and the hybridization matrix we can make a second estimation for the error rates. This process can be repeated until the error rates do not change any more.

Section 7.1 will give computational results on real world data and a comparison between the WC1P and a Hamming distance approach.

6.2 Seriation in archaeology

Seriation or **sequence dating** in archaeology was first formulated in 1899 by Flinders Petrie (see Kendall [Ken69]). In this problem we are given a set of graves (corresponding to the columns of our matrix) and a set of artifacts (corresponding to the rows of the

matrix) which were found (1-entry in the matrix) or not (0-entry in the matrix) in a certain grave. Assuming that the artifacts were in use over a certain time-interval, we can try to reconstruct the chronology of the graves from the "grave/artifact" matrix. Indeed, if our assumption would be true, then the PQ-tree algorithm would give all possible seriations of the graves. Again, as in the physical mapping problem one has to deal with errors. For example, one artifact was not found in a certain grave though it was in use at that time. Given estimations for the probability of such errors we can obtain the sequence dating by solving a WC1P similar to the previous section.

6.3 Making movies

A further application of the WC1P arises when **making a movie**. Normally the actors of a movie have to be paid from their first shooting day up to their last. Assume we are given the binary matrix which says for each actor and each scene of the film whether the actor is involved in that scene or not. In addition we know the honorarium per day for each actor. To save money the film company is interested in organizing the shooting in such a way that the actors have as few days without work as possible. To be more exact, taking the different honorariums into account, we have to determine the optimal order of the scenes to be shot. This can be done by solving a WC1P, where the objective function depends on the payments per day for the actors. In addition variables associated with a 1 in the "actor/scene" matrix must be fixed to 1 to enforce that an actor is present if he is involved in a scene. The reference to this interesting application was given by Pochet [Poc03].

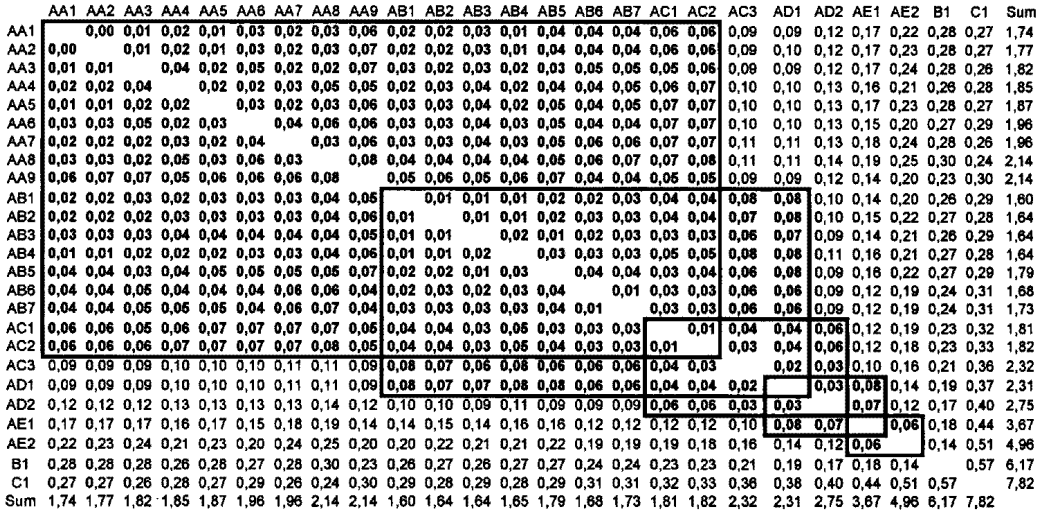
6.4 Analyzing inorganic crystal structure types

6.4.1 Computing clusters

Assume a set S of elements is given and in addition a distance matrix D where d_{ij} denotes the distance between elements i and j. We now want to compute **clusters**. Clusters are (not necessarily disjoint) subsets of S with "small" distances inside the subsets.

This problem arises whenever one wants to classify certain objects into groups given a measure of "similarity" specified by a distance for each pair of the objects. The values should be nonnegative and a distance value of 0 means that the objects are nearly identical.

Figure 6.3 for example shows a symmetric distance matrix for 25 representatives of inorganic crystal structure types. We do not elaborate on the mineralogical background and how the differences d_{ij} between the structures are calculated, but refer to Bergerhoff et al. [BBBD99]. The rows and columns of the matrix were permuted in such a way that symmetrical submatrices with small distances arise. They correspond to clusters of related crystal structure types.



	AA1	AA2	AA3	AA4	AA5	AA6	AA7	AA8	AA9	AB1	AB2	AB3	AB4	AB5	AB6	AB7	AC1	AC2	AC3	AD1	AD2	AE1	AE2	B1	C1	Sum
AA1		0,00	0,01	0,02	0,01	0,03	0,02	0,03	0,06	0,02	0,02	0,03	0,01	0,04	0,04	0,04	0,06	0,06	0,09	0,09	0,12	0,17	0,22	0,28	0,27	1,74
AA2	0,00		0,01	0,02	0,01	0,03	0,02	0,03	0,07	0,02	0,02	0,03	0,01	0,04	0,04	0,04	0,06	0,06	0,09	0,10	0,12	0,17	0,23	0,28	0,27	1,77
AA3	0,01	0,01		0,04	0,02	0,05	0,02	0,02	0,07	0,03	0,02	0,03	0,02	0,03	0,05	0,05	0,05	0,06	0,09	0,09	0,12	0,17	0,24	0,28	0,26	1,82
AA4	0,02	0,02	0,04		0,02	0,02	0,03	0,05	0,05	0,02	0,03	0,04	0,02	0,04	0,04	0,05	0,06	0,07	0,10	0,10	0,13	0,16	0,21	0,26	0,28	1,85
AA5	0,01	0,01	0,02	0,02		0,03	0,02	0,03	0,06	0,03	0,03	0,04	0,02	0,05	0,04	0,05	0,07	0,07	0,10	0,10	0,13	0,17	0,23	0,28	0,27	1,87
AA6	0,03	0,03	0,05	0,02	0,03		0,04	0,06	0,06	0,03	0,03	0,04	0,03	0,05	0,04	0,04	0,07	0,07	0,10	0,10	0,13	0,15	0,20	0,27	0,29	1,96
AA7	0,02	0,02	0,02	0,03	0,02	0,04		0,03	0,06	0,03	0,03	0,04	0,03	0,05	0,06	0,06	0,07	0,07	0,11	0,11	0,13	0,18	0,24	0,28	0,26	1,96
AA8	0,03	0,03	0,02	0,05	0,03	0,06	0,03		0,08	0,04	0,04	0,04	0,04	0,05	0,06	0,07	0,07	0,08	0,11	0,11	0,14	0,19	0,25	0,30	0,24	2,14
AA9	0,06	0,07	0,07	0,05	0,06	0,06	0,06	0,08		0,05	0,06	0,05	0,06	0,07	0,04	0,04	0,05	0,05	0,09	0,09	0,12	0,14	0,20	0,23	0,30	2,14
AB1	0,02	0,02	0,03	0,02	0,03	0,03	0,03	0,04	0,05		0,01	0,01	0,01	0,02	0,02	0,03	0,04	0,04	0,08	0,08	0,10	0,14	0,20	0,26	0,29	1,60
AB2	0,02	0,02	0,02	0,03	0,03	0,03	0,03	0,04	0,06	0,01		0,01	0,01	0,02	0,03	0,03	0,04	0,04	0,07	0,08	0,10	0,15	0,22	0,27	0,28	1,64
AB3	0,03	0,03	0,03	0,04	0,04	0,04	0,04	0,05	0,01	0,01	0,01		0,02	0,01	0,02	0,03	0,03	0,03	0,06	0,07	0,09	0,14	0,21	0,26	0,29	1,64
AB4	0,01	0,01	0,02	0,02	0,02	0,03	0,03	0,04	0,06	0,01	0,01	0,02		0,03	0,03	0,03	0,05	0,05	0,08	0,08	0,11	0,16	0,21	0,27	0,28	1,64
AB5	0,04	0,04	0,03	0,04	0,05	0,05	0,05	0,05	0,07	0,02	0,02	0,01	0,03		0,04	0,04	0,03	0,04	0,06	0,08	0,09	0,16	0,22	0,27	0,29	1,79
AB6	0,04	0,04	0,05	0,04	0,04	0,04	0,06	0,06	0,04	0,02	0,03	0,02	0,03	0,04		0,01	0,03	0,03	0,06	0,06	0,09	0,12	0,19	0,24	0,31	1,68
AB7	0,04	0,04	0,05	0,05	0,05	0,04	0,06	0,07	0,04	0,03	0,03	0,03	0,03	0,04	0,01		0,03	0,03	0,06	0,06	0,09	0,12	0,19	0,24	0,31	1,73
AC1	0,06	0,06	0,05	0,06	0,07	0,07	0,07	0,07	0,05	0,04	0,04	0,03	0,05	0,03	0,03	0,03		0,01	0,04	0,04	0,06	0,12	0,19	0,23	0,32	1,81
AC2	0,06	0,06	0,06	0,07	0,07	0,07	0,07	0,08	0,05	0,04	0,04	0,03	0,05	0,04	0,03	0,03	0,01		0,03	0,04	0,06	0,12	0,18	0,23	0,33	1,82
AC3	0,09	0,09	0,09	0,10	0,10	0,10	0,11	0,11	0,09	0,08	0,07	0,06	0,08	0,06	0,06	0,06	0,04	0,03		0,02	0,03	0,10	0,16	0,21	0,36	2,32
AD1	0,09	0,09	0,09	0,10	0,10	0,10	0,11	0,11	0,09	0,08	0,07	0,07	0,08	0,08	0,06	0,06	0,04	0,04	0,02		0,03	0,08	0,14	0,19	0,37	2,31
AD2	0,12	0,12	0,12	0,13	0,13	0,13	0,14	0,12	0,10	0,10	0,09	0,11	0,09	0,09	0,09	0,09	0,06	0,06	0,03	0,03		0,07	0,12	0,17	0,40	2,75
AE1	0,17	0,17	0,17	0,16	0,17	0,15	0,18	0,19	0,14	0,14	0,15	0,14	0,16	0,16	0,12	0,12	0,12	0,12	0,10	0,08	0,07		0,06	0,18	0,44	3,67
AE2	0,22	0,23	0,24	0,21	0,23	0,20	0,24	0,25	0,20	0,20	0,22	0,21	0,21	0,22	0,19	0,19	0,19	0,18	0,16	0,14	0,12	0,06		0,14	0,51	4,96
B1	0,28	0,28	0,28	0,26	0,28	0,27	0,28	0,30	0,23	0,26	0,27	0,26	0,27	0,27	0,24	0,24	0,23	0,23	0,21	0,19	0,17	0,18	0,14		0,57	6,17
C1	0,27	0,27	0,26	0,28	0,27	0,29	0,26	0,24	0,30	0,29	0,28	0,29	0,28	0,29	0,31	0,31	0,32	0,33	0,36	0,38	0,40	0,44	0,51	0,57		7,82
Sum	1,74	1,77	1,82	1,85	1,87	1,96	1,96	2,14	2,14	1,60	1,64	1,64	1,65	1,79	1,68	1,73	1,81	1,82	2,32	2,31	2,75	3,67	4,96	6,17	7,82	

	Formula	COL		Formula	COL		Formula	COL
AA1	MgTiO3	65794	AB1	MgGeO3	200415	AC2	MgSiO3	31176
AA2	CoTiO3	48107	AB2	MnTiO3	60006	AC3	CdGeO3	30971
AA3	MnFe.5Sb.5O3	24431	AB3	ZnGeO3	33722	AD1	LiAsO3	202862
AA4	NiMnO3	31853	AB4	FeTiO3	67046	AD2	NaSbO3	78011
AA5	NiTiO3	33854	AB5	CaSnO3	29204	AE1	CrSiTe3	71020
AA6	CoMnO3	31854	AB6	NaMnCl3	2552	AE2	(Cu.5Al.5)PSe3	67892
AA7	ZnTiO3	22382	AB7	CuVO3	19046	B1	KSbO3	33546
AA8	MnSnO3	29203	AC1	MnGeO3	69591	C1	NH4SnF3	1452
AA9	CdTiO3	15989						

Figure 6.3: Distance matrix with clusters from Bergerhoff et al. [BBBD99]

6.4.2 Modeling as WSC1P

The shape of the matrix in figure 6.3 resembles a matrix being C1PS. Indeed, if the entries inside the submatrices are assumed to be 1 and the entries outside to be 0, the resulting binary matrix is C1PS. Distances inside should have small values, those outside higher values. This leads to the idea to use a WSC1P to find the clusters. Note that C1PS matrices are not necessarily symmetric, therefore we have to enforce this separately. In the branch-and-cut code this is done by identifying the variables x_{ij} and x_{ji}.

$$\begin{aligned} &\min C \circ A \\ s.t.\quad &A \text{ is C1PS and } A = A^T. \end{aligned}$$

Of course the objective function matrix C should depend on the distance matrix D. We use the following transformation. Assume a user specified bound b is given with the property that two elements should lie inside a common cluster if their distance is less than b. Therefore c_{ij} should be negative if $d_{ij} < b$ and positive if $d_{ij} > b$ and it suggests itself to set $c_{ij} = d_{ij} - b$.

We applied these ideas to the distance matrix of figure 6.4. Note that for convenience the original values of the distances are multiplied by 100 and rounded to the next integers. Using $b = 0.10$ we obtained figure 6.5 as optimal solution of the WSC1P. A comparison of both figures shows that the computed clusters are almost the same as the "hand-optimized"

d_{ij}	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
1	–	13	14	15	14	15	14	14	12	7	7	7	7	18	16	17	15	15	14	13	14	15	16	17	17	20	24	26
2	13	–	1	1	2	3	4	4	7	7	7	7	7	10	11	11	12	13	13	13	14	15	16	17	16	20	24	26
3	14	1	–	1	1	3	3	4	6	7	7	7	7	9	11	11	12	13	13	13	13	14	15	16	15	19	23	25
4	15	1	1	–	1	2	3	3	5	7	7	7	7	9	10	10	11	12	12	12	13	14	14	16	14	19	22	24
5	14	2	1	1	–	1	2	2	5	7	7	7	7	8	9	9	10	11	11	11	12	13	14	15	13	18	21	24
6	15	3	3	2	1	–	1	1	3	8	8	8	8	6	8	8	9	10	10	10	11	11	12	13	12	17	20	22
7	14	4	3	3	2	1	–	1	3	7	7	7	6	6	7	8	9	9	10	10	10	11	12	13	12	16	19	22
8	14	4	4	3	2	1	1	–	2	7	7	7	7	5	7	7	8	9	9	9	9	10	11	12	11	15	19	21
9	12	7	6	5	5	3	3	2	–	5	5	5	5	5	4	5	6	6	7	7	7	8	9	10	9	13	16	18
10	7	7	7	7	7	8	7	7	5	–	0	0	0	10	9	9	8	8	7	6	6	7	8	9	10	12	16	18
11	7	7	7	7	7	8	7	7	5	0	–	0	0	10	9	9	8	8	7	6	6	7	8	9	10	12	16	18
12	7	7	7	7	7	8	7	7	5	0	0	–	0	10	9	9	8	8	7	6	6	7	8	9	10	12	16	18
13	7	7	7	7	7	8	6	7	5	0	0	0	–	10	9	9	7	7	6	6	6	7	8	9	10	12	15	18
14	18	10	9	9	8	6	6	5	5	10	10	10	10	–	1	2	3	3	3	4	4	5	6	7	5	10	13	15
15	16	11	11	10	9	8	7	7	4	9	9	9	9	1	–	0	1	2	2	3	3	3	4	5	4	8	12	13
16	17	11	11	10	9	8	8	7	5	9	9	9	9	2	0	–	1	2	2	3	3	3	4	5	4	8	11	13
17	15	12	12	11	10	9	9	8	6	8	8	8	7	3	1	1	–	1	1	2	1	2	3	4	3	7	10	12
18	15	13	13	12	11	10	9	9	6	8	8	8	7	3	2	2	1	–	1	2	1	1	2	3	2	6	9	11
19	14	13	13	12	11	10	10	9	7	7	7	7	6	3	2	2	1	1	–	1	1	1	2	3	3	6	9	11
20	13	13	13	12	11	10	10	9	7	6	6	6	6	4	3	3	2	2	1	–	1	1	2	3	4	6	9	11
21	14	14	13	13	12	11	10	9	7	6	6	6	6	4	3	3	1	1	1	1	–	1	2	3	3	6	9	11
22	15	15	14	14	13	11	11	10	8	7	7	7	7	5	3	3	2	1	1	1	1	–	1	2	3	5	8	10
23	16	16	15	14	14	12	12	11	9	8	8	8	8	6	4	4	3	2	2	2	2	1	–	1	3	4	7	9
24	17	17	16	16	15	13	13	12	10	9	9	9	9	7	5	5	4	3	3	3	3	2	1	–	2	3	6	8
25	17	16	15	14	13	12	12	11	9	10	10	10	10	5	4	4	3	2	3	4	3	3	3	2	–	4	7	9
26	20	20	19	19	18	17	16	15	13	12	12	12	12	10	8	8	7	6	6	6	6	5	4	3	4	–	3	5
27	24	24	23	22	21	20	19	19	16	16	16	16	15	13	12	11	10	9	9	9	9	8	7	6	7	3	–	2
28	26	26	25	24	24	22	22	21	18	18	18	18	18	15	13	13	12	11	11	11	11	10	9	8	9	5	2	–

Figure 6.4: Clusters computed by Bergerhoff [Ber99]

d_{ij}	1	2	3	4	5	6	7	8	10	9	13	11	12	14	15	16	17	20	18	19	21	25	22	23	24	26	27	28
1	–	13	14	15	14	15	14	14	7	12	7	7	7	18	16	17	15	13	15	14	14	17	15	16	17	20	24	26
2	13	–	1	1	2	3	4	4	7	7	7	7	7	10	11	11	12	13	13	13	14	16	15	16	17	20	24	26
3	14	1	–	1	1	3	3	4	7	6	7	7	7	9	11	11	12	13	13	13	13	15	14	15	16	19	23	25
4	15	1	1	–	1	2	3	3	7	5	7	7	7	9	10	10	11	12	12	12	13	14	14	14	16	19	22	24
5	14	2	1	1	–	1	2	2	7	5	7	7	7	8	9	9	10	11	11	11	12	13	13	14	15	18	21	24
6	15	3	3	2	1	–	1	1	8	3	8	8	8	6	8	8	9	10	10	10	11	12	11	12	13	17	20	22
7	14	4	3	3	2	1	–	1	7	3	6	7	7	6	7	8	9	10	9	10	10	12	11	12	13	16	19	22
8	14	4	4	3	2	1	1	–	7	2	7	7	7	5	7	7	8	9	9	9	9	11	10	11	12	15	19	21
10	7	7	7	7	7	8	7	7	–	5	0	0	0	10	9	9	8	6	8	7	6	10	7	8	9	12	16	18
9	12	7	6	5	5	3	3	2	5	–	5	5	5	5	4	5	6	7	6	7	7	9	8	9	10	13	16	18
13	7	7	7	7	7	8	6	7	0	5	–	0	0	10	9	9	7	6	7	6	6	10	7	8	9	12	15	18
11	7	7	7	7	7	8	7	7	0	5	0	–	0	10	9	9	8	6	8	7	6	10	7	8	9	12	16	18
12	7	7	7	7	7	8	7	7	0	5	0	0	–	10	9	9	8	6	8	7	6	10	7	8	9	12	16	18
14	18	10	9	9	8	6	6	5	10	5	10	10	10	–	1	2	3	4	3	3	4	5	5	6	7	10	13	15
15	16	11	11	10	9	8	7	7	9	4	9	9	9	1	–	0	1	3	2	2	3	4	3	4	5	8	12	13
16	17	11	11	10	9	8	8	7	9	5	9	9	9	2	0	–	1	3	2	2	3	4	3	4	5	8	11	13
17	15	12	12	11	10	9	9	8	8	6	7	8	8	3	1	1	–	2	1	1	1	3	2	3	4	7	10	12
20	13	13	13	12	11	10	10	9	6	7	6	6	6	4	3	3	2	–	2	1	1	4	1	2	3	6	9	11
18	15	13	13	12	11	10	9	9	8	6	7	8	8	3	2	2	1	2	–	1	1	2	1	2	3	6	9	11
19	14	13	13	12	11	10	10	9	7	7	6	7	7	3	2	2	1	1	1	–	1	3	1	2	3	6	9	11
21	14	14	13	13	12	11	10	9	6	7	6	6	6	4	3	3	1	1	1	1	–	3	1	2	3	6	9	11
25	17	16	15	14	13	12	12	11	10	9	10	10	10	5	4	4	3	4	2	3	3	–	3	3	2	4	7	9
22	15	15	14	14	13	11	11	10	7	8	7	7	7	5	3	3	2	1	1	1	1	3	–	1	2	5	8	10
23	16	16	15	14	14	12	12	11	8	9	8	8	8	6	4	4	3	2	2	2	2	3	1	–	1	4	7	9
24	17	17	16	16	15	13	13	12	9	10	9	9	9	7	5	5	4	3	3	3	3	2	2	1	–	3	6	8
26	20	20	19	19	18	17	16	15	12	13	12	12	12	10	8	8	7	6	6	6	6	4	5	4	3	–	3	5
27	24	24	23	22	21	20	19	19	16	16	15	16	16	13	12	11	10	9	9	9	9	7	8	7	6	3	–	2
28	26	26	25	24	24	22	22	21	18	18	18	18	18	15	13	13	12	11	11	11	11	9	10	9	8	5	2	–

Figure 6.5: Clusters obtained by an optimized WSC1P

ones. Most of the differences are due to the fact that an entry ij with $d_{ij} = b$ can lie both in- and outside a cluster if both possibilities are feasible.

This problem was solved in the root node of the branch-and-cut tree and took about 20 CPU seconds on a SUN Ultra 80 with UltraSparc IIi 450 CPU. Section 7.4 gives more computational results on distance matrices with different sizes using different bounds.

6.5 Identifying blocks of matrices

As seen in the previous section we are able to find submatrices with certain properties, such as low differences inside the submatrices, with the help of a WSC1P.

An application arising from integer and linear programming is to decompose a matrix into so-called **bordered block diagonal form** (see Borndörfer et al. [BFM98]). That is, every row of the matrix is assigned to at most one of β blocks, each block contains at most κ rows, and no two rows in different blocks have a common nonzero entry in a column. The set of rows that are not assigned to a block is called the border. Such a form of an LP or MIP matrix (mixed integer programming) is helpful for speeding up the solution process.

Except for the border, matrices of this form are C1PS. Therefore one way to identify the block structure of a matrix could be to solve a WSC1P. The objective function values of the nonzero entries should be positive and those of the zero entries negative. Furthermore the absolute values of the nonzero entries should be chosen bigger than the others. We tried to apply this idea to the LP and MIP matrices of [BFM98] but overall we were not able to solve them to optimality. Maybe the reason is, that we do not handle the border so far.

But the following example shows that this approach is promising. We created a $(200, 100)$ binary matrix including 15 blocks at random. The entries inside the blocks are set to "nonzero" with probability 90%, all the other entries are set to "0". The objective function value was 10 for the nonzero entries and -1 for the remainder. In contrast to the last section we are maximizing. Figure 6.6 shows the generated matrix on the left-hand side. The right-hand side shows the decomposition into blocks according to the optimal WSC1P solution. All the 15 blocks were found by this solution. The branch-and-cut tree required 47 nodes. The total running time was almost 7 hours on a SUN Ultra 80 with UltraSparc IIi 450 CPU.

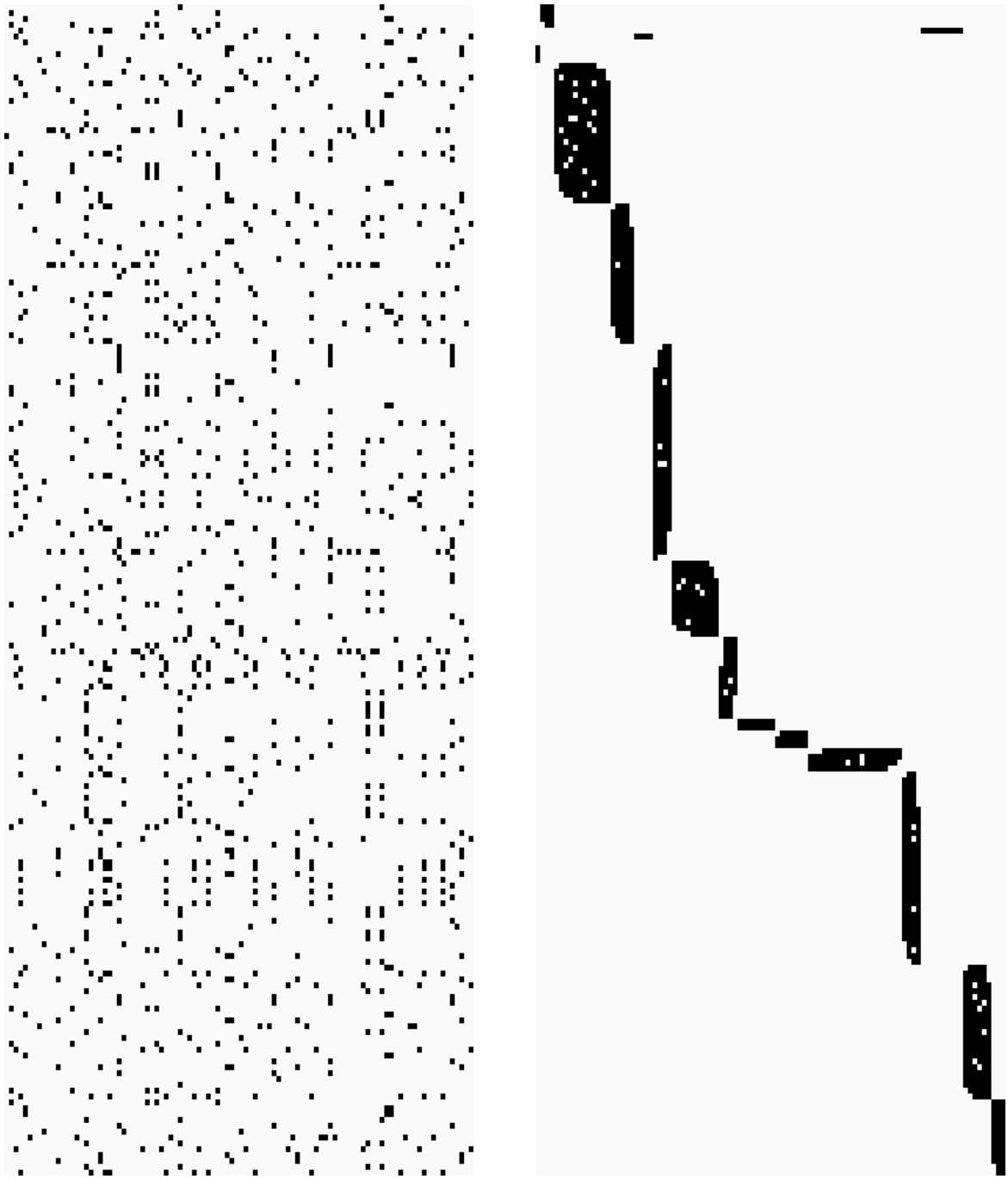


Figure 6.6: Randomly generated block matrix and the WSC1P solution

Chapter 7

Computational Results

We implemented a branch-and-cut code for the WC1P and the WSC1P using the software framework ABACUS [Thi95, JT97a, JT97b] with CPLEX 6.5.3 as LP-Solver. In addition we used an implementation of the PQ-tree algorithm as Template Class in C++ provided by Leipert [Lei97]. The feasibility test, the rounding heuristic, and all separation procedures described in chapter 5 were implemented in C++.

All computations were performed on a SUN Ultra 80 with UltraSparc IIi 450 CPU. For the separation of cutting planes defined on small submatrices we first used the SIR-cut separation procedure. But as section 7.2 will show, the separation procedure based on the pattern principle is much more effective. Therefore our current code only uses the separation by patterns and the exact separation of the staircase inequalities. As already mentioned in chapter 5 the running times of both separation procedures strongly depend on the size of the submatrix they were performed on. Therefore we called them in a hierarchical fashion, i.e., first on small submatrices and, if no cuts were found, the submatrices were increased up to a maximal column-pattern length of 5, row-pattern length of 6 resp., and up to a maximal number of 90 columns for the staircase separation. The submatrices were chosen heuristically as described in section 5.3.4.

The rounding heuristic (as described in section 5.2.2) was performed after every solution of an LP. Normally the number of permutations represented by the generated PQ-tree is too large to be enumerable. Therefore only a few permutations are selected at random. Whenever a new solution increasing the primal bound is found, the number of permutations to be checked is increased.

Almost all parts of the branch-and-cut implementation can be used both for solving the WC1P and for solving the WSC1P. Of course, for testing if the the current LP solution is feasible for WSC1P, we have to perform the PQ-tree algorithm both for the LP matrix and its transpose. In this case also the separation of the staircase inequalities is extended to the transposed LP solution. For the pattern separation we only need to use different feasible sets for the standard and the simultaneous problem. The remainder of this separation procedure can be applied to both cases.

As mentioned in section 5.2.2 the rounding heuristic works very effectively only in the standard case. We tried a modified version also for the simultaneous case but mostly the

optimum was found by a feasible LP solution (see section 7.4).

7.1 Physical mapping results

We applied our code on the physical mapping problem arising from the **Xylella fastidiosa genome project** (see Silvestri et al. [SSdS+00] and Frohme et al. [FCH+00]). The hybridization matrix was obtained from Frohme [Fro00].

The data contains 1037 clones and 181 probes and the objective function was computed as described in section 6.1. It was not possible to solve the WC1P associated to the whole hybridization matrix to optimality in reasonable time. Therefore we additionally worked on smaller submatrices. These submatrices were chosen in such a way that they also represent complete subintervals of the chromosome (the original solution was known).

Table 7.1 shows the computational results for the physical mapping problems defined on 6 submatrices including the whole hybridization matrix. As usual m and n denote the

m	n	t_{tot}	t_{best}	n_{sub}	n_{lp}	lb	ub	hd
230	30	0:01:09	0:00:29	3	12	16999	16999	16394
267	40	0:02:29	0:00:45	3	14	20805	20805	20200
310	50	0:04:26	0:01:36	5	21	25813	25813	25496
320	60	0:05:20	0:01:49	3	19	28527	28527	27881
430	70	>200:00:00	0:12:24	>172	>2000	40259	40316	39523
1037	181	>200:00:00	≈ 50:00:00	>14	>500	104573	105702	103910

Table 7.1: Experiments on submatrices of the Xylella fastidiosa hybridization matrix, the times are given in hours:minutes:seconds

number of rows and columns of the submatrices. The total running time of the branch-and-cut process is denoted by t_{tot}. t_{best} is the time required to find the best known feasible solution. This solution was always found by the rounding heuristic. n_{sub}, n_{lp} resp., are the number of subproblems, LPs resp., to be solved. The value of the global lower bound is denoted by lb. Since we have a maximization problem this value corresponds to the best feasible solution. The global upper bound obtained by the maximal LP relaxation value of all open subproblems is denoted by ub. The last column is the value of the Hamming distance heuristic described in section 5.2.1. Figure 7.1 shows the whole hybridization matrix where the columns are permuted according to the optimum solution of the Hamming distance TSP. The objective function value corresponding to the arising WPEP is 103910. This Hamming distance approach is widely used by biologists in practice (see [AKWZ94, AKNW95, GI95, Heb01]). But in contrast to these approaches we solve the arising TSP problems exactly with the help of the branch-and-cut code from Jünger et al. [JRT94, JRR95].

On the one hand the computational results show that huge problems take very long to be solved to optimality. This can be done only for problems up to 60 columns in a

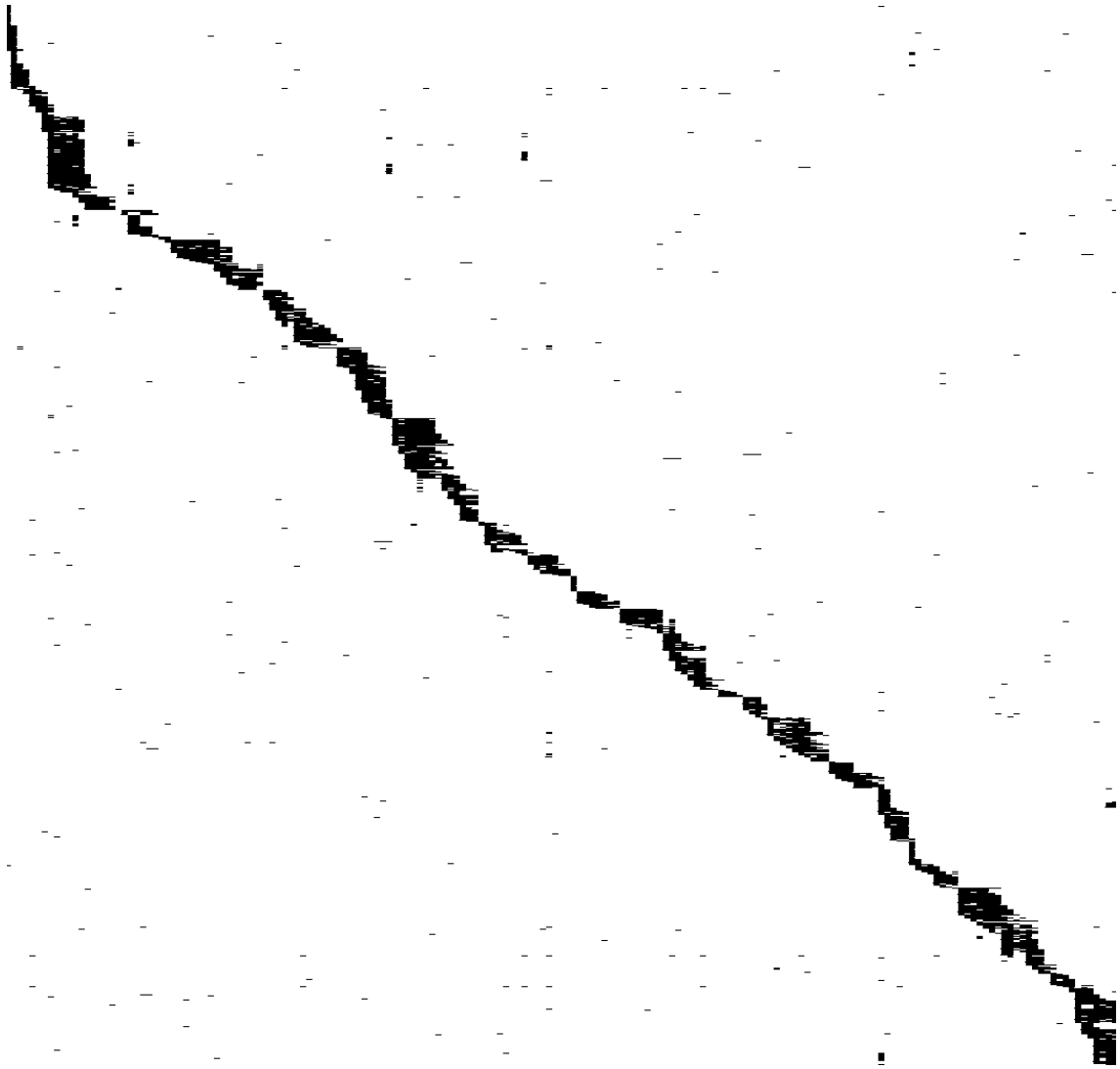


Figure 7.1: Hamming distance solution

reasonable amount of time. But on the other hand the rounding heuristic works very well. In all cases the optimum solution, or at least a solution clearly dominating the Hamming distance value, was found very early in the branch-and-cut process. Here "dominating" means that, assuming the correctness of the mathematical model of section 6.1, the best solutions we found have a higher probability to be the real hybridization matrices than the Hamming distance ones. Figure 7.2 displays the hybridization matrix permuted along the best known solution with an objective function value of 104573.

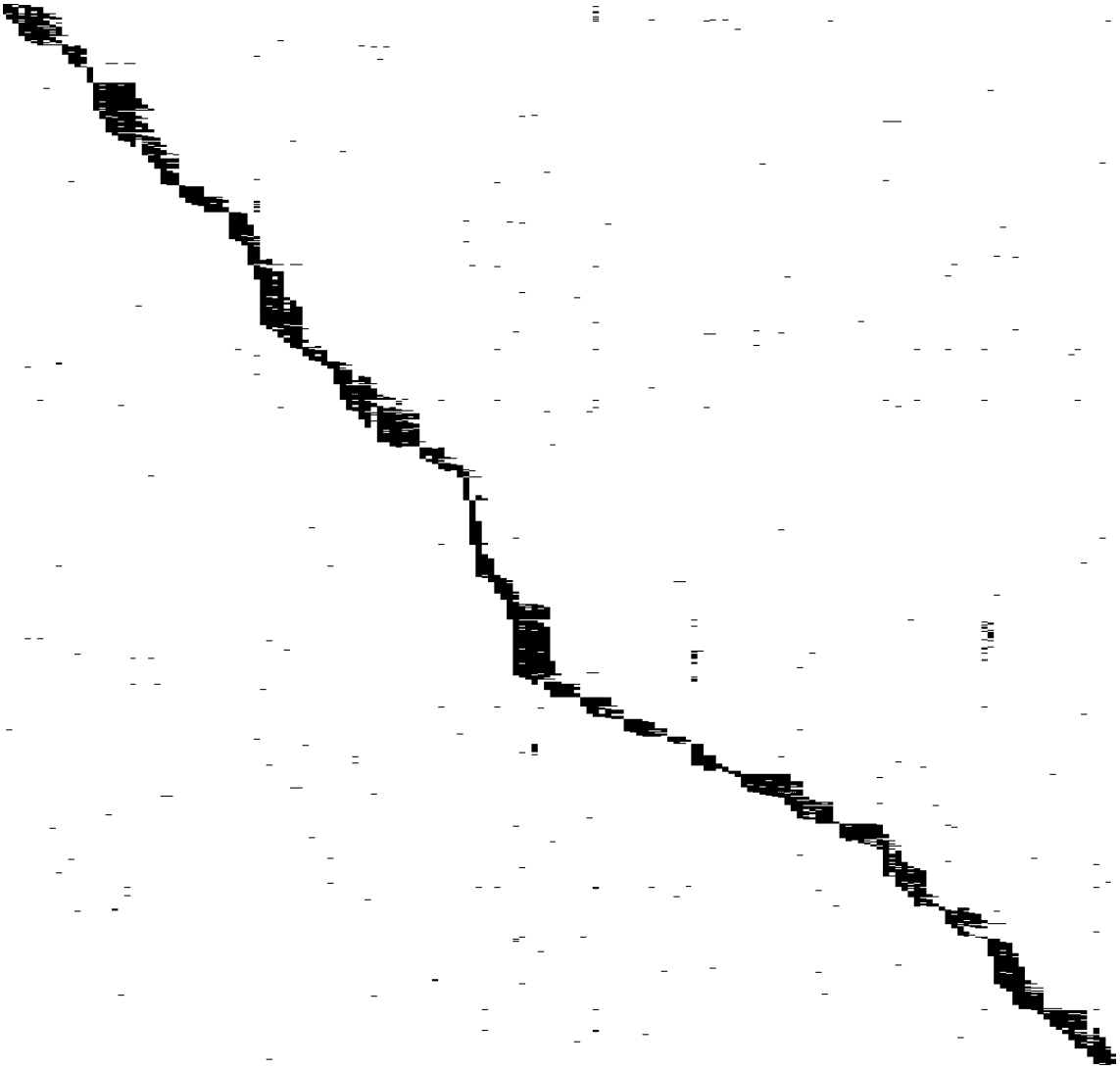


Figure 7.2: Best known solution

7.2 Random matrices

For a given randomly generated (n, n) binary matrix we are interested in the expected value of the minimal number of entries that have to be switched to obtain a C1PR matrix. Or in other words we want to investigate, how far a random 0/1 matrix is away from being C1PR. In addition we want to compare the efficiency of the SIR separation and the pattern based separation procedure, both described in chapter 5.

To this end we made the following computational tests. We generated binary (n, n)-matrices for $n = 5, \ldots, 19$ at random with densities of the "1"-entries varying between 10% and 90%. For every problem we computed the minimal number of entries to be switched to obtain a C1PR matrix. All problems are solved twice, first using the SIR

separation and second using the pattern separation. Staircase inequalities are separated in both approaches.

For every size and density we created 10 instances. Up to $n = 12$ all instances could be solved in reasonable time with both the SIR and the pattern approach. Larger problems turned out to be more difficult depending on their density. Table 7.2 displays the average percentage of entries that have to be switched for the various problem sizes. The most entries apparently have to be switched for densities around 60%. These problems also are the most difficult ones for our branch-and-cut algorithm.

	10%	20%	30%	40%	50%	60%	70%	80%	90%
$(5,5)$	0.0	0.0	1.2	2.0	0.8	0.8	0.4	0.8	0.0
$(6,6)$	0.0	0.3	2.5	2.2	1.9	3.3	2.5	1.9	0.6
$(7,7)$	0.2	0.6	3.1	3.7	4.9	5.9	4.9	2.9	0.6
$(8,8)$	0.0	0.3	1.6	3.9	5.2	7.5	6.3	4.4	2.7
$(9,9)$	0.1	1.1	3.6	5.4	6.3	7.4	5.9	5.2	2.2
$(10,10)$	0.3	1.9	4.3	6.2	7.4	8.4	8.0	6.6	2.7
$(11,11)$	0.2	2.5	5.1	6.9	9.6	10.1	9.3	6.9	3.7
$(12,12)$	0.4	3.2	6.0	8.5	10.1	11.7	10.7	8.8	5.0
$(13,13)$	0.4	3.6	6.3	11.1		11.1	10.5	8.2	4.4
$(14,14)$	0.6	4.3					11.3	8.8	4.8
$(15,15)$	0.8	4.1						9.6	5.3
$(16,16)$	0.9	4.6						9.3	5.3
$(17,17)$	0.9								5.6
$(18,18)$	1.4								5.3
$(19,19)$	1.5								6.0

Table 7.2: Percentage of entries to be switched.

Tables 7.3 and 7.4 show the average computation times for $n \leq 19$. Only problem series that were solved within 24 hours are displayed. For example, for $n = 14$ and density 50% it can even take several days to solve one problem.

The variance of CPU times for problems of the same size and density is very high. For example, for the problems of size $(12, 12)$ and density 50% the fastest computation took 34 seconds and the slowest 76 minutes using the SIR separation. For the pattern-based approach the gap is even more remarkable, it took 6 seconds in the fastest and 70 minutes in the slowest case.

The comparison of tables 7.3 and 7.4 show that the average execution times for the pattern-based runs are almost always faster than for the SIR runs. The speed up is significant, especially for problems with a high density of 90%. Without illustration we want to mention that the number of branch-and-bound nodes decreases in a similar way. In the next section we will see that the separation of patterns provides us an exact separation procedure for inequalities with at most 4 rows or columns. On the other hand in the SIR

	10%	20%	30%	40%	50%	60%	70%	80%	90%
(5, 5)	0:01	0:01	0:01	0:01	0:01	0:01	0:01	0:01	0:01
(6, 6)	0:01	0:01	0:01	0:01	0:01	0:01	0:01	0:01	0:01
(7, 7)	0:01	0:01	0:01	0:01	0:02	0:02	0:01	0:01	0:01
(8, 8)	0:01	0:01	0:01	0:01	0:02	0:04	0:02	0:02	0:01
(9, 9)	0:01	0:01	0:05	0:05	0:06	0:07	0:03	0:02	0:01
(10, 10)	0:01	0:01	0:21	0:21	0:23	0:39	0:16	0:16	0:01
(11, 11)	0:02	0:07	2:26	9:14	8:01	7:10	3:07	0:15	0:02
(12, 12)	0:02	0:09	4:21	25:26	28:16		12:29	4:23	0:13
(13, 13)	0:02	1:15	48:00				63:04	5:55	0:22
(14, 14)	0:02	13:34						17:16	0:45
(15, 15)	0:02	10:21						84:33	3:04
(16, 16)	0:03								5:52
(17, 17)	0:11								18:47
(18, 18)	1:01								21:06
(19, 19)	1:26								66:57

Table 7.3: Average execution time over 10 instances using SIR separation.

	10%	20%	30%	40%	50%	60%	70%	80%	90%
(5, 5)	0:01	0:01	0:01	0:01	0:01	0:01	0:01	0:01	0:01
(6, 6)	0:01	0:01	0:01	0:01	0:01	0:01	0:01	0:01	0:01
(7, 7)	0:01	0:01	0:01	0:01	0:01	0:01	0:01	0:01	0:01
(8, 8)	0:01	0:01	0:01	0:01	0:01	0:02	0:01	0:01	0:01
(9, 9)	0:01	0:01	0:02	0:03	0:05	0:03	0:02	0:01	0:01
(10, 10)	0:01	0:01	0:17	0:14	0:16	0:27	0:17	0:06	0:01
(11, 11)	0:01	0:03	0:48	3:21	4:45	12:53	1:53	0:03	0:02
(12, 12)	0:01	0:04	1:44	16:40	23:20		11:07	3:16	0:06
(13, 13)	0:02	0:31	24:21				45:36	3:28	0:05
(14, 14)	0:02	3:21						7:05	0:06
(15, 15)	0:02	4:37						76:53	0:46
(16, 16)	0:03								1:06
(17, 17)	0:08								13:18
(18, 18)	0:41								3:46
(19, 19)	1:46								10:10

Table 7.4: Average execution time using pattern-based separation

separation facet classes of these sizes are only separated heuristically by solving a quadratic assignment problem. Supposedly this is the reason for the speed up.

Table 7.5 and table 7.6 display some statistics about the number and type of generated cuts of both branch-and-cut runs for a $(19, 19)$-matrix with density 10%. The running time was about 2 minutes for both approaches. 113 LPs had to be solved in 45 branch-and-bound nodes in the SIR case and 31 subproblems with 93 LPs in the pattern-based case.

(m, n)	3	4	5	6
3	2	29	0	14
4	0	8	15	176
5	0	11	0	0

k	1	2	3	4	5	6	7	8
F_{1_k}	13	62	77	62	34	16	3	2
F_{2_k}	93	253	103	21	2	0	0	0

Table 7.5: Number and types of generated cuts in the SIR case.

The right-hand side of table 7.5 shows the number of generated staircase cuts, the left-hand side the number of SIR-cuts of size (m, n). This separation of inequalities from small instances proved to be very helpful. Many of them are identified quickly by our heuristic. But, as table 7.6 shows, the pattern principle is even more effective.

(m, n)	3	4	5	6	7	8
3	35	71	15	3	0	0
4	14	26	29	133	21	6
5	0	2	8	9	0	0
6	0	0	2	3	0	0
7	0	0	1	1	0	0

k	1	2	3	4	5	6	7
F_{1_k}	62	111	108	80	81	6	0
F_{2_k}	492	330	159	50	23	8	0

Table 7.6: Number and types of generated cuts in the pattern case.

The left-hand side displays the number of violated inequalities of size (m, n) constructed from a violated pattern-based inequality (see section 5.3.3). Note that in contrast to the SIR approach not all of these cutting planes are facet-defining. But the number and variety of different sizes is definitely greater than in the previous figure. Furthermore also the number of generated staircase cuts on the right-hand side exceeds the first approach. Probably this is because the LP structure is modified by the different pattern-based cuts in such a way that the staircase separation works more effectively.

At last we want to mention the primal heuristic described in section 5.2.2, which turned out to be very good. Considering the average time over both approaches, it took only about 6 minutes to find the optimal solutions for the $(12, 12)$-matrices with density 50%. But proving optimality took additional 20 minutes. For 8 of the 20 instances the optimal primal bound was found within 6 seconds, in 13 cases within 1 minute.

7.3 Problems with few columns or rows

In section 3.6 we have shown that the number of facets of $P^{m,n}_{\text{C1R}}$ is polynomial in n if m is fixed and polynomial in m if n is fixed. The WC1P is even solvable in linear time for a fixed number of columns or rows but as also shown in 3.6 the constant is very huge.

In this section we want to verify that also our branch-and-cut code shows a "good behavior" in solving these kind of problems. To this end we first generated random objective function matrices for the the WC1P with $3-7$ columns, doubling the number of rows step by step from 10 to 2560. The values of the objective function are uniformly distributed from -10 to 10. Since the separation by patterns is much more adequate for solving such problems than the staircase separation procedure, we dropped down the latter one for the purpose of saving time. Table 7.7 shows the average execution times and the average number of created subproblems over 10 instances. This data is only displayed for problem series that were solved within 24 hours.

(m,n)	3	4	5	6	7
10	2	2	2	5	7
20	2	2	2	4	29
40	2	3	3	5	394
80	3	3	5	12	460
160	4	7	16	533	
320	13	26	100		
640	51	208	1273		
1280	250	1790			
2560	1275				

(m,n)	3	4	5	6	7
10	1.0	1.0	1.0	1.0	1.4
20	1.0	1.0	1.0	1.0	1.6
40	1.0	1.0	1.0	1.2	3.2
80	1.0	1.0	1.2	4.6	5.0
160	1.0	1.0	2.0	13.2	
320	1.0	1.0	3.8		
640	1.0	1.0	12.6		
1280	1.0	1.0			
2560	1.0				

Table 7.7: Average execution time in seconds and average number of subproblems over 10 instances with small n

The first striking observation is that all problems with 3 or 4 columns were solved at the root node of the branch-and-cut tree. This indicates that the separation procedure based on the row-patterns of length 3 and 4 is an exact one in the sense that all violated inequalities of sizes $(m,3)$ and $(m,4)$ were detected. Unfortunately this does not hold for row-patterns of length 5 and 6. One of very few facet-defining inequalities of size $(5,5)$ that were neither separated by row patterns nor by column patterns is

$$\begin{pmatrix} 1 & -1 & 1 & -1 & 1 \\ 1 & 1 & 0 & -1 & 0 \\ 0 & -1 & 1 & 1 & -1 \\ 0 & -1 & -1 & 1 & 1 \\ -1 & 1 & 0 & 1 & 0 \end{pmatrix} \circ x \leq 9.$$

Being compatible with the theory the running times show a polynomial behavior for $n \leq 4$. For $n = 3$ the quotients of the running times of two consecutive problem sizes

seem to be bounded by about 5. This indicates a polynomial running time of $O(m^{\frac{\log(5)}{\log(2)}}) \approx O(m^{2.3})$. Similarly for $n = 4$, in this case 9 is such a bound, leading to a experimentally derived time complexity of about $O(m^{3.2})$.

Problems where the number of columns is fixed to a value greater than 4 were more difficult to solve. Because the pattern-based separation provides no exact separation procedure in this case, the number of subproblems to be solved increase significantly, and so do the total running times. Consequently no polynomial behavior is visible for $n \geq 5$.

Now we consider problems with few number of rows. The number of rows varies from 3 to 6, the number of columns from 10 to 640. As in the previous experiments for each problem size 10 objective function matrices are created at random with entries uniformly distributed from -10 to 10.

The computational results are displayed in table 7.8. Again we only considered problem series with a total running time less than 24 hours.

(m, n)	10	20	40	80	160	320	640
3	2	2	3	7	65	408	3317
4	2	3	6	22	213	2188	
5	3	12	41	443			
6	4	11	95				

(m, n)	10	20	40	80	160	320	640
3	1.0	1.0	1.0	1.0	1.0	1.0	1.0
4	1.0	1.0	1.0	1.0	1.0	1.0	
5	1.0	1.2	1.2	8.6			
6	1.0	1.4	3.2				

Table 7.8: Average execution time in seconds and average number of subproblems over 10 instances with small m

Both the table of the running times and the table of the number of subproblems to be solved shows similar results to the previous case. Again all problems with at most 4 rows were solved to optimality without any branching whereas for solving problems with 5 or 6 rows more and more subproblems were required. With similar arguments to the previous case we can estimate the running times to grow approximatively like $O(n^3)$ for $m = 3$ and approximatively like $O(n^{3.4})$ for $m = 4$. But note that for all problems discussed in this section theorem 3.38 even provides a linear time algorithm. The main purpose for using our branch-and-cut code to solve these problems is to demonstrate the ability of the separation by patterns which provides an exact separation procedure for violated inequalities with not more than 4 rows or columns.

7.4 Computing clusters of crystal structure types

According to our model described in section 6.4 we want to present some computational results arising in determining clusters of crystal structure types. As input we used 4 distance matrices of different sizes obtained from Bergerhoff [Ber99]. The problems are named "167EB", "136FA", "62DC5", and "62DC4A" and their sizes vary from $(28, 28)$ to $(162, 162)$. The smallest problem "167EB" has already been discussed in section 6.4 using a bound of 0.100.

We solved each problem using 4 different bounds from 0.025 up to 0.200. Figure 7.3 displays the computational results.

name	size	$\overline{d}$	b	t_{tot}	t_{opt}	n_{sub}	n_{lp}	n_{pat}	n_{stc}
167EB	$(28, 28)$	0.083	0.025	0:05	0:05	1	2	109	718
			0.050	0:05	0:02	1	2	124	1753
			0.100	0:16	0:09	1	3	822	5364
			0.200	0:02	0:02	1	1	0	0
136FA	$(54, 54)$	0.088	0.025	1:42	1:42	1	5	2416	10728
			0.050	1:31	1:31	1	2	683	2682
			0.100	0:02	0:02	1	1	0	0
			0.200	0:02	0:02	1	1	0	0
62DC5	$(85, 85)$	0.095	0.025	0:14	0:13	1	2	14	29
			0.050	64:01	52:50	5	76	6135	85120
			0.100	5:55	5:54	1	11	5544	26820
			0.200	2:31	2:30	1	7	546	14751
62DC4A	$(162, 162)$	0.028	0.025	209:24	209:22	13	60	4831	107467
			0.050	133:33	133:08	3	40	13731	95515
			0.100	828:56	827:37	43	233	24767	459340
			0.200	6:25	6:24	1	2	10	56

Figure 7.3: Statistics on WSC1P runs arising in cluster finding of crystal structure types. All times are given in minutes:seconds

$\overline{d}$ denotes the average distance value in the matrix and b the used bound. The objective function value for entry ij is $c_{ij} = d_{ij} - b$. The total running time of the branch-and-cut algorithm is denoted by t_{tot} and t_{opt} is the time required to find the optimum solution. n_{sub}, n_{lp} resp., are the number of subproblems, LPs resp., to be solved and the number of generated pattern-based cuts, staircase cuts resp., is denoted by n_{pat}, by n_{stc} resp.

The computational results for these WSC1P problems were somehow contrary to the physical mapping WC1P runs in section 7.1. One the one hand, up to two exceptions the optimal solutions were always found by feasible LP solutions. As already mentioned in section 5.2.2 this is because the rounding heuristic does not work well in the simultaneous case, since the weighted problem remains $\mathcal{NP}$-hard for fixed row and column permutation

(see section 4.2.3). On the other hand, in spite of the worse primal heuristic even problems of size $(162, 162)$ could be solved to optimality. This fact qualifies the WSC1P approach as a possibility for determining clusters of a given distance matrix. As shown in section 6.4 the optimal solutions were very similar to the expected clusters.

The huge number of generated cutting planes show that both the pattern-based and the staircase separation procedure works also efficiently in the simultaneous case. As expected, problems with a big difference of $\overline{d}$ and b are easy to solve and require only few cutting planes. For some of the problems even the initial LP solution is feasible. Take for example the problem "167EB" displayed in figure 6.4 with a bound of 0.200 (note that the entries in the matrix are multiplied by 100). If all entries less than 0.200 are set to "1" and the other entries set to "0", the resulting matrix is already C1PS and therefore the branch-and-cut process terminates after the first solved LP.

Chapter 8

Discussion

In this work we have made intensive investigations on weighted consecutive ones problems with the aim of both getting more theoretical knowledge and developing an efficient branch-and-cut algorithm solving the $\mathcal{NP}$-hard WC1P to optimality.

For these purposes we first addressed the facial structure of the associated polytope. Besides showing useful properties, such as lifting and melting of valid inequalities, we constructed an IP formulation of the WC1P consisting only of facet-defining inequalities for the corresponding polytope $P^{m,n}_{\mathrm{C1R}}$. We first used a characterization of Tucker to derive a weak IP formulation, after that the inequalities were strengthened to obtain facets. This strengthened formulation served as a basis for our branch-and-cut code, which turned out to be very efficient, since we could show later on that the separation problem for this IP formulation can be solved in polynomial time.

We also showed that there is a close relationship between the consecutive ones and the betweenness problem in the sense that we can derive facets for $P^{m,n}_{\mathrm{C1R}}$ from all non-trivial facets of the betweenness polytope P^{n}_{BW}. By generalizing this method and introducing the concept of maximal feasible sets of columns and rows we were able to show that the number of facets of $P^{m,n}_{\mathrm{C1R}}$ grows only polynomially in m if n is fixed and grows polynomially in n if m is fixed with the consequence that for these cases the WC1P can be solved in polynomial time. This result is surprising since it contrasts with many combinatorial optimization problems which are polynomially solvable, but the corresponding polytope of which has an exponential number of facets.

The following part of the work is dedicated to the simultaneous consecutive ones problem. Here we dealt with matrices that have both the consecutive ones property for rows and for columns. Most of the complexity results concerning the standard problem can be transferred to the simultaneous problem. In both cases the weighted problems are $\mathcal{NP}$-hard in general and become polynomially solvable if the number of rows or the number of columns is fixed. But a remarkable difference arises if we consider the weighted problems with fixed column and row permutations. The WC1P becomes linearly solvable in this case, whereas the WSC1P remains $\mathcal{NP}$-hard. For the latter we proved the $\mathcal{NP}$-completeness of a suitable augmentation version of the decision problem by a transformation to the Hamiltonian path problem.

Polyhedral investigations of the simultaneous polytope $P_{\mathrm{C1S}}^{m,n}$ results similar to the standard polytope. Most of the properties could be transferred though their proofs turned out to be harder. In this context the staircase inequalities have to be mentioned. They were shown to be facet-defining also in the simultaneous case and therefore provided an IP formulation consisting of facets only.

In the following we addressed the question of how the theoretical knowledge obtained before could be used for an efficient branch-and-cut approach for both the WC1P and the WSC1P. The fact that the WC1P is linearly solvable for a fixed column permutation provided us with a very effective primal heuristic. Suitable candidates for this column permutation can be obtained by a heuristic based on the PQ-tree algorithm performed on a rounded LP solution. Computational results have shown that usually this primal heuristic finds the optimal solution early in the branch-and-cut process, long before its optimality has been proven. Unfortunately this heuristic cannot be transferred to the simultaneous problem since the WSC1P for fixed column and row permutation remains $\mathcal{NP}$-hard. We tried a modified version of the WC1P heuristic and also implemented a simulated annealing heuristic for the WSC1P, but the results of both approaches were not good in the sense that in most of the branch-and-cut experiments there was a big gap between the best solution found by the primal heuristic and the optimal solution. Therefore a development of a new effective WSC1P heuristic remains to be done in future research.

Furthermore we presented polynomial separation procedures for the two classes of staircase inequalities which play the most important role in the IP formulation of both the WC1P and the WSC1P. Since the running times of both separation algorithms are highly dependent on the number of columns, we had to restrict ourselves to problems up to about 90 columns using the exact separation. For bigger problems we worked on submatrices of that size that were chosen heuristically.

In addition we wanted to make use of the fact, that the complexity of both the WC1P and the WSC1P is polynomial if the number of rows or columns is fixed. Therefore we introduced a separation procedure based on the concept of patterns of a fixed length. The computational results have shown that for a pattern length up to 4 all violated inequalities are found by this procedure. Also for higher pattern lengths the separation procedure worked very efficiently, though it provided no exact separation in this case. Closing this gap by introducing new patterns of length 5 and 6 as well as proving theoretically that all facets of the corresponding size are separated remains an interesting subject for further research.

Finally we have shown that our branch-and-cut code provides a useful tool for tackling WC1P and WSC1P problems occurring in practice. Although we were not able to solve real world physical-mapping problems to optimality, our solutions obtained by the rounding heuristic clearly outperform the optimal solutions of the Hamming distance TSP approach which is widely used in computational biology and, moreover, solved only approximately there. WSC1P problems occurring in analyzing clusters of crystal structures showed a slightly different behavior. The primal heuristic usually gave no good solutions, nevertheless the problems could be solved to optimality and the results were very similar to the ones expected in crystallography.

Appendix A

Outer Descriptions of Polytopes

A.1 Complete description of $P^{3,6}_{\mathrm{C1R}}$

$$x_{ij} \geq 0 \qquad\qquad x_{ij} \leq 1$$

$$\begin{pmatrix} 1 & 1 & -1 \\ 1 & -1 & 1 \\ -1 & 1 & 1 \end{pmatrix} \circ x \leq 5$$

$$\begin{pmatrix} 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \circ x \leq 5$$

$$\begin{pmatrix} 1 & 1 & 1 & 0 & -1 & -1 \\ 1 & 0 & -1 & 1 & 1 & -1 \\ 0 & 1 & -1 & 1 & -1 & 1 \end{pmatrix} \circ x \leq 8$$

$$\begin{pmatrix} 1 & 1 & 1 & 0 & -1 & -1 \\ 1 & 0 & -1 & 1 & 1 & -1 \\ 0 & -1 & 1 & -1 & 1 & 1 \end{pmatrix} \circ x \leq 8$$

$$\begin{pmatrix} 1 & 1 & 1 & 0 & -1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 0 \\ 1 & -1 & -1 & 1 & -1 & 1 \end{pmatrix} \circ x \leq 8$$

A.2 Complete description of $P^{4,4}_{\mathrm{C1R}}$

$$x_{ij} \geq 0 \qquad x_{ij} \leq 1$$

$$\begin{pmatrix} 1 & 1 & -1 \\ 1 & -1 & 1 \\ -1 & 1 & 1 \end{pmatrix} \circ x \leq 5$$

$$\begin{pmatrix} 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \circ x \leq 5$$

$$\begin{pmatrix} 1 & 1 & 0 & -1 \\ 1 & 0 & 1 & -1 \\ 0 & 1 & -1 & 1 \\ 0 & -1 & 1 & 1 \end{pmatrix} \circ x \leq 7$$

$$\begin{pmatrix} 1 & 1 & 0 & -1 \\ 1 & 1 & -1 & 0 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \circ x \leq 7$$

$$\begin{pmatrix} 1 & 1 & 0 & -1 \\ 1 & 0 & -1 & 1 \\ -1 & 1 & 1 & -1 \\ -1 & 1 & 0 & 1 \end{pmatrix} \circ x \leq 7$$

$$\begin{pmatrix} 1 & 1 & 1 & -1 \\ 1 & -1 & -1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & -1 & 1 & 1 \end{pmatrix} \circ x \leq 7$$

$$\begin{pmatrix} 1 & 1 & 1 & -1 \\ 1 & 0 & -1 & 1 \\ 0 & -1 & 1 & 1 \\ -1 & 1 & 0 & 1 \end{pmatrix} \circ x \leq 8$$

$$\begin{pmatrix} 1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 \\ 1 & -1 & 1 & 1 \\ -1 & 1 & 1 & 1 \end{pmatrix} \circ x \leq 10$$

$$\begin{pmatrix} 2 & 2 & -1 & -2 \\ 2 & -1 & 2 & -2 \\ 2 & -2 & -2 & 2 \\ -2 & 1 & 1 & 2 \end{pmatrix} \circ x \leq 13$$

$$\begin{pmatrix} 2 & 2 & 2 & -2 \\ 1 & 1 & -2 & 2 \\ 1 & -2 & 1 & 2 \\ -2 & 1 & 1 & 2 \end{pmatrix} \circ x \leq 15$$

$$\begin{pmatrix} 2 & 2 & 2 & -2 \\ 2 & -1 & -1 & 2 \\ -1 & 2 & -1 & 2 \\ -1 & -1 & 2 & 2 \end{pmatrix} \circ x \leq 15$$

A.3 Complete description of $P^{4,4}_{\text{C1S}}$

$x_{ij} \geq 0$

$$\begin{pmatrix} 1 & 1 & -1 \\ 1 & -1 & 1 \\ -1 & 1 & 1 \end{pmatrix} \circ x \leq 5$$

$$\begin{pmatrix} 1 & 1 & 0 & -1 \\ 1 & 1 & -1 & 0 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \circ x \leq 7$$

$$\begin{pmatrix} 1 & 1 & 1 & -1 \\ 1 & -1 & -1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & -1 & 1 & 1 \end{pmatrix} \circ x \leq 7$$

$$\begin{pmatrix} 1 & 1 & 1 & -1 \\ 1 & 0 & -1 & 1 \\ 0 & -1 & 1 & 1 \\ -1 & 1 & 0 & 1 \end{pmatrix} \circ x \leq 8$$

$$\begin{pmatrix} -1 & -1 & 1 & 1 \\ -1 & 2 & 1 & 1 \\ 2 & -2 & 2 & -1 \\ 2 & 2 & -1 & -1 \end{pmatrix} \circ x \leq 12$$

$$\begin{pmatrix} 2 & 2 & 1 & -1 \\ -2 & -2 & 2 & 1 \\ -2 & 2 & -2 & 2 \\ 2 & -2 & -2 & 2 \end{pmatrix} \circ x \leq 13$$

$$\begin{pmatrix} 2 & 2 & 2 & -2 \\ 1 & 1 & -2 & 2 \\ 1 & -2 & 1 & 2 \\ -2 & 1 & 1 & 2 \end{pmatrix} \circ x \leq 15$$

$x_{ij} \leq 1$

$$\begin{pmatrix} 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \circ x \leq 5$$

$$\begin{pmatrix} 1 & 1 & 0 & -1 \\ 1 & 0 & -1 & 1 \\ -1 & 1 & 1 & -1 \\ -1 & 1 & 0 & 1 \end{pmatrix} \circ x \leq 7$$

$$\begin{pmatrix} 1 & 1 & 0 & -1 \\ 1 & 0 & 1 & -1 \\ 0 & 1 & -1 & 1 \\ 0 & -1 & 1 & 1 \end{pmatrix} \circ x \leq 7$$

$$\begin{pmatrix} 1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 \\ 1 & -1 & 1 & 1 \\ -1 & 1 & 1 & 1 \end{pmatrix} \circ x \leq 10$$

$$\begin{pmatrix} 2 & 2 & -1 & -2 \\ 2 & -1 & 2 & -2 \\ 2 & -2 & -2 & 2 \\ -2 & 1 & 1 & 2 \end{pmatrix} \circ x \leq 13$$

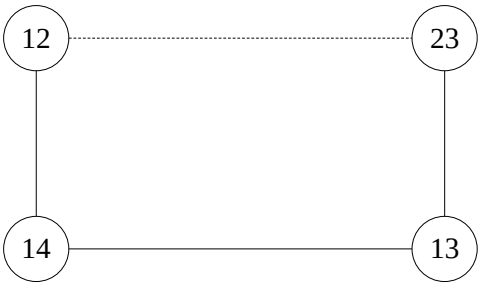
$$\begin{pmatrix} 2 & 2 & 2 & -2 \\ 2 & -1 & -1 & 2 \\ -1 & 2 & -1 & 2 \\ -1 & -1 & 2 & 2 \end{pmatrix} \circ x \leq 15$$

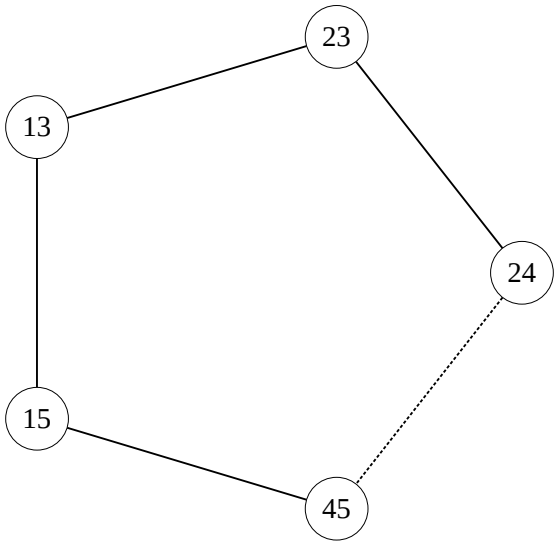
$$\begin{pmatrix} 2 & 3 & -1 & 2 \\ 3 & -3 & 3 & -2 \\ -2 & 3 & 3 & -1 \\ 1 & -1 & -2 & 2 \end{pmatrix} \circ x \leq 19$$

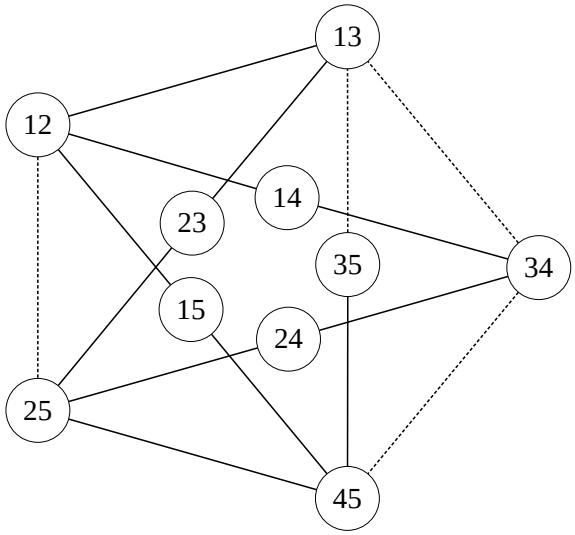
With an inequality $A \circ x \leq a_0$ also the transposed version $A^T \circ x \leq a_0$ is facet-defining.

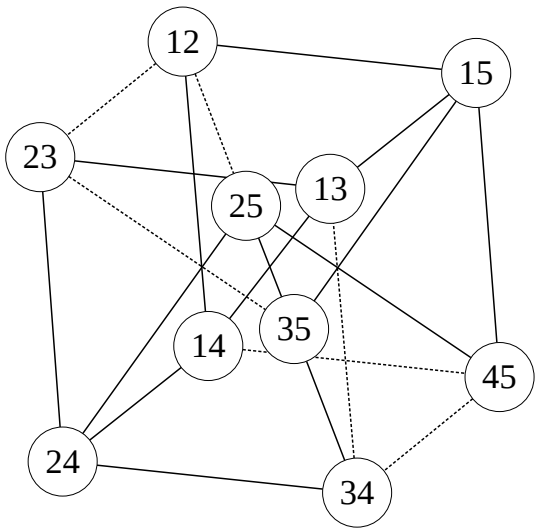
A.4 Complete description of P^5_{BW}

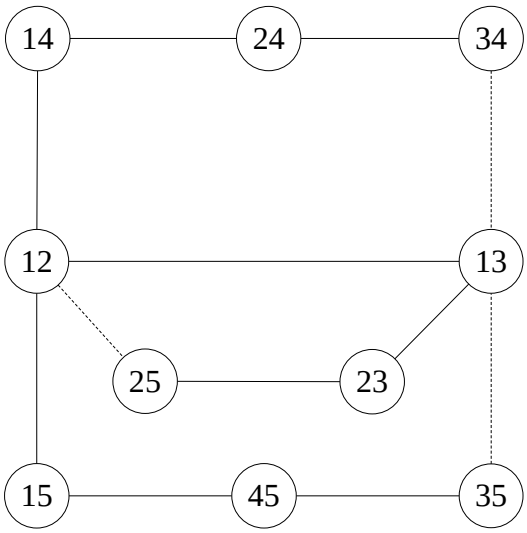
There are 11 classes of facet-defining inequalites for P^5_{BW}. Only the first also defines a facet for P^4_{BW}. Each of the following 11 graphs shows a representative of these classes in normal form (see definition 3.18). The way to construct the inequality from the graph is the following. If nodes ij and jk are connected by a dashed edge, the variable $x_{i(j)k}$ has coefficient 1. The same holds if ij and kj or ji and jk are connected by a solid edge. The right hand side of the inequality equals the number of dashed edges. For example the first graph corresponds to the inequality $x_{1(2)3} + x_{1(3)2} + x_{2(1)4} + x_{3(1)4} \geq 1$ and the last one to $x_{1(2)3} + x_{1(3)2} + x_{3(1)4} + x_{1(4)3} + x_{4(3)5} + x_{2(5)3} + x_{1(2)5} + x_{2(1)4} + x_{1(4)5} + x_{1(5)4} + x_{2(1)5} \geq 3$.

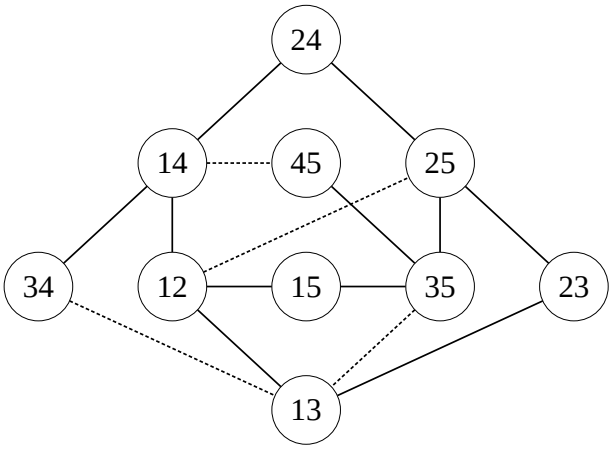


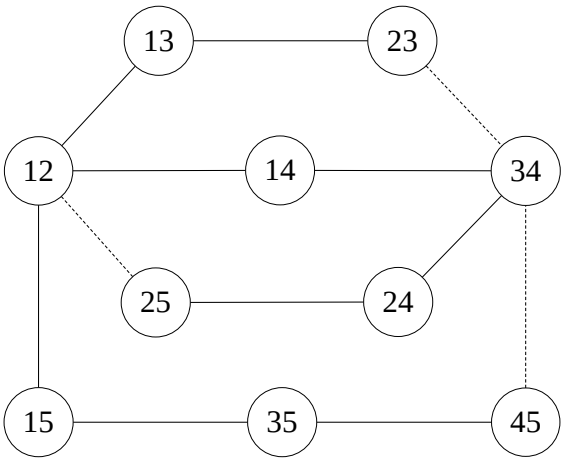


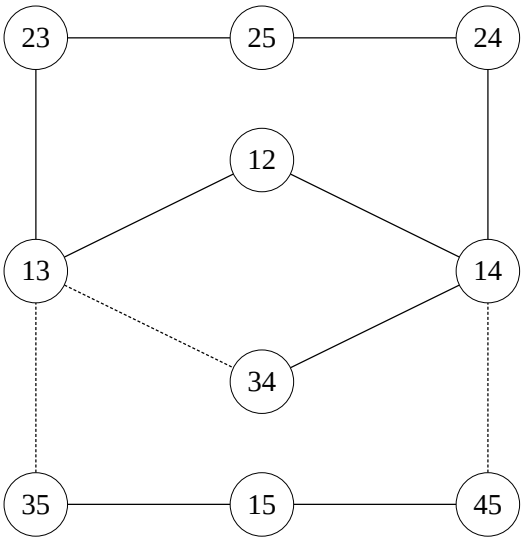


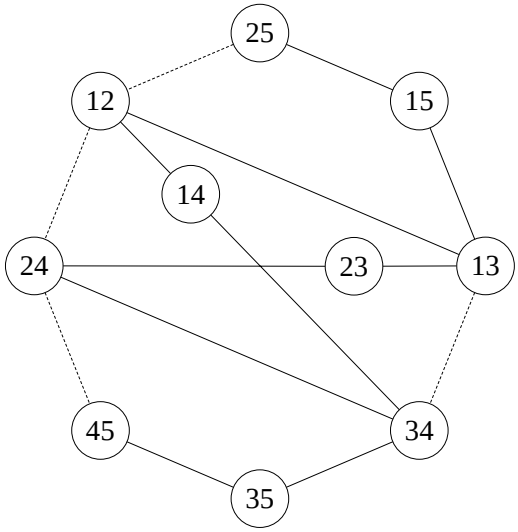

12
15
23
13
25
14
35
45
24
34

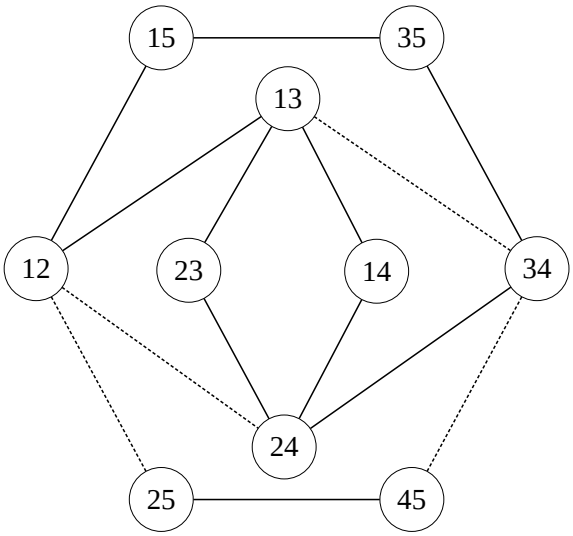

14
24
34
12
13
25
23
15
45
35

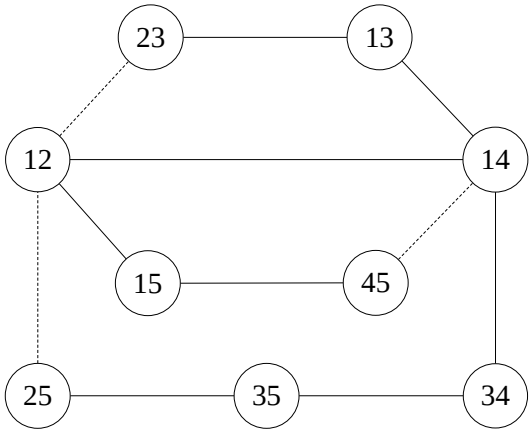

24
14
45
25
34
12
15
35
23
13


13
23
12
14
34
25
24
15
35
45


23
25
24
12
13
14
34
35
15
45


25
12
15
14
24
23
13
45
34
35


15
35
13
12
23
14
34
24
25
45


23
13
12
14
15
45
25
35
34

Bibliography

[ABCC01] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. TSP cuts which do not conform to the template paradigm. In M. Jünger and D. Naddef, editors, *Computational Combinatorial Optimization*, volume 2241 of *Lecture Notes in Computer Science*, pages 261–303. Springer, 2001.

[AKNW95] F. Alizadeh, R.M. Karp, L.A. Newberg, and D.K. Weisser. Physical mapping of chromosomes: A combinatorial problem in molecular biology. *Algorithmica*, 13(1/2):52–76, 1995.

[AKWZ94] F. Alizadeh, R.M. Karp, D.K. Weisser, and G. Zweig. Physical mapping of chromosomes using unique probes. In *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 489–500. ACM Press, 1994.

[BBBD99] G. Bergerhoff, M. Berndt, K. Brandenburg, and T. Degen. Concerning inorganic crystal structure types. *Acta Crystallographica Section B*, 55:147–156, 1999.

[Ber99] G. Bergerhoff. Personal communication, 1999.

[BFM98] R. Borndörfer, C.E. Ferreira, and A. Martin. Decomposing matrices into blocks. *SIAM Journal on Optimization*, 9:236–269, 1998.

[BL76] K. Booth and G. 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.

[Boo75] K.S. Booth. *PQ-Tree Algorithms.* PhD thesis, University of California, Berkeley, 1975.

[Chr97] T. Christof. *Low-Dimensional 0/1-Polytopes and Branch-and-Cut in Combinatorial Optimization.* PhD thesis, University of Heidelberg, 1997.

[Chv83] V. Chvátal. *Linear Programming.* Freeman, New York, 1983.

[CL98] T. Christof and A. Loebel. PORTA - a polyhedron representation algorithm, 1998. `www.informatik.uni-heidelberg.de/groups/comopt/software/PORTA/index.html`.

[CLR90] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms.* MIT Press, Cambridge, 1990.

[COR98] T. Christof, M. Oswald, and G. Reinelt. Consecutive ones and a betweenness problem in computational biology. In R.E. Bixby, E.A. Boyd, and R.Z. Ríos-Mercado, editors, *Integer Programming and Combinatorial Optimization*, pages 213–228, 1998. Proceedings of the 6th International IPCO Conference, Houston.

[Die00] R. Diestel. *Graphentheorie.* Springer, 2000.

[FCH+00] M. Frohme, A.A. Camargo, S. Heber, C. Zink, A. Simpson, J. Hoheisel, and A. de Souza. Mapping analysis of the xylella fastidiosa genome. *Nucleic Acids Research*, 28(16):3100–3104, 2000.

[FG65] D.R. Fulkerson and O.A. Gross. Incidence matrices and interval graphs. *Pacific Journal of Mathematics*, 15:835–855, 1965.

[Fis02] G. Fischer. *Lineare Algebra.* Vieweg, 2002.

[FJF58] L.R. Ford Jr. and D.R. Fulkerson. A suggested computation for maximal multicommodity network flows. *Management Science*, 5:97–101, 1958.

[Fro00] M. Frohme. Personal communication, 2000.

[GI95] D.S. Greenberg and S. Istrail. Physical mapping by STS hybridization: Algorithmic strategies and the challenge of software evaluation. *Journal of Computational Biology*, 2(2):219–273, 1995.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Freeman, San Francisco, 1979.

[GJR84] M. Grötschel, M. Jünger, and G. Reinelt. A cutting plane algorithm for the linear ordering problem. *Operations Research*, 32:1195–1220, 1984.

[GJS76] M.R. Garey, D.S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical computer science*, 1:237–267, 1976.

[GLS93] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics.* Springer, 1993.

[Heb01] S. Heber. *Algorithms for Physical Mapping.* PhD thesis, University of Heidelberg, 2001.

[Jew58] W.S. Jewell. Optimal flow through networks. Technical Report 8, Massachusetts Institute of Technology, 1958.

[JRR95] M. Jünger, G. Reinelt, and G. Rinaldi. The traveling salesman problem. In M. Ball, T. Magnanti, C.L. Monma, and G.L. Nemhauser, editors, *Handbooks in Operations Research and Management Sciences*, volume 7: Network Models. North-Holland, 1995.

[JRT94] M. Jünger, G. Reinelt, and S. Thienel. Provably good solutions for the traveling salesman problem. *Zeitschrift für Operations Research*, 40:183–217, 1994.

[JRT95] M. Jünger, G. Reinelt, and S. Thienel. Practical problem solving with cutting plane algorithms in combinatorial optimization. In W. Cook, L. Lovász, and P. Seymour, editors, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Volume 20: Combinatorial Optimization*. American Mathematical Society, 1995.

[JT97a] M. Jünger and S. Thienel. The design of the branch-and-cut system ABACUS. Technical Report 95.260, Universität zu Köln, Germany, 1997.

[JT97b] M. Jünger and S. Thienel. Introduction to ABACUS - A Branch-And-CUt system. Technical Report 95.263, Universität zu Köln, Germany, 1997.

[Ken69] D.G. Kendall. Incidence matrices, interval graphs and seriation in archaeology. *Pacific Journal of Mathematics*, 28(3):565–570, 1969.

[KV00] B. Korte and J. Vygen. *Combinatorial Optimization*, volume 21 of *Algorithms and Combinatorics*. Springer, 2000.

[Lei97] S. Leipert. PQ-trees, an implementation as template class in C++. Technical Report 97.259, Universität zu Köln, Germany, 1997.

[LPR94] Y. Li, P.M. Pardalos, and M.G.C. Resende. A greedy randomized adaptive search procedure for the quadratic assignment problem. In P.M. Pardalos and H. Wolkowicz, editors, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Volume 16*, pages 237–261. American Mathematical Society, 1994.

[Min96] H. Minkowski. *Geometrie der Zahlen*. Leipzig: Teubner, 1896.

[Pad95] M. Padberg. *Linear Optimization and Extensions*, volume 12 of *Algorithms and Combinatorics*. Springer, 1995.

[Pap76] Ch.H. Papadimitriou. The NP-completeness of the bandwidth minimization problem. *Computing*, 16:263–270, 1976.

[Poc03] Y. Pochet. Personal communication, 2003.

[PR87] M.W. Padberg and G. Rinaldi. Optimization of a 532 city symmetric traveling salesman problem by branch and cut. *Operations Research Letters*, 6:1–7, 1987.

[PR91] M.W. Padberg and G. Rinaldi. A branch and cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, 33:60–100, 1991.

[PRW94] P.M. Pardalos, F. Rendl, and H. Wolkowicz. The quadratic assignment problem: A survey and recent developments. In P.M. Pardalos and H. Wolkowicz, editors, *DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Volume 16*. American Mathematical Society, 1994.

[Pul89] W.R. Pulleyblank. Polyhedral combinatorics. In G.L. Nemhauser, A.H.G. Rinnoy Kan, and M.J. Todd, editors, *Handbooks in Operations Research and Management Sciences: Optimization*. North-Holland, 1989.

[Rei91] G. Reinelt. TSPLIB - a traveling salesman problem library. *ORSA Journal on Computing*, 3:376–384, 1991. An updated version is available at `www.informatik.uni-heidelberg.de/groups/comopt/software/TSPLIB95/index.html`.

[Rob76] F.S. Roberts. *Discrete Mathematical Models with Applications to Social, Biological, and Environmental Problems.* Prentice-Hall, Englewood Cliffs, NJ, 1976.

[Sch86] A. Schrijver. *Theory of Linear and Integer Programming.* Chichester: John Wiley & Sons, 1986.

[SSdS+00] M.L. Silvestri, W.J. Siqueira, A.A. de Souza, A.P. de Souza, M.F. Terenzi, D. Truffi, S.M. Tsai, M.H. Tsuhako, H. Vallada, M.A. Van Sluys, S. Verjovski-Almeida, A.L. Vettore, M.A. Zago, M. Zatz, J. Meidanis, and J.C. Setubai. The genome sequence of the plant pathogen xylella fastidiosa. *Nature*, 406:151–157, 2000.

[Thi95] S. Thienel. *ABACUS A Branch-And-CUt System.* PhD thesis, Universität zu Köln, 1995.

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

[Wey35] H. Weyl. Elementare Theorie der konvexen Polyeder. *Commentari Mathematici Helvetici*, 7:290–306, 1935.

[Zie95] G. Ziegler. *Lectures on Polytopes*, volume 152 of *Graduate Texts in Mathematics.* Springer, 1995.

Index