Mathematical Programming Computation (2022) 14:43–84
https://doi.org/10.1007/s12532-021-00203-z

FULL LENGTH PAPER

Optimal patchings for consecutive ones matrices

Marc E. Pfetsch[1] · **Giovanni Rinaldi**[2] · **Paolo Ventura**[2]

Received: 23 November 2018 / Accepted: 12 April 2021 / Published online: 7 June 2021

Abstract
We study a variant of the weighted consecutive ones property problem. Here, a 0/1-matrix is given with a cost associated to each of its entries and one has to find a minimum cost set of zero entries to be turned to ones in order to make the matrix have the consecutive ones property for rows. We investigate polyhedral and combinatorial properties of the problem and we exploit them in a branch-and-cut algorithm. In particular, we devise preprocessing rules and investigate variants of "local cuts". We test the resulting algorithm on a number of instances, and we report on these computational experiments.

Keywords Consecutive ones property · Tucker matrices · Polyhedral combinatorics · Branch-and-cut

Mathematics Subject Classification Primary 90C10; Secondary 90C57 · 52B12

1 Introduction

A 0/1-matrix A has the *strict consecutive ones property for rows* (strict C1P) if the ones in each row appear sequentially. A matrix A has the *consecutive ones property for rows* (C1P) if the columns of A can be permuted such that the resulting matrix has the strict consecutive ones property for rows (see Fulkerson and Gross [16]). If A has the (strict) C1P, we say (somewhat abusing language) that A is (strictly) C1P.

✉ Marc E. Pfetsch
pfetsch@opt.tu-darmstadt.de

Giovanni Rinaldi
giovanni.rinaldi@iasi.cnr.it

Paolo Ventura
paolo.ventura@iasi.cnr.it

[1] Department of Mathematics, Technische Universität Darmstadt, Dolivosstraße 15, 64293 Darmstadt, Germany

[2] Istituto di Analisi dei Sistemi ed Informatica "Antonio Ruberti" – CNR, Via dei Taurini 19, 00185 Rome, Italy

Therefore, being C1P is a property of 0/1-matrices that is preserved under row and column permutations.

The problem of checking whether a given 0/1-matrix is C1P can be solved in linear time using the algorithm of Booth and Lueker [5]. The situation is quite different for optimizing over the set of all C1P matrices of a given size. Indeed, the *Weighted C1P Problem* (WC1P), i.e., finding an $m \times n$ C1P matrix of minimum cost with respect to a linear objective function, is NP-hard (Booth [4] and Papadimitriou [30]).

The weighted C1P problem was investigated by Oswald and Reinelt [27–29] who provided a polyhedral study and a branch-and-cut algorithm that produces a certified optimal solution. They also introduce applications in the fields of developmental psychology [31], computational biology [7], archaeology [22], and film-making. Their computational results show that the problem is very difficult to solve to certified optimality even for relatively small matrices. For example, in [29], Oswald and Reinelt report on computational results with randomly generated square instances. Even some instances of size 13×13 were not solvable within one hour by their implementation and hardware.

Fortunately, there are applications of the Weighted C1P Problem having a particular structure that can be exploited to make it possible to attack larger instances. This is the case, for example, when, in addition to the cost matrix C, a second matrix $M \in \{0, 1\}^{m \times n}$ is given, typically non C1P, and one is asked for the least cost *C1P positive patching* of M, i.e., a C1P matrix obtained from M by switching some of its 0 entries to 1. This *Weighted Positive C1P Patching Problem* is the subject of this paper.

The complementary weighted negative C1P patching problem, in which the least cost C1P matrix obtained from M by switching some of its 1 entries to 0 is determined, is less frequent in applications; we therefore concentrate on the positive C1P patching problem. Thus, from now on we will use the name "C1P patching" for "positive C1P patching" and we will denote this problem by *WC1PP*. It generalizes WC1P, since the two problems coincide when M is the zero matrix. Even if we restrict the objective function coefficients to be all nonnegative, the WC1PP Problem is NP-hard, since it generalizes the so-called average order spread problem, proved to be NP-hard by Fink and Voss [14]. Such a proof also works when the entries of the objective function are all 1s. In this case, we denote the WC1PP Problem as the *Minimum Length C1P Patching (MLC1PP)* Problem.

1.1 Applications

There are a number of applications of the WC1PP Problem that have been discussed in the literature. Actually, some of the examples mentioned in [27] are of this type, like the ones in archaeology and film-making. We give here a short description of some other typical applications and variants (see, e.g., [10] for additional details).

Open Stacks Problem The Open Stacks problem is to generate a set of cutting patterns that cut raw material into smaller items of required sizes and quantities in order to minimize the waste. Once the optimal patterns have been used to perform the cutting operations, another optimization problem arises in practical applications. Indeed, the items cut from the panels are stacked around the cutting machine. Such stacks remain

"open" during the complete production time of the related items and the same stack can be used only for items whose production does not overlap over time. Then one has to find a cutting pattern permutation that minimizes either the total stack opening time or the maximum number of stacks that are simultaneously open during the cutting process. In the literature, the total stack occupation time and the maximum number of simultaneously open stacks problems are known as Time of Open Stacks (TOS) Problem and Maximum number of Open Stacks (MOS) Problem, respectively (see Linhares and Yanasse [25]). In both cases, a feasible solution is a positive patching X of the binary *production matrix* M in which columns (rows) are associated with the panels (items) and $M(i, j) = 1$ if and only if at least one item of type i is produced in the panel j. The MOS Problem then seeks for such an X that minimizes the maximum number of 1s in each column, while the TOS Problem reduces to an MLC1PP Problem. The MOS problem has been investigated in the literature, see, for instance, Baptiste [2], de la Banda and Stuckey [11], and [10].

Very Large Scale Integration circuit design (VLSI) In VLSI design, the *gates* correspond to circuit nodes and different connections between them are required. Each connection involves a subset of nodes and is called *net*. Note that to connect the gates of a net, it may be necessary to cross other gates not included in the net, depending on the gate layout sequence. Also, a single *connection track* can be used to place non-overlapping net wires. The total wire length determines the connection cost, while the number of tracks determines the total circuit area, which may be limited by design constraints or efficiency issues. Both indicators give an estimate of the circuit layout efficiency and depend on how gates are sequenced. We define the Gate Matrix Connection Cost minimization Problem (GMCCP) as the problem of finding a gate permutation such that the connection cost is minimized and the number of required tracks is limited. Let M be the incidence matrix of the circuit, i.e., $M(i, j) = 1$ if net i requires the connection of gate j, and $M(i, j) = 0$, otherwise. Then, a feasible solution to the GMCCP is a positive patching X of M and the number of required tracks for each net is the number of 1s in the corresponding column of X. Therefore, it is not difficult to see that GMCCP reduces to an MLC1PP Problem with the further request to have a bounded value for the maximum number of 1s in each column, see [10].

1.2 Outline of the paper

In Sect. 2 we will introduce some notation and resume some polyhedral results related to the Weighted C1P Problem. In Sect. 3 we will consider the convex hull of the positive C1P patchings of a given matrix. In particular we will: i) describe how to extend the facet defining inequalities introduced for the C1P case to this polytope, ii) give some conditions for a 0-lifting procedure to obtain facet defining inequalities, iii) discuss some polyhedral properties of the dominant polyhedron. We will describe in Sect. 4 the cutting planes we used within the branch-and-cut procedure that we implemented to solve the WC1PP Problem. In particular, we will give special emphasis on the *oracle-based* generated cutting planes ("local cuts"). The implementation details of the branch-and-cut algorithm will be given in Sect. 5, while the computational experiments

will be described and commented in Sect. 6. Finally, we will draw up our conclusions in Sect. 7.

2 Basic definitions and results

We collect here several definitions and results that we will use in the following.

We denote by $\mathcal{P}_{\mathrm{C1}}^{m,n}$ the set of all C1P matrices with m rows and n columns and the corresponding *C1P polytope* by $P_{\mathrm{C1}}^{m,n} := \operatorname{conv}(\mathcal{P}_{\mathrm{C1}}^{m,n})$. We will always assume that n, $m \in \mathbb{N} := \{1, 2, \dots\}$.

Let M be a given $0/1$ matrix of size $m \times n$. Recall that a *positive C1P patching* of M is a C1P matrix A of size $m \times n$ such that $A \geq M$. Let $\mathcal{P}^+(M)$ be the set of all positive C1P patchings of M and $P^+(M) := \operatorname{conv}(\mathcal{P}^+(M))$ be the corresponding *positive C1P patching polytope*. The set $\mathcal{P}^+(M)$ is nonempty, since it contains the all ones matrix $\mathbb{1}^{m\times n}$ (or just $\mathbb{1}$, if the size is clear from the context). In the same way, we define $\mathbb{0}^{m\times n}$ and $\mathbb{0}$.

Moreover, we use the following notation. For $k \in \mathbb{N}$, we write $[k] := \{1, \dots, k\}$, and we denote by $\mathcal{O}(k)$ the set of all ordered subsets of $[k]$. For a matrix $A \in \mathbb{R}^{m\times n}$ we write $A(i, j)$ for the entry of A at position $(i, j) \in [m] \times [n]$. Let $N_0(A)$ denote the set of indices (i, j) such that $A(i, j) = 0$, and let $n_0(A)$ be its cardinality. We use the inner product $\langle A, B\rangle = \sum_{i=1}^{m} \sum_{j=1}^{n} A(i, j)\, B(i, j)$ for two $m \times n$ matrices A and B.

Booth and Lueker [5] gave a linear time algorithm to test whether a given matrix is C1P, which is very important in this context. It uses the so-called PQ-tree algorithm. However, if the matrix is not C1P, the algorithm does not generate a certificate. Such a certificate can be given by certain matrices that appear as minors. Indeed, Tucker [33] gave a characterization of C1P using the five types of *Tucker matrices* shown in Fig. 1: the infinite series of matrices T_k^1, T_k^2, and T_k^3 for every $k \in \mathbb{N}$ and the fixed size matrices T^4 and T^5. We need the following notation in order to state his result.

For two ordered sets $I \in \mathcal{O}(n)$ and $J \in \mathcal{O}(n)$, by A_{IJ} we denote the matrix obtained by selecting the rows and the columns of A with indices in I and in J, respectively, taken in the corresponding order. We say that A_{IJ} is a *minor* of A. Finally, Tucker's characterization can be stated as follows.

Theorem 1 (Tucker [33]) *A matrix $M \in \{0, 1\}^{m\times n}$ is C1P if and only if none of its minors is a Tucker matrix.*

Based on this characterization, Oswald and Reinelt [28,29] gave an integer programming formulation for optimizing a linear objective function over the set $\mathcal{P}_{\mathrm{C1}}^{m,n}$ described by inequalities that are all facet defining for the polytope $P_{\mathrm{C1}}^{m,n}$. Such a formulation is provided by four types of inequalities based on the matrices shown in Figs. 2 and 3. The result reads as follows.

Theorem 2 (Oswald and Reinelt [28])

(1) *The inequalities $\langle F_k^1, X_{IJ}\rangle \leq 2k+3$ for all $k \in \mathbb{N}$ and all ordered index sets I, $J \in \mathcal{O}(k+2)$ are facet-defining for $P_{\mathrm{C1}}^{m,n}$ with $m \geq k+2$, $n \geq k+2$.*
(2) *The inequalities $\langle F_k^2, X_{IJ}\rangle \leq 2k+3$ for all $k \in \mathbb{N}$ and all ordered index sets $I \in \mathcal{O}(k+2)$ and $J \in \mathcal{O}(k+3)$ are facet-defining for $P_{\mathrm{C1}}^{m,n}$ with $m \geq k+2$, $n \geq k+3$.*

$$
\begin{array}{c|cccccc}
 & 1 & 2 & 3 & \cdots & k+1 & k+2 \\
1 & 1 & 1 & 0 & \cdots & 0 & 0 \\
2 & 0 & 1 & 1 & \cdots & 0 & 0 \\
\vdots & \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\
k & 0 & 0 & \cdots & 1 & 1 & 0 \\
k+1 & 0 & 0 & \cdots & 0 & 1 & 1 \\
k+2 & 1 & 0 & \cdots & 0 & 0 & 1
\end{array}
\qquad
(T_k^1)
$$

$$
\begin{array}{c|cccccc}
 & 1 & 2 & 3 & \cdots & k+2 & k+3 \\
1 & 1 & 1 & 0 & \cdots & 0 & 0 \\
2 & 0 & 1 & 1 & \cdots & 0 & 0 \\
\vdots & \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\
k+1 & 0 & 0 & \cdots & 1 & 1 & 0 \\
k+2 & 1 & 1 & \cdots & 1 & 0 & 1 \\
k+3 & 1 & 1 & \cdots & 1 & 1 & 1
\end{array}
\qquad
(T_k^2)
$$

$$
\begin{array}{c|cccccc}
 & 1 & 2 & 3 & \cdots & k+2 & k+3 \\
1 & 1 & 1 & 0 & \cdots & 0 & 0 \\
2 & 0 & 1 & 1 & \cdots & 0 & 0 \\
\vdots & \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\
k+1 & 0 & 0 & \cdots & 1 & 1 & 0 \\
k+2 & 0 & 1 & \cdots & 1 & 0 & 1
\end{array}
\qquad
(T_k^3)
$$

$$
\begin{array}{c|cccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 \\
1 & 1 & 1 & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 1 & 1 & 0 & 0 \\
3 & 0 & 0 & 0 & 0 & 1 & 1 \\
4 & 0 & 1 & 0 & 1 & 0 & 1
\end{array}
\qquad
(T^4)
$$

$$
\begin{array}{c|ccccc}
 & 1 & 2 & 3 & 4 & 5 \\
1 & 1 & 1 & 0 & 0 & 0 \\
2 & 1 & 1 & 1 & 1 & 0 \\
3 & 0 & 0 & 1 & 1 & 0 \\
4 & 1 & 0 & 0 & 1 & 1
\end{array}
\qquad
(T^5)
$$

Fig. 1 Tucker matrices, where $k \in \mathbb{N}$

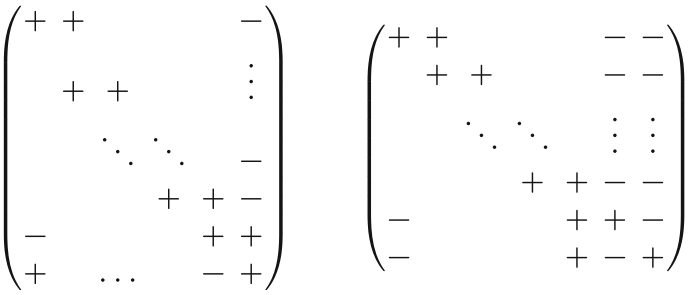

Fig. 2 Oswald-Reinelt matrices F_k^1 of size $(k+2) \times (k+2)$ and F_k^2 of size $(k+2) \times (k+3)$. Here, "+" stands for "+1", "−" for "−1", and an empty entry stands for "0". Note that F_k^1 is transposed with respect to the one defined in [28]

Fig. 3 Oswald-Reinelt matrices F^3 of size 4×6 and F^4 of size 4×5. As usual "+" stands for "+1", "−" for "−1"

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

(3) *The inequalities* $\langle F^3, X^{IJ} \rangle \leq 8$ *for all ordered index sets* $I \in \mathcal{O}(4)$ *and* $J \in \mathcal{O}(6)$ *are facet-defining for* $P_{\mathrm{C1}}^{m,n}$ *with* $m \geq 4$, $n \geq 6$.
(4) *The inequalities* $\langle F^4, X^{IJ} \rangle \leq 8$ *for all ordered index sets* $I \in \mathcal{O}(4)$ *and* $J \in \mathcal{O}(5)$ *are facet-defining for* $P_{\mathrm{C1}}^{m,n}$ *with* $m \geq 4$, $n \geq 5$.
(5) *The inequalities in Parts (1)–(4), together with the trivial inequalities* $X \geq$ *and* $X \leq \mathbb{1}$, *define an integer programming formulation of the set* $\mathcal{P}_{\mathrm{C1}}^{m,n}$.

Remark 1 In the paper of Oswald and Reinelt, there is a typo in the definition of matrix F^4. We verified the proof of the above theorem for the corrected matrix by enumerating all C1P matrices of size 4×5, checking that the inequality $\langle F^4, X^{IJ} \rangle \leq 8$ is valid, and that there are 20 affinely independent vectors satisfying the inequality with equality for each $I \in \mathcal{O}(4)$ and $J \in \mathcal{O}(5)$ (Oswald and Reinelt proved that $P_{\mathrm{C1}}^{m,n}$ is full-dimensional). The result then follows from the trivial lifting theorem of Oswald and Reinelt [28, Theorem 2].

Remark 2 The set $\mathcal{P}^+(M)$ and the polytope $P^+(M)$ are not monotone, i.e., if $X \in P^+(M)$ and $X \leq Y$ then not necessarily $Y \in P^+(M)$. Indeed, the matrix

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

is C1P, and hence $M \in \mathcal{P}^+(M)$. However,

(a) $\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} \notin \mathcal{P}^+(M)$, since it coincides with the Tucker matrix T_1^1, while
(b) $\begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} \in \mathcal{P}^+(M)$, since switching columns 2 and 3 yields a strict C1P matrix.

More recently, other inequalities, which are also facet-defining for $P_{\mathrm{C1}}^{m,n}$, have been presented by de Giovanni et al. [9].

3 Polyhedral properties of the C1P patching polytope

In this section, let $M \in \{0, 1\}^{m \times n}$ be the given matrix.

3.1 Basic results

The polytope $P^+(M)$ has the following basic properties.

Proposition 1 *Let* $M \in \{0, 1\}^{m \times n}$ *and* $X \in P^+(M)$. *Then*

$$X(i, j) = 1 \quad \text{for all } (i, j) \text{ with } M(i, j) = 1. \tag{1}$$

Moreover, $\dim(P^+(M)) = n_0(M)$.

Proof Equation (1) follow by definition for all vertices of $P^+(M)$ and by convexity for any $X \in P^+(M)$. Therefore, $\dim(P^+(M)) \leq n_0(M)$. We denote the $m \times n$ matrix with entry (i, j) equal to one and all the other entries equal to zero by E_{ij}. Then the $n_0(M) + 1$ matrices $\mathbb{1}$ and $\mathbb{1} - E_{ij}$ for all (i, j) with $M(i, j) = 0$ are contained in $\mathcal{P}^+(M)$ and are affinely independent. □

Because of Proposition 1, from now on we assume that all inequalities $\langle A, X\rangle \leq \alpha$ that are valid for $P^+(M)$ are in the standard form, i.e., $A(i, j) = 0$, for all $M(i, j) = 1$.

Proposition 2 *Let* $M \in \{0, 1\}^{m\times n}$. *The trivial inequalities* $0 \leq X(i, j) \leq 1$ *are facet defining for* $P^+(M)$ *for all* (i, j) *with* $M(i, j) = 0$.

Proof For $X(i, j) \geq 0$, consider the matrices $\mathbb{1} - E_{ij}$ and $\mathbb{1} - E_{k\ell} - E_{ij}$ for all $(k, \ell) \neq (i, j)$ with $M(k, \ell) = 0$, where E_{ij} is defined as in the proof of Proposition 1. These matrices are contained in $\mathcal{P}^+(M)$, since any matrix containing at most two 0s is C1P (the corresponding columns can be permuted to opposite ends of the matrix). These $n_0(M)$ affinely independent matrices satisfy $X(i, j) = 0$, which completes the proof.

For $X(i, j) \leq 1$, consider the matrices $\mathbb{1}$ and $\mathbb{1} - E_{k\ell}$ for all $(k, \ell) \neq (i, j)$ with $M(k, \ell) = 0$. They satisfy $X(i, j) = 1$, are contained in $\mathcal{P}^+(M)$ and are affinely independent. □

Another fundamental fact is that $P^+(M)$ is a face of $P^{m,n}_{\mathrm{C1}}$. Indeed, any entry (i, j) of a matrix $X \in P^+(M)$ for which $M(i, j) = 1$ is fixed to 1. Thus, the inequalities in Theorem 2 yield nontrivial valid inequalities for $P^+(M)$. In particular, their IP-formulation yields a valid formulation for the positive C1P patching problem as well. It turns out, however, that they do not always define facets. The following results deal with the corresponding conditions.

The *support* of a matrix A is the submatrix obtained from A by removing all its zero rows and its zero columns. Let $\overline{M} = \mathbb{1} - M \in \{0, 1\}^{m\times n}$ denote the *complement* of M. We say that $A \in \{0, 1\}^{m\times n}$ *supports* $B \in \{0, 1\}^{m\times n}$ if $A(i, j) \geq B(i, j)$, for all $i, j \in [m] \times [n]$.

Theorem 3 (1) *The inequality* $\langle F^1_k, X_{IJ}\rangle \leq 2k + 3$ *with* $k \in \mathbb{N}$ *is facet defining for* $P^+(M_{IJ})$ *if* $k \geq 2$ *and either* $M_{IJ} \in \{0, 1\}^{k+2\times k+2}$ *is supported by* T^1_k *or* $M_{IJ} = T^2_{k-1}$.

(2) *The inequality* $\langle F^2_k, X_{IJ}\rangle \leq 2k + 3$ *with* $k \in \mathbb{N}$ *is facet defining for* $P^+(M_{IJ})$ *if* $M_{IJ} \in \{0, 1\}^{k+3\times k+2}$ *is supported by a matrix obtained from* T^3_k *by removing the 1s in the last row.*

Proof The proof of Claim (1) is given by slightly modifying the arguments in [29]. To shorten notation, let $A = F^1_k$ and $\alpha = 2k + 3$. Thus, $\langle A, X_{IJ}\rangle \leq \alpha$ ((A, α), for short) is the inequality under investigation. Let $\langle B, X_{IJ}\rangle \leq \beta$ be a valid inequality that is satisfied with equality by all the feasible solutions that satisfy (A, α) with equality. In order to prove the claim, it will be enough to show that there exists $\delta > 0$ such that $B(i, j) = \delta A(i, j)$ for any $(i, j) \in N_0(M_{IJ})$; recall that all variables $X_{IJ}(i, j)$ with

Fig. 4 The matrix Z used in the proof of Theorems 3 and 5. Observe that Z is strictly C1P and a root of $\langle F_k^1, X\rangle \le 2k+3$

	1	2	3	$\cdots$	$k+1$	$k+2$
1	1	1	1	$\cdots$	1	0
2	1	1	1	$\cdots$	1	0
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\ddots$	$\vdots$	$\vdots$
k	1	1	$\cdots$	1	1	0
$k+1$	0	1	$\cdots$	1	1	1
$k+2$	1	1	$\cdots$	1	1	1

$(i, j) \in (I \times J) \setminus N_0(M_{IJ})$ are set to 1 and that $P^+(M_{IJ})$ has dimension $n_0(M_{IJ})$ (as for Claim (2) in Proposition 1). Moreover, we will show that there exists at least one positive patching of M_{IJ} that satisfies both equalities, which implies $\beta = \delta\alpha$. Consequently, the faces defined by $\langle A, X_{IJ}\rangle \le \alpha$ and $\langle B, X_{IJ}\rangle \le \beta$ coincide.

First partition $N_0(M_{IJ})$ into $\mathcal{S}^+ := \{(i, j) \in N_0(M_{IJ}) : A(i, j) = 1\}$, $\mathcal{S}^- := \{(i, j) \in N_0(M_{IJ}) : A(i, j) = -1\}$ and $\mathcal{S}^0 := \{(i, j) \in N_0(M_{IJ}) : A(i, j) = 0\}$.

Consider the matrix

$$Z := M_{IJ} + E_{k+2,k+1} + \sum_{(i,j)\in\mathcal{S}^0\cup\mathcal{S}^+} E_{ij},$$

as depicted in Fig. 4. Moreover, for each $(i, j) \in \mathcal{S}^0$, let Z^{ij} be defined as follows: $Z^{ij} := Z - E_{ij}$, if $i \le k+1$, and $Z^{ij} := Z - E_{ij} + E_{k+1,1} - E_{k+2,k+1}$, if $i = k+2$. Observe that $\langle A, Z\rangle = \langle A, Z^{ij}\rangle = \alpha$. In the following we prove that Z and Z^{ij} are positive patchings of M_{IJ}. First, notice that both Z and Z^{ij} support M_{IJ}, by construction. Then, Z is strict C1P, and it is not difficult to check that also Z^{ij} is C1P. Indeed, consider the case with $i \le k+1$. If $j = 1$, Z^{ij} is strictly C1P, and if $j > 1$ one has to exchange columns j and $k+1$ (columns j and 2, if $i = k+1$) to get a strict C1P matrix. On the other hand, if $i = k+2$, moving columns j and $k+1$ in the first two positions returns a strict C1P matrix. Therefore, Z and Z^{IJ} are roots of (A, α) (and thus also of (B, β)). Hence, by subtracting the equations $\langle B, X\rangle = \beta$ for $X = Z$ and $X = Z^{ij}$, we obtain

$$\langle B, Z^{ij}\rangle - \langle B, Z\rangle = B(i, j) = \beta - \beta = 0.$$

Hence, $B(i, j) = 0$ for all $(i, j) \in \mathcal{S}^0$.

Consider the matrix $W^{ij} = T_{k-1}^2 + E_{ij}$, for each $(i, j) \in \mathcal{S}^-$. It is not difficult to see that W^{ij} satisfies (A, α) with equality. We now show that $W^{ij} \in \mathcal{P}^+(M)$. Indeed $W^{ij} \ge M$. Observe that $\mathcal{S}^- = \{(k+2, k+1), (k+1, 1)\} \cup \{(i, k+2) : i = 1, \ldots, k\}$. If $(i, j) = (k+2, k+1)$ we have that W^{ij} is strictly C1P. If $i \le k$ and $j = k+2$, then it is enough to move column $k+2$ between columns i and $i+1$ to get a strict C1P matrix. Finally, if $(i, j) = (k+1, 1)$, we obtain a strict C1P matrix by moving column $k+2$ in first position. Since all these W^{ij} are roots of (A, α) (and therefore of (B, β)) that differ only for one element, we have that all coefficients $B(i, j)$ for $(i, j) \in \mathcal{S}^-$ have the same value, say γ.

If $M_{IJ} = T_{k-1}^2$ or $M_{IJ} = T_k^1$ then we are done, since, in this case, $\mathcal{S}^+ = \varnothing$.

Therefore assume that $M_{IJ} \leq T_k^1$ and $\mathcal{S}^+ \neq \varnothing$. Notice that $T_k^1 - E_{ij}$ is a C1P positive patching of M_{IJ} for all $(i, j) \in \mathcal{S}^+$. Moreover, every such vector is a root of (A, α) and hence of (B, β). It follows that all coefficients $B(i, j)$ for $(i, j) \in \mathcal{S}^+$ have the same value, say δ. Let $(\bar{\imath}, \bar{\jmath}) \in \mathcal{S}^+$ (in particular, $B(\bar{\imath}, \bar{\jmath}) = \delta$). Then

$$Y := T_k^1 - E_{\bar{\imath}\bar{\jmath}} \quad \text{and} \quad Y' := T_k^1 + \sum_{j\,:\,T_k^1(\bar{\imath}, j)=0} E_{\bar{\imath}j}$$

are C1P positive patchings of M_{IJ} and (B, β) is tight to both of them. If $\bar{\imath} \leq k$, it follows that

$$\langle B, Y' \rangle - \langle B, Y \rangle = B(\bar{\imath}, k+2) + B(\bar{\imath}, \bar{\jmath}) = \gamma + \delta = 0.$$

Similarly, $\gamma + \delta = 0$ holds for $\bar{\imath} = k + 1$ and $\bar{\imath} = k + 2$, too.

In conclusion, we know that $B(i, j) = \delta$, if $(i, j) \in \mathcal{S}^+$, $B(i, j) = -\delta$, if $(i, j) \in \mathcal{S}^-$, and $B(i, j) = 0$, if $(i, j) \in \mathcal{S}^0$. Thus, (B, β) is a multiple of (A, α), which concludes the proof.

Claim (2) can be proved by the same arguments used by Oswald and Reinelt [29] to prove case (3) of Theorem 2. □

Remark 3 The statement of Part (1) in Theorem 3 becomes false, if $M_{IJ} \geq T_k^1$, $M_{IJ} \neq T_k^1$ (instead of $T_k^1 \geq M_{IJ}$) and $M_{IJ} \leq T_{k-1}^2$, $M_{IJ} \neq T_{k-1}^2$ (instead of $M_{IJ} = T_{k-1}^2$): For $k = 3$, the inequality $\langle F_3^1, X \rangle \leq 9$ does not define a facet for $P^+(T_2^2 - E_{4,4})$.

Similarly, Part (2) in Theorem 3 becomes false, if $M_{IJ} = T_k^3$: For $k = 2$, the inequality $\langle F_2^2, X \rangle \leq 7$ does not define a facet for $P^+(T_2^3)$.

Following the line of Theorem 3, we also checked whether inequalities $\langle F^3, X \rangle \leq 8$ and $\langle F^4, X \rangle \leq 8$ are facet defining for $P^+(T^4)$ and $P^+(T^5)$, respectively. The following two observations address these two questions.

Remark 4 Applying $\langle F^3, X \rangle \leq 8$ to T^4 yields the inequality $\langle A, X \rangle \geq 1$ with

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

which defines a facet for $P^+(T^4)$.

Remark 5 Applying $\langle F^4, X \rangle \leq 8$ to T^5 yields the inequality $\langle A, X \rangle \geq 1$ with

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

which does not define a facet for $P^+(T^5)$.

Remarks 3, 4, and 5 can be checked by using `polymake` [17–19]. The convex hull algorithm used by `polymake` is `cdd` [15], which in turn relies on the exact arithmetic of GMP [20]. The same holds for the rank computations that we have to perform in `polymake`.

3.2 Projection and lifting

In the following we will deal with the relation between the positive patching polytope $P^+(M)$ and the polytope $P^+(M_{IJ})$ for a minor M_{IJ}.

Lemma 1 *For $M \in \{0,1\}^{m\times n}$ and $I \in \mathcal{O}(m)$, $J \in \mathcal{O}(n)$, we have:*

$$\{X_{IJ} : X \in \mathcal{P}^+(M)\} \subseteq \mathcal{P}^+(M_{IJ}).$$

Equality holds if $|J| = n$.

Proof If $X \in \mathcal{P}^+(M)$, then by definition X is C1P and so is the submatrix X_{IJ}. Furthermore, since $M \le X$, then $M_{IJ} \le X_{IJ}$. Hence, $X_{IJ} \in \mathcal{P}^+(M_{IJ})$. This shows the first claim.

If $|J| = n$, for $Y \in \mathcal{P}^+(M_{IJ})$ define the following matrix:

$$X(i,j) = \begin{cases} Y(i,j) & \text{if } i \in I \\ 1 & \text{otherwise} \end{cases} \qquad \text{for all } (i,j) \in [m] \times [n].$$

It follows from the construction that $X \in \mathcal{P}^+(M)$ and $X_{IJ} = Y$, which shows the second claim. □

As we will show in the following subparagraph, an inequality keeps the property of being facet-defining if we restrict the inequality to its support (i.e., if we remove rows and columns with all zero coefficients). In order to prove this property, we first need the following result. Here, an inequality $\langle A, X\rangle \le \alpha$, with $A, X \in \mathbb{R}^{I\times J}$, is said to be obtained by *(trivially) lifting* the inequality $\langle A', X'\rangle \le \alpha$, with $A', X' \in \mathbb{R}^{I'\times J'}$ and $I' \subseteq I$, $J' \subseteq J$, if $A(i,j) = A'(i,j)$, for all $(i,j) \in I' \times J'$, and $A(i,j) = 0$, for all $(i,j) \in (I \setminus I') \times (J \setminus J')$.

Lemma 2 *Let $A \in \mathbb{R}^{m\times n}$ and $\langle A_{IJ}, X_{IJ}\rangle \le \beta$ be a valid inequality for $P^+(M_{IJ})$, where $I \in \mathcal{O}(m)$ and $J \in \mathcal{O}(n)$. Assume that $A(i,j) = 0$ for $(i,j) \notin I \times J$. Then the trivially lifted inequality $\langle A, X\rangle \le \beta$ is valid for $P^+(M)$.*

Proof Let $X^\star \in \mathcal{P}^+(M)$. By Lemma 1, $X^\star_{IJ} \in P^+(M_{IJ})$. Because the inequality $\langle A_{IJ}, X_{IJ}\rangle \le \beta$ is valid for $P^+(M_{IJ})$, it follows that

$$\langle A, X^\star\rangle = \langle A_{IJ}, X^\star_{IJ}\rangle \le \beta,$$

i.e., the lifted inequality is valid for $P^+(M)$. □

We obtain the following.

Lemma 3 *Let $\langle A, X\rangle \le \beta$ be a facet defining inequality for $P^+(M)$ and let I and J be such that $A(i,j) = 0$ for each $(i,j) \notin I \times J$. Then $\langle A_{IJ}, X_{IJ}\rangle \le \beta$ is facet defining for $P^+(M_{IJ})$.*

Proof Assume that there exists $B \in \mathbb{R}^{m \times n}$ with $B(i, j) = 0$ for all $(i, j) \notin I \times J$, such that $\langle B_{IJ}, X_{IJ} \rangle \leq \gamma$ defines a facet for $P^+(M_{IJ})$ and

$$\{X_{IJ} \in \mathcal{P}^+(M_{IJ}) : \langle B_{IJ}, X_{IJ} \rangle = \gamma\} \supsetneq \{X_{IJ} \in \mathcal{P}^+(M_{IJ}) : \langle A_{IJ}, X_{IJ} \rangle = \beta\},$$

i.e., $\langle A_{IJ}, X_{IJ} \rangle \leq \beta$ does not define a facet. Since, both A and B are zero outside $I \times J$, by Lemma 2, $\langle B, X \rangle \leq \gamma$ is valid for $P^+(M)$ and

$$\{X \in \mathcal{P}^+(M) : \langle B, X \rangle = \gamma\} \supsetneq \{X \in \mathcal{P}^+(M) : \langle A, X \rangle = \beta\},$$

i.e., $\langle A, X \rangle \leq \beta$ does not define a facet for $P^+(M)$, a contradiction. □

The reverse question asks whether we can get valid or even facet-defining inequalities from inequalities valid for minors.

Oswald and Reinelt [28,29] proved that every trivial row and column lifting yields facets for the C1P polytope $P_{\mathrm{C1}}^{m,n}$. For positive C1P patching polytopes, this is not true.

Remark 6 Consider the matrix

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

By enumerating all positive C1P patchings with respect to M, one can verify that $\langle A, X \rangle \leq 7$ with

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

defines a facet of $P^+(M)$. However, $\langle \begin{pmatrix} A \\ 0^T \end{pmatrix}, X \rangle \leq 7$ does not define a facet for

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

Thus, trivial row lifting does not always result in a facet-defining inequality.

Remark 7 Similarly, consider the matrix

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

As above, one can verify that $\langle A, X \rangle \leq 7$ with

$$A := \begin{pmatrix} 1 & 2 & -1 & -1 \\ -1 & 1 & -1 & 1 \\ -1 & 1 & 1 & -1 \\ 1 & -2 & 1 & 1 \end{pmatrix}$$

defines a facet of $P^+(M)$.

Moreover, the trivially column lifted inequality $\langle [A,\ 0],\ X\rangle \leq 7$ does not define a facet for the polytope $P^+([M,\ \mathbb{1}^{m\times 1}])$. (Note that in this case we have $\dim P^+(M) = \dim P^+([M,\ \mathbb{1}^{m\times 1}]) = 8$).

Despite these examples, we can give some sufficient conditions for matrix M to obtain facet defining inequalities of $P^+(M)$ by trivially lifting the inequalities defined in Theorem 3. The arguments are similar to the ones given in the proof of Theorem 3. We first consider the case of a general facet defining inequality.

Theorem 4 *Consider the matrix*

$$M = \begin{pmatrix} M_{IJ} \\ \overline{M} \end{pmatrix}.$$

Assume $\langle A_{IJ}, X_{IJ}\rangle \leq \beta$ to be facet defining for $P^+(M_{IJ})$ and $\overline{M}$ strictly C1P for any permutation of its columns. Then $\langle A, X\rangle \leq \beta$, obtained as the trivial lifting of $\langle A_{IJ}, X_{IJ}\rangle \leq \beta$ to the M-space, is facet defining for $P^+(M)$.

Proof Let $M_{IJ} \in \{0,1\}^{m_1\times n}$ and $\overline{M} \in \{0,1\}^{m_2\times n}$. Moreover, let $z_1 = n_0(M_{IJ})$ and $z_2 = n_0(\overline{M})$. In the following we will construct $z_1 + z_2$ affinely independent roots of $\langle A, X\rangle \leq \beta$. Since by Proposition 1 the dimension of $P^+(M)$ is $z_1 + z_2$, this will prove the claim of the theorem.

Since $\langle A, X\rangle \leq \beta$ is facet defining for $P^+(M_{IJ})$, there exist z_1 affinely independent roots that can easily be extended to the $(m_1 + m_2) \times n$ space of M by adding the rows of matrix $\overline{M}$, so to obtain matrices $X_1, \ldots, X_{z_1}$. In particular, and w.l.o.g., let the columns of M be ordered in such a way that X_{z_1} is strictly C1P. Observe now that, in order to be strictly C1P for any permutation of the columns, each row of $\overline{M}$ must either contain at most one 1, or have all 1 entries. Now, iteratively for $p = z_1+1, \ldots, z_1+z_2$, let X_p be obtained from X_{p-1} by switching to 1 an entry $(i_p, j_p) \in N_0(X_{p-1})$ such that: *i)* $m_1+1 \leq i_p \leq m_1+m_2$ and *ii)* X_p is strictly C1P (as X_{p-1} is also strictly C1P, this can be easily done). Therefore, by construction, the matrices of $\{X_1, \ldots, X_{z_1+z_2}\}$ are affinely independent roots of $\langle A, X\rangle \leq \beta$, which proves the claim. □

In some cases, also inequalities in Theorem 2 can be trivially lifted and still induce facets:

Theorem 5 *Let*

$$M = \begin{pmatrix} M_{IJ} & M_3 \\ M_1 & M_2 \end{pmatrix},$$

assume $(M_1|M_2)$ to be strictly C1P, and $M_3 = 0$.

(1) *If $M_{IJ} \in \{0,1\}^{(k+2)\times(k+2)}$, with $k \geq 2$, is supported by T^1_k or $M_{IJ} = T^2_{k-1}$, then the trivial lifting of $\langle F^1_k, X_{IJ}\rangle \leq 2k+3$ to the M-space is facet defining for $P^+(M)$.*

(2) *If $M_{IJ} \in \{0,1\}^{(k+2)\times(k+3)}$, with $k \geq 2$, is supported by a matrix obtained from T^3_k by removing the 1s in the last row, then the trivial lifting of $\langle F^2_k, X_{IJ}\rangle \leq 2k+3$ to the M-space is facet defining for $P^+(M)$.*

Proof The proof of this theorem mimics the one given for Theorem 3.

Claim (1). Let $M_2 \in \{0,1\}^{m_1 \times n_1}$; consequently, $M_1 \in \{0,1\}^{m_1 \times (k+2)}$, $M_3 = \{0\}^{(k+2) \times n_1}$, and $M \in \{0,1\}^{(k+2+m_1) \times (k+2+n_1)}$. Moreover, let (A, α), with $\alpha = 2k+3$, be the trivial lifting of $\langle F_k^1, X_{IJ} \rangle \leq 2k+3$ to the $(k+2+m_1) \times (k+2+n_1)$ space, and let (B, β) be an inequality that is satisfied with equality by all the feasible solutions that satisfy (A, α) with equality.

By the same arguments used in the proof of Theorem 3 we can show that $B(i, j) = A(i, j)$ for all $i, j \in [k+2]$. Now, let $Z \in \{0,1\}^{(k+2) \times (k+2)}$ be defined as in the proof of Theorem 3 and illustrated in Fig. 4, and let

$$Z_0 = \begin{pmatrix} Z & 0 \\ M_1 & M_2 \end{pmatrix}.$$

Observe that Z_0 is strictly C1P and it is a root of (A, α) (and consequently of (B, β)). Let z be the number of 0s in the matrix (M_1, M_2) (i.e., $z = n_0([M_1, M_2])$) and, iteratively for each $s = 1, \ldots, z$, let Z_s be obtained from Z_{s-1} by setting an entry $(i_s, j_s) \in N_0(Z_{s-1})$, with $k+3 \leq i_s \leq k+2+m_1$ to 1, such that Z_s is still strictly C1P (always possible, since Z_{s-1} is strict C1P). Since (A, α) (and then also (B, β)) is tight for Z_s, this proves that $B(i_s, j_s) = 0$, for each $(i_s, j_s) \in N_0([M_1, M_2])$.

It remains to consider the coefficients of B corresponding to the matrix M_3. In this case, let $\overline{Z}_0$ be the matrix obtained from Z_0 by setting all the entries of $[M_1, M_2]$ to 1. Again $\overline{Z}_0$ is strictly C1P and a root of both (A, α) and (B, β). Now, iteratively for each row i from $k+2$ to 1 and from each column j from $k+3$ to $k+2+n_1$, let $\overline{Z}_s$ be obtained from $\overline{Z}_{s-1}$ by setting $\overline{Z}_s(i, j) = 1$. Again, it is not difficult to see that $\overline{Z}_s$ is C1P (if $i \leq k$, it suffices to move column $k+2$ in the last position to get a strictly C1P matrix) and that it is a root of (A, α) (and thus of (B, β)). This implies that $B(i, j) = 0$ for all the entries corresponding to the matrix M_3. Therefore, we have that (B, β) coincides with (A, α) and this concludes the proof.

Similar arguments to those used for Claim (1) also prove Claim (2). □

It is an open question whether one can obtain necessary and sufficient conditions for trivial lifting in general.

3.3 The dominant polyhedron

Of special interest is the case when the objective function of the WC1PP problem to be minimized has only nonnegative coefficients. This, for example, is the case in real world applications where turning a zero into a one is a costly, rather than a profitable, operation.

Clearly, minimizing a nonnegative linear function over $P^+(M)$ is equivalent to minimizing the same function over $D^+(M)$, the dominant of $P^+(M)$ defined by the Minkowski sum of $P^+(M)$ and $\mathbb{R}_+^{m \times n}$:

$$D^+(M) = P^+(M) + \mathbb{R}_+^{m \times n}.$$

Unfortunately, we do not have at hand an integer linear description of $D^+(M)$. Despite this fact, we can derive facet defining inequalities for $D^+(M)$ by minimiz-

ing over such polyhedron, as it will be discussed in Sect. 4. Therefore, here and in Sect. 4.3.5, we will discuss some properties of $D^+(M)$ that we will use to obtain some algorithmic advantages exploited in the solution algorithm. We start with the following result on the facet defining inequalities of $D^+(M)$. As it is usual in the literature about dominant polyhedra, here the valid inequalities will be presented in $\langle A, X\rangle \geq \alpha$ form, so to assume, w.l.o.g., $A \geq \mathbb{0}$ and $\alpha \geq 0$.

Theorem 6 *Let $M \in \{0,1\}^{m\times n}$ and let $\langle A, X\rangle \geq \alpha$ define a facet of $D^+(M)$ that is not equal to a facet defined by $X(i,j) \geq 0$ for some $i \in [m]$ and $j \in [n]$.*

(1) *If row $\bar{\imath}$ of M contains at most one 1 entry, then $A(\bar{\imath}, j) = 0$ for each $j \in [n]$.*
(2) *If column $\bar{\jmath}$ of M contains all 0 entries, then $A(i, \bar{\jmath}) = 0$ for each $i \in [m]$.*

Proof We will first prove Case (1). Since for $X \in D^+(M)$ we have that $X + Y \in D^+(M)$ for all $Y \in \mathbb{R}^{m\times n}_+$, validity of $\langle A, X\rangle \geq \alpha$ implies that $A(i,j) \geq 0$ for all $i \in [m]$ and $j \in [n]$; consequently, in order the inequality to be supporting, $\alpha \geq 0$. Let $Q := \{j : M(\bar{\imath}, j) = 0\}$. Then assume, by contradiction, that there exists $\bar{\jmath} \in Q$ such that $A(\bar{\imath}, \bar{\jmath}) > 0$.

Now observe that there exists at least one positive patching of M, say $\bar{X}$, with $\langle A, X\rangle = \alpha$ (i.e., $\bar{X}$ is a root of (A, α)) such that $\bar{X}(\bar{\imath}, \bar{\jmath}) = 1$. Indeed, if not, all roots of $\langle A, X\rangle \geq \alpha$ will also be roots of $X(\bar{\imath}, \bar{\jmath}) \geq 0$, contradicting the hypothesis that (A, α) defines a facet that does not arise from a trivial inequality of some variable.

Moreover, let $\tilde{X}$ be obtained from $\bar{X}$ by setting to 0 all the elements $(\bar{\imath}, j)$ with $j \in Q$. Then, row $\bar{\imath}$ of $\tilde{X}$ coincides with row $\bar{\imath}$ of M and thus, as it contains at most one 1 entry, cannot contribute to any Tucker minor. Consequently, since $\bar{X} \in \mathcal{P}^+(M)$, also $\tilde{X} \in \mathcal{P}^+(M)$. Therefore, $\langle A, \tilde{X}\rangle < \langle A, \bar{X}\rangle = \alpha$, contradicting the assumption that $\langle A, X\rangle \geq \alpha$ is valid for $D^+(M)$.

Similar arguments also prove Case (2) (here we use the observation that no Tucker minor contains a column with all 0 entries). □

The algorithmic consequences of this theorem will be detailed in Remark 9 below.

4 Separation

In the branch-and-cut algorithm described in Sect. 5 below we make use of three kinds of cutting planes:

- A dictionary of inequalities derived from the Tucker matrices that appear as minors of the given matrix M;
- exactly and heuristically separated inequalities as stated in Theorem 2, generated from the current fractional solution, and
- cutting planes based on an optimization oracle (local cuts).

We describe the corresponding separation algorithms in the following.

4.1 A dictionary of inequalities

In order to create a dictionary of Tucker minors of the input matrix M, we use the following three procedures:

(1) Relying on the proof of Tucker [33] for the characterization of C1P-matrices in Theorem 1, we use the method of Lekkerkerker and Boland [24] for recognizing interval graphs. Let $G = (U \cup V, E)$ be the bipartite graph associated to to M, where U and V are nodes corresponding to the rows and columns of M, respectively, and $E = \{\{u, v\} : u \in U,\ v \in V,\ M(u, v) \neq 0\}$. Each Tucker minor corresponds to an *asteroidal triple* in V, i.e., a triple $(a, b, c) \in V \times V \times V$ such that there exists a path between each of the two nodes that avoids the neighborhood of the third. Hence, to generate Tucker minors, one needs to enumerate all triples and, for each triple, all possible legal paths between each pair of its nodes. To keep the running time acceptable, we compute only one path for each pair of nodes. Then we check whether the resulting matrices form Tucker minors.
(2) Starting from M, we iteratively (temporarily) remove rows and columns from front to back while the resulting matrix does not have the C1P. The matrix T at the end of this method is a Tucker minor, by Theorem 1. Then we randomly remove from M a row or a column containing a row or column of T and we repeat the process until a C1P matrix results.
(3) For each of the Tucker minors found by applying (1) or (2), we look for other Tucker minors of the same type that can be generated by replacing one row or column in the current submatrix by equal parts of different rows or columns of M.

For each Tucker minor T provided by the procedures (1), (2), and (3), we generate the following inequalities that are stored in a pool, which is separated during the branch-and-cut procedure:

(a) inequalities from Theorem 2. In particular, if T is T^1_k or T^2_{k-1} with $k > 1$, we generate $k + 2$ symmetric copies of the corresponding inequality $\langle F^1_k, X_{IJ} \rangle \leq 2k + 3$, all violated by T. In this case, observe that, because of Theorem 3, the produced inequalities are facet defining for $P^+(T)$.
(b) if $T \in \{T^1_1, T^1_2, T^2_1, T^2_2, T^3_1, T^3_2, T^5\}$, we produce all the nontrivial facet defining inequalities of $P^+(T)$;
(c) if $T \in \{T^1_3, T^1_4, T^3_3, T^3_4, T^2_3, T^4\}$, we produce all the nontrivial facet defining inequalities of $D^+(T)$.

The facet defining inequalities produced for the cases (a) and (b) are generated off-line using the software suite `polymake` mentioned above.

4.2 Inequalities from Oswald and Reinelt

In order to separate the current fractional LP-solution $X^\star$, we generate inequalities as stated in Theorem 2. In particular, we apply two separation procedures. First, we use a rounding algorithm as described by Oswald and Reinelt [28,29]. The general idea is to first round $X^\star$ to an integer matrix $\overline{X}$ and then, if $\overline{X}$ is not C1P, apply the Steps (2) and (3) described in the previous section to $\overline{X}$. Here, for each Tucker minor T, we only generate inequalities from Theorem 2, since the facet defining inequalities from $P^+(T)$ or $D^+(T)$ could, in general, not be valid for $P^+(M)$.

Then, for the inequalities (1) and (2) of Theorem 2, we also apply the exact separation procedures described in [29]. Such algorithms reduce the corresponding

separation problem to the solution of a sequence of shortest path problems in a set of suitable graphs. Their overall complexity is rather time consuming ($O(n^3\,(n+m))$ and $O(n^4\,(n+m))$, respectively); therefore, we apply them only if no cuts are generated by the rounding procedure described above.

All the generated inequalities are stored in a pool that is used for separation at every later cutting plane phase. See [28,29] for more details on this method.

4.3 Oracle-based separation

One can generate valid (facet defining) inequalities violated by an arbitrary given point by means of an *optimization oracle*, in the case when the size of the matrix M is small. Our approach refers to so-called "local cuts", see Applegate, Bixby, Chvátal, and Cook [1] and to the so-called "target cuts", see Buchheim et al. [6].

4.3.1 The local cuts method

We first describe the general idea, which is nothing but a rephrasing of the simpler of the two directions of the polynomial-time equivalence of the separation and the optimization problem given, e.g., by Grötschel, Lovász, and Schrijver [21].

Assume that we are given a nonempty polyhedron $P \subseteq \mathbb{R}^d$, a point $x^\star \in \mathbb{R}^d$, and we want to solve the separation problem for P with respect to $x^\star$. In addition, we have an (efficient) *optimization oracle* for the following problem:

$$\min \{\langle c, x\rangle \,:\, x \in P\}, \tag{2}$$

for any $c \in \mathbb{R}^d$.

The goal is to find an inequality $\langle a, x\rangle \geq a_0$ that is valid for P and violated by $x^\star$, or show that $x^\star \in P$ (for reasons that will be clear in the forthcoming discussion, here we deal with valid inequalities in the $\geq$ form). This can be obtained by solving the following *separation problem*:

$$\begin{aligned}
(\text{LCSP}) \quad \min \quad & \langle x^\star, a\rangle - a_0 & & \\
& \langle v, a\rangle - a_0 \geq 0 & \text{for all } v \in \operatorname{conv}(V), & \quad (3)\\
& \langle r, a\rangle \geq 0 & \text{for all } r \in \operatorname{cone}(R), & \quad (4)\\
& -1 \leq a_i \leq 1 & \text{for all } i = 1, \ldots, d, & \quad (5)\\
& a,\ a_0 \text{ free}, & &
\end{aligned}$$

where V and R are the sets of vertices and of extreme rays, respectively, of P. Clearly, an optimal solution $(a^\star, a_0^\star)$ to (LCSP) defines an inequality $\langle a^\star, x\rangle \geq a_0^\star$ that is valid for P because it satisfies (3) and (4) and that is violated by $x^\star$ if the optimal value of (LCSP) is negative. The constraints (5) are used to guarantee that (LCSP) is always bounded.

Usually (LCSP) is solved with a delayed row generation, i.e., with a cutting plane procedure that iteratively constructs the sets of constraints (3)–(4). At each iteration of

the algorithm, the current solution $(\bar{a}, \bar{a}_0)$ is checked for feasibility w.r.t. P. This can be done by solving (2) with $c = \bar{a}$, by means of the optimization oracle. If the optimal value is at least $\bar{a}_0$, then $(\bar{a}, \bar{a}_0)$ is optimal for (LCSP) and we stop. Otherwise, the oracle either returns a finite optimal solution $\bar{x}$ with $\langle \bar{a}, \bar{x} \rangle < \bar{a}_0$ or a direction $\bar{r}$ with $\langle \bar{r}, \bar{a} \rangle < 0$. In this case, the inequality $\langle \bar{x}, a \rangle - a_0 \geq 0$ or the inequality $\langle \bar{r}, a \rangle \geq 0$, respectively, is added to the current constraint set (3)–(4), and the procedure iterates.

Recall that the separation problem (LCSP) discussed in this section has the purpose to provide the inequalities of a cutting plane procedure that solves an optimization problem over P. However, we just showed how to solve the separation problem (LCSP) over the same polyhedron P by solving a series of optimization problems over P, although with different objective functions. This approach may look bizarre at a first sight: why not to use the oracle upfront to optimize over P?

The key idea is to set up a procedure where Problem (2) is solved over a polyhedron $\overline{P}$ whose size is much smaller than the one of the original polyhedron P. Once a separating inequality is generated in the space of $\overline{P}$, some lifting technique is used to end up with an inequality in the original space. We sketch such a procedure for the case of *WC1PP*:

LC1) We identify a submatrix $\overline{M}$ of M, and we call $\overline{X}^\star$ the corresponding submatrix of matrix $X^\star$ and $\overline{P} = P^+(\overline{M})$ (for details, see Sect. 4.3.5);
LC2) We apply the above described "local cut" procedure to find a valid separating inequality $\langle \bar{A}, \overline{X} \rangle \geq \bar{\alpha}$, using an optimization oracle over the polytope $\overline{P}$;
LC3) we finally lift such an inequality $(\bar{A}, \bar{\alpha})$ to an inequality (A, α) that is valid for $P^+(M)$ and is violated by $X^\star$. To do so, we apply the trivial lifting procedure, whose polyhedral properties have been investigated in Sect. 3.2.

Observe that, if matrix $\overline{M}$ is chosen sufficiently small, even an optimization oracle based on total enumeration of all feasible patchings can be used in the step LC2 of the above procedure. However, even if $X^\star \notin P^+(M)$, there is no guarantee that also $\overline{X}^\star$ falls outside $P^+(\overline{M})$. Therefore, if $\overline{M}$ is too small, the odds that the procedure terminates with no separating inequality found are pretty high. Thus, a tradeoff has to be made in practice.

The separation process described so far is usually called *dual separation*. An alternative approach is the so-called *primal separation* where one seeks for a valid inequality violated by the current fractional solution that, in addition, is satisfied at equality by a given integral vertex p of the polyhedron P. The rationale behind this kind of separation is that, if p turns out to be an optimal solution, no inequalities will be generated that are not tight at the optimum and thus not necessary to prove optimality. When, as in our case, P has only 0/1 vertices, primal separation and optimization are polynomially equivalent [12].

In our context, primal separation with respect to an integer vertex p of P is simply achieved by adding the constraint

$$\langle p, a \rangle - a_0 = 0$$

to the linear program (LCSP).

4.3.2 Generating local cuts of high dimensions

The inequalities generated by the method described in Sect. 4.3.1 in general define faces of the polytope $\overline{P}$ of dimensions that are not necessarily maximal. Moreover, the lifting procedures, to generate the inequality in the original space, typically do not increase the dimension, unless significant computational efforts are spent. Therefore, it is advisable to modify the "local cut" scheme in order to produce inequalities that define high dimensional faces of $\overline{P}$.

Applegate et al. [1] (see also Chvátal et al. [8]) presented a procedure, called "tilting", that takes a separating inequality (possibly not facet defining) and terminates with a separating inequality that defines a facet of $\overline{P}$. This procedure starts with a maximal set S of affinely independent points of $\overline{P}$ which are roots of the current inequality and iteratively extends S with a new point that is found by a (possibly long) series of calls to the optimization oracle. The procedure stops when $|S| = \dim(\overline{P})$.

Here we use the following, slightly different, approach to obtain the same result. As usual, we are given a polyhedron $\overline{P}$ and a point $\bar{x}^\star \notin \overline{P}$ to be separated. Let $\langle a, \bar{x}\rangle \geq a_0$ be a valid inequality for $\overline{P}$ with $\langle a, \bar{x}^\star\rangle < a_0$. We are also given a point $\bar{x}_0 \in \overline{P}$. Possibly such a point is chosen to be in the interior of the polyhedron. Let $\bar{z}$ be the intersection of the segment $[\bar{x}^\star, \bar{x}_0]$ with a facet of $\overline{P}$. With probability 1 such a facet F is unique and $\bar{z}$ belongs to its interior. If this is not the case, let F be the intersection of all the facets of $\overline{P}$ containing $\bar{z}$. The procedure terminates with an inequality that defines F.

The algorithm iteratively generates a sequence of points in the segment $[\bar{x}^\star, \bar{z}]$, that starts with $\bar{x}^\star$ and ends with $\bar{z}$, and with a separating inequality for each of these points. At each iteration i, we have a point $\bar{x}_i$ that needs to be separated and we find a separating inequality by means of the optimization oracle. If such an inequality does not exist, $\bar{x}_i = \bar{z}$ and we are done. Otherwise, let $\langle a^i, \bar{x}\rangle \geq a_0^i$ be the inequality generated; then we set $\bar{x}_{i+1}$ to the point of the segment $[\bar{x}^\star, \bar{x}_0]$ that satisfies $\langle a^i, \bar{x}\rangle \geq a_0^i$ at equality.

4.3.3 The target cuts method

A similar method was proposed by Buchheim et al. [6] for the case when P is a polytope and a point x_0 in its interior is known, in particular, P is full-dimensional. The corresponding model is based on the solution of the following linear program

$$\begin{aligned} \text{(TP)} \quad \min \quad & \langle x^\star - x_0, a\rangle & \\ & \langle v - x_0, a\rangle \geq -1 & \text{for all } v \in \operatorname{conv}(V), \\ & a \text{ free}, & \end{aligned}$$

where, as before, V is the set of vertices of P. Let $P_0 := \{x - x_0 \in \mathbb{R}^d : x \in P\}$ be the polytope P shifted by x_0. By assumption, 0 belongs to the interior of P_0. Observe that (TP) is derived from (LCSP) by setting $P = P_0$. Therefore, (4) can be removed because P_0 is a polytope and a_0 can be set to -1 without loss of generality, since 0 is an interior point of P_0. Moreover also (5) can be removed because, by setting a_0 to -1, Problem (TP) cannot be unbounded.

An optimal solution to (TP) provides an inequality valid for P_0 that is violated by $x^\star - x_0$ if its value is strictly less than -1.

The advantage of this approach is that such an inequality $\langle a^\star, x - x_0\rangle \geq -1$ is also facet defining for P_0, if (TP) is solved by the simplex algorithm or by any other method that provides vertex solutions. This can be seen by observing that an optimal basis has n rows, corresponding to points of P_0 that are necessarily linearly independent and are roots of $(a^\star, -1)$, see [6].

Besides the fact that the knowledge of an interior point of P is mandatory, a possible drawback of this method is that the constraint matrix of (TP) is usually dense and has non-integral coefficients (due to the shifting by the vector x_0) also in the case when the vertices of P are sparse and binary.

As in the case of local cuts, Problem (TP) is solved with a delayed row generation performed by calling an optimization oracle for Problem (2).

4.3.4 Interior point

The procedures described in Sects. 4.3.2 and 4.3.3 need to be given a point x_0 in the (strict) interior of P as input. Due to the well known Carathéodory's Theorem, such a point can be obtained as the (strict positive) convex combination of $d + 1$ affinely independent points $x_1, \ldots, x_{d+1} \in P$. This task can be achieved rather easily in our case. Indeed, let $\overline{M}^{\overline{m}\times\overline{n}}$ be the submatrix of M that we identified in order to produce a violated inequality and recall that the dimension d is the number of 0 entries of $\overline{M}$. Consider the matrix $\mathbb{1}^{\overline{m}\times\overline{n}}$ and the d matrices $\mathbb{1}^{\overline{m}\times\overline{n}} - E_{ij}$, for all (i, j) such that $\overline{M}(i, j) = 0$ (again E_{ij} is the $\overline{m} \times \overline{n}$ matrix with entry (i, j) equal to one and all the other entries equal to zero). It is not difficult to see that these $d + 1$ matrices are affinely independent and that they are all C1P. If we take their convex combination with weights $\frac{1}{d+1}$ we get a point $X_0 \in [0, 1]^{\overline{m}\times\overline{n}}$ in the strict interior of $P^+(\overline{M})$. In particular, $X_0(i, j) = 1$, if $M(i, j) = 1$ and $X_0(i, j) = \frac{d}{d+1}$, otherwise.

4.3.5 Optimization oracles

As mentioned before, a key element in local or target cut generation is the optimization oracle for P. Since the optimization over $P^+(M)$ is an NP-hard problem, it seems hard to avoid some kind of pseudo-enumeration. In particular, for small sizes of the matrix M a brute-force approach is reasonably fast.

It is possible to generate all feasible solutions of $\mathcal{P}^+(M)$ by generating all permutations of the columns of M. This operation, which is obviously done in $n!$ steps, can be implemented with the Johnson-Trotter algorithm (see, e.g., [23]), which has the advantage of generating the next permutation by exchanging two consecutive columns, thus simplifying the objective function computation. For each such permutation, it is then easy to generate all positive patchings that make the permuted matrix, say $\overline{M}$, strongly C1P, and to find the patching, say $X^\star$, that minimizes the objective function $\langle C, X\rangle$. For each row i of $\overline{M}$, let i_ℓ and i_r be the column indices of its leftmost and of its rightmost 1-entry, respectively. Then necessarily, $X^\star(i, j) = 1$ for all $i_\ell \leq j \leq i_r$. Moreover, we can extend the sequence of 1's of $X^\star$ to the left of i_ℓ and to the right

of i_r if the contribution to the objective function corresponding to each of such two extensions is negative. If row i in $\overline{M}$ contains at least one 1, this operation can clearly be performed in $O(n)$ time. If all entries of row i of $\overline{M}$ are 0's, the optimal sequence of consecutive 1's can be found, still in $O(n)$, by Kadane's algorithm (see, e.g., Column 7 in [3]). In conclusion, the oracle runs in $O(n\,m\,n!)$ time.

An interesting simplification occurs when WC1PP has a nonnegative objective function, which is the case in most of the applications, in particular in all those mentioned in the Sect. 1.

When $C \geq 0$, the optimal solutions over $P^+(M)$ and over $D^+(M)$ coincide. Therefore, any optimal solution $X^\star$ of the current linear relaxation is either optimal for WC1PP or it lies outside $D^+(M)$. Thus, we can separate $X^\star$ from $D^+(M)$ instead of $P^+(M)$. Since all valid inequalities of $D^+(M)$ have nonnegative coefficients, the set R in Problem (LCSP) is the set of the d rays of $\mathbb{R}^d_+$ and we can simplify the formulation accordingly.

Moreover, in this case, as an optimization oracle we can use a straightforward adaptation of the dynamic programming algorithm that de la Banda and Stuckey [11] presented for the Open Stack problem. For completeness we describe the version of this algorithm for the Weighted Positive C1P Patching problem with the cost matrix $C \in \mathbb{R}^{m\times n}_+$.

Assume that we build up the strictly C1P optimal patching $X^\star$ column by column, such that at a given point in the algorithm we have constructed a submatrix $X^\star_{[m],S}$ whose set of columns is S, and we still have to complete it with columns from $\overline{S} := [n] \setminus S$.

Let $s \in \overline{S}$ be some column that will be placed after S and before $\overline{S} \setminus \{s\}$. Then, to make $X^\star_{[m],S\cup\{s\}}$ strongly C1P, we have to put 1's at the following rows in column s:

$$\mathcal{R}(s) := \left\{ i \;:\; \begin{array}{ll} M(i,s) = 1, & \text{or} \\ M(i,s) = 0 & \text{and there exist } \ell \in S \text{ and } j \in \overline{S} \text{ such} \\ & \text{that } M(i,\ell) = 1 \text{ and } M(i,j) = 1 \end{array} \right\}.$$

If we call $\mathrm{OP}(S)$ the optimal value of the matrix with set of columns S, we get the following dynamic programming recursion:

$$\mathrm{OP}(S) = \min_{s\in S} \big(\mathrm{OP}(S \setminus \{s\}) + \sum_{i\in\mathcal{R}(s)} C(i,s) \big).$$

When adding a new column s to S, the set $\mathcal{R}(s)$ can be updated in $O(m)$ time. Thus, the computation of $\mathrm{OP}(S)$ takes $O(m\,n)$ time, if $\mathrm{OP}(S')$ has been computed for all $S' \subset S$. Since we have to consider all subsets S, we get a total time of $O(m\,n\,2^n)$.

Remark 8 The implementation of this dynamic programming algorithm runs amazingly fast. Instances with up to 25 columns can be solved in under a minute, almost independently of the number of rows. For slightly more columns, however, the method breaks down due to memory requirements.

Observe that the dynamic programming based oracle cannot be used to generate target cuts for $D^+(M)$, since the inequalities produced may need negative coefficients in general.

Remark 9 One key decision is how the submatrices on which we generate local cuts is chosen. In our implementation, we always choose submatrices with an adjustable number of columns (in the computations we evaluate using 8, 9, 10, 11 or 12 columns). Since both optimization oracles mentioned above are relatively insensitive to the number of rows, we always include all rows of the original matrix. Moreover, when we optimize over the dominant polyhedron $D^+(M)$, we can use Theorem 6 in order to remove all rows with less than two 1s, or columns with all 0 entries, without weakening the facial properties of the generated separating inequality.

Algorithm 1: Initialization of Submatrices for Local Cuts

Input: K = target number of columns of submatrices, c = size indicator for Tucker matrices, R = number of random submatrices
$\mathcal{S} \leftarrow \varnothing$ (list of submatrices);
$k \leftarrow \min\{K, n\}$ (number of columns);
add R random $m \times k$ matrices to $\mathcal{S}$;
for $i = k - c, \ldots, k$ **do**
  for *all Tucker matrices* T *of type* T^1_{i-2} *with* i *columns* **do**
    possibly fill up the columns J of T to size k using random columns;
    add the submatrix of M with columns indexed by J to $\mathcal{S}$;

Algorithm 2: Separation of Local Cuts

Input: $\mathcal{S}$ list of candidate submatrices
foreach *submatrix* $S \in \mathcal{S}$ **do**
  separate local cuts for submatrix S;
if *separate submatrices based on LP values* **then**
  sort columns according to sum of LP-values;
  separate local cuts for submatrix formed by columns with largest values;
  sort columns according to sums of differences to best primal solution;
  separate local cuts for submatrix formed by columns with largest values;
remove submatrices without generated cut from $\mathcal{S}$;
resort $\mathcal{S}$ according to geometric mean violation of generated cuts;
possibly fill up $\mathcal{S}$ with random matrices to have size R;

We have experimented with several methods to choose the submatrices and a combination of Tucker minors and a random choice turned out to be the best one. More explicitly, we perform a filtering technique to detect important submatrices. At the beginning, we generate a list of candidate submatrices, which we initialize with random matrices and with submatrices containing a Tucker minor (possibly filling up with random columns). The details are presented in Algorithm 1; we use $R = 40$, $c = 3$, as well as by default $K = 10$ for the dominant and $K = 7$ for the polytope case.

During the algorithm, we generate cuts for each submatrix currently available and store the violation (efficacy) of the cut produced with respect to the current optimal LP-relaxation point. The matrices are then considered according to non-increasing geometric mean violation in the next round, see Algorithm 2 for details. This means that a submatrix that produced the most efficient cut the last time is used first in the next separation round. If submatrices do not produce a violated cut, they are removed from the list, possibly filling up with new random matrices. Additionally, two submatrices, which are selected using the current LP-solution, are used for separation, see Algorithm 2.

5 Algorithmic aspects

We have implemented a branch-and-cut algorithm to solve the positive patching problem. In this section, we describe the main algorithmic tools that have been used.

5.1 Preprocessing

Preprocessing steps are indispensable for solving practical problem instances of almost any optimization problem. We first consider some rules to reduce the size of the input matrix $M \in \{0, 1\}^{m \times n}$. Recall that the objective function is defined by the nonnegative $C \in \mathbb{R}_+^{m \times n}$.

We consider the following general fact.

Lemma 4 *Let $X^\star$ be a feasible solution for the positive patching problem with respect to the matrix M and let $I \in \mathcal{O}(m)$, $J \in \mathcal{O}(n)$. Then the matrix $X^\star_{IJ}$ is feasible for the positive patching problem w.r.t. the matrix M_{IJ} and $\langle C_{IJ}, X^\star_{IJ} \rangle \leq \langle C, X^\star \rangle$.*

Proof Since $X^\star$ is C1P, by Lemma 1, $X^\star_{IJ}$ is a positive C1P patching for M_{IJ}. Since $C \geq 0$, the claim about the objective function follows. □

We now have the following preprocessing steps.

Proposition 3 (1) *One can remove any row i of M that contains at most one* 1 *without changing the optimal value.*
(2) *Rows with all ones can be removed in M without changing the optimal value.*
(3) *One can remove zero columns in M without changing the optimal objective value.*
(4) *If row i dominates row k, i.e., $M(i, j) \geq M(k, j)$ for all $j \in [n]$, then there exists some optimal solution that satisfies the following inequalities:*

$$X(k, j) \leq X(i, j) \quad \text{for all } j \in [n].$$

If row i is equal to row k then there exists an optimal solution that satisfies:

$$X(k, j) = X(i, j) \quad \text{for all } j \in [n].$$

(5) *If column j and ℓ ($j \neq \ell$) of M are equal then one can remove either column and replace the cost coefficients for the other by the sum of the original coefficients of both columns without changing the optimal value.*
(6) *If the bipartite graph that has M as adjacency matrix is disconnected, one can treat the connected components separately.*

Proof (1) Consider an optimal solution $X^\star$ for the total matrix. By Lemma 4, we have $\langle C_{IJ}, X^\star_{IJ}\rangle \leq \langle C, X^\star\rangle$ for $i \in [m]$, $I \in \mathcal{O}(m)$ with $i \notin I$, and $J \in \mathcal{O}(n)$ with $|J| = n$. Conversely, we can trivially lift every solution for M_{IJ} to a solution of M, if the number of ones in row i is at most 1: because there exists no Tucker matrix with at most one 1 in a row, the lifted matrix yields a feasible solution with the same objective value.
(2) A similar argument as for Part (1) applies.
(3) Using that there exists no Tucker matrix containing a zero column, we can again use similar arguments.
(4) Assume that an optimal solution $X^\star$ satisfies $X^\star(k, j) = 1$ for some $j \in [n]$. Then either we can set $X^\star(k, j) = 0$ and still obtain an (optimal) solution or there are 1s in the original matrix M to the left and right of column j in any C1P ordering of $X^\star$. Hence, there exist distinct columns $s, t \in [n] \setminus \{j\}$, such that $M(k, s) = M(k, t) = 1$. Since row k is dominated by row i, we have $M(i, s) = M(i, t) = 1$, as well. It follows that $X^\star(i, j) = 1$ is necessary to obtain a C1P matrix.
The second statement follows by reversing the roles of k and i.
(5) We prove that there exists an optimal solution in which columns j and ℓ agree: Consider an optimal solution $X^\star$ and assume w.l.o.g. that

$$\sum_{i\in[m]} C(i, j) \leq \sum_{i\in[m]} C(i, \ell).$$

Then we obtain a feasible solution $Y^\star$ by setting

$$Y^\star(r, s) = \begin{cases} X^\star(r, s) & \text{if } s \neq \ell \\ X^\star(r, j) & \text{if } s = \ell \end{cases} \qquad \forall\, r \in [m],\ s \in [n].$$

Solution $Y^\star$ is feasible, because if $Y^\star$ contains a Tucker minor, then $X^\star$ contains a Tucker minor as well (there is no Tucker minor that contains equal columns). This shows that $\langle C, Y^\star\rangle \leq \langle C, X^\star\rangle$, i.e., if $X^\star$ is optimal, $Y^\star$ is optimal as well.
(6) If the bipartite graph is disconnected, M can be reordered into a block diagonal form:

$$\begin{pmatrix} M_1 & 0 & \dots & 0 \\ 0 & M_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & M_k \end{pmatrix},$$

where the M_i's are rectangular submatrices of M. Clearly, to turn M C1P, it suffices to make the submatrices C1P. Since $C \geq 0$, the result follows.

□

Remark 10 All of the preprocessing steps of Proposition 3 are used in our code, except for Part (6), since no disconnected graphs occurred in any of the instances. The inequalities of Part (4) are created in advance and added on demand during the branch-and-cut loop.

Remark 11 An extension of the cases discussed in Proposition 3 is not easily possible:

(1) There are Tucker matrices that contain rows with a single 0 (T_1^1 and T_k^2), columns with all ones (T_1^2 and T_1^3), columns with exactly one 1 (T^5), and columns with exactly one 0 (T_1^1, T_1^3, and T^5). Thus, one cannot (easily) preprocess these cases.
(2) If in the bipartite graph defined by M there exists an articulation node, i.e., a node such that the graph is disconnected when deleting this node, one cannot (easily) decompose the problems. This case occurs, when there is a row or column that is shared by two "blocks", see Fig. 5. A decomposition is not possible here, because using Tucker matrix T^4 yields examples that "cross the blocks", see again Fig. 5. Hence, one cannot solve the two parts independently, because in the example each block is C1P, while the total matrix is not.

5.2 Primal heuristic

To obtain good feasible solutions, we use methods of Oswald and Reinelt, see [28,29]. The idea is as follows. We generate some order of the rows of the matrix M by using the current fractional LP-solution. We then add one row after the other and test whether the resulting matrix is C1P. Once the matrix is not C1P anymore, we backtrack one step and consider all permutations that certify the C1P—these permutations can be generated from the PQ-tree. We compute the cost of generating a C1P solution by adding 1s according to this fixed permutation; this is easy: just order the columns and fill in the 1s that are needed for a strict C1P matrix. Each of the permutations yields a feasible primal solution.

This method is very successful, because it is able to test quite a number of permutations in a small amount of time.

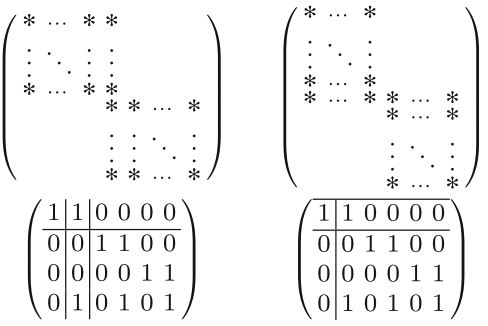

Fig. 5 Examples of articulation nodes for columns (*left*) and rows (*right*) in the bipartite graph. The bottom figure shows two examples corresponding to the cases with a column and row articulation node, respectively

6 Computational results

We implemented the discussed algorithms using a bugfix version of SCIP 4.0.1, see [26,32]. We use CPLEX 12.7.1 as the underlying LP solver. The computations were performed on a cluster with 3.5 GHz Intel Xeon E5-1620 Quad-Core CPUs, having 32 GB main memory and 10 MB cache running Linux. All computations were performed single threaded. The time limit is 1 h.

6.1 Data sets

In order to evaluate the performance of the algorithms, in particular, local cuts, we created a testset as follows. We considered the following instances for the Weighted Positive C1P Patching problem:

- 5807 instances from the Constraint Programming Modeling Challenge 2005, available at http://www.dcs.st-and.ac.uk/~ipg/challenge,
- 250 instances from Faggioli and Bentivoglio [13],
- 11 instances based on the VLSI application [10].

We first filtered out all instances that had less then 20 columns, since these can be easily handled by dynamic programming. After this filtering the number of columns for all instances ranges from 20 to 30. We then sorted out instances for which a basic version of our code (`RS-base`, see below) took longer than one hour. This leaves 197 instances in a testset, which we call `testopt`. The instances not solved by the basic version within one hour are sorted by their gap into sets with gap in (0, 10]%, (10, 20]%, and (20, 30]%; we call the testsets `testgap0-10`, `testgap10-20`, and `testgap20-30`, respectively.

Details of the following results are given in an online supplement.

6.2 Results for preprocessing

On each input, we apply the preprocessing steps described in Sect. 5.1. Table 4 in the "Appendix" shows the number of rows and columns before and after preprocessing for each instance in the `testopt` testset. The effect depends on the particular matrix. The number of removed rows varies from 0 to 20 and the number of removed columns from 0 to 6. The average number of removed rows and columns is only 0.78 and 0.83, respectively. Thus, the effect is limited on average on this testset. However, preprocessing is cheap and it can be extremely effective on some instances. For example, when applied to some real world instances from manufacturing for the open stack problem, the resulting sizes become so small that all instances can be solved within seconds; we therefore do not report these results here.

Moreover, during preprocessing we search for Tucker minors as described in Sect. 4.1. The results are again shown in Table 4 in the "Appendix". The number of found minors varies, but can be quite substantial. As described in Sect. 4.1, these Tucker minors are used to generate inequalities into a pool, which are later used in separation. Table 4 shows that the number of these cuts varies from 35 to 603 869.

6.3 Results with local cuts

We ran several variants of local cuts on the `testopt` test set. This includes local cuts (`LC`), local cuts with tilting (`LCT`), and target cuts (`TC`). For each basic variant, we consider subvariants. We vary the frequency of depths at which cuts are separated (from 0, i.e., only at the root node, to 5, i.e., every fifth depth level of the tree); this is indicated by the number attached to the basic variants, e.g., `LCT1`. Moreover, we vary the number of columns in the submatrix considered for separation (between 6 and 12), where 10 is the default for `LC` and `LCT`; this is indicated by attaching "`size`", e.g., `LCT1-size8`. Finally, we consider the variant with primal separation, e.g., `LCT1-primal`, and a variant in which we turn the reduction of the submatrix sizes off, e.g., `LCT1-nored`. In order to reduce the effects of heuristics, we initialized the runs with an optimal solution in this section.

As a base case to compare with, we use three different settings using the separation methods described in Sect. 4 that do not apply oracles:

- `RS-base` refers to the separation of dictionary inequalities (Sect. 4.1) and rounding inequalities (Sect. 4.2);
- `OR-base` refers to the techniques that were used by Oswald and Reinelt, i.e., rounding inequalities and the exact separation of the inequalities in Theorem 2, see Sect. 4.2;
- `base` refers to the separation of all three previously mentioned techniques.

The results are given in Table 1. The table shows the shifted geometric mean[1] of the number of nodes in the branch-and-bound-tree, the shifted geometric mean of the CPU

[1] The *shifted geometric mean* of values $t_1, \ldots, t_n$ with shift s is defined as

$$\left(\prod (t_i + s)\right)^{1/n} - s.$$

We use a shift $s = 10$ for time and $s = 100$ for branch-and-bound nodes in order to decrease the strong influence of very easy instances in the mean values.

Table 1 Comparison of different local cut separation variants on `testopt`

Setting	#nodes	Time	#opt	#calls	#cuts	septime	rootgap
RS-base	130,745.5	1088.88	197	308,179.7	2,667,390.8	404.41	56.47
OR-base	30,915.1	452.00	197	86,885.1	9,40,920.1	254.52	40.98
base	28,232.7	423.09	197	79,479.5	8,59,287.4	239.04	40.67
LCT0	4447.3	202.95	197	116.2	1170.5	25.26	8.60
LCT1	2436.9	176.20	197	184.9	1830.1	38.74	8.60
LCT1-nored	2431.3	181.77	197	184.4	1836.4	42.27	8.62
LCT1-primal	3016.4	184.36	197	173.5	1607.6	36.85	9.64
LCT1-size8	6882.3	193.77	197	189.6	1463.4	9.74	13.56
LCT1-size9	4107.7	173.18	197	188.8	1733.3	19.37	11.03
LCT1-size11	4107.7	173.52	197	188.8	1733.3	19.39	11.03
LCT1-size12	1163.2	310.84	197	160.0	1670.2	168.50	5.48
LCT2	2826.8	179.83	197	157.5	1637.9	32.24	8.60
LCT3	3236.0	186.48	197	145.5	1546.3	29.47	8.60
LCT4	3640.5	192.69	197	135.5	1447.3	27.64	8.60
LCT5	4033.5	197.77	197	128.8	1371.2	26.14	8.60
LC0	9240.4	346.10	197	133.8	1120.5	30.79	9.38
LC1	5871.4	305.40	197	219.5	1767.6	44.08	9.38
LC1-primal	5716.6	290.36	197	197.9	1533.5	43.18	9.96
LC1-size6	10 862.3	231.33	197	205.2	1066.7	3.11	16.20
LC1-size7	11 660.6	268.81	197	205.1	1334.0	6.18	15.21
LC1-size8	9899.4	276.13	197	208.3	1514.1	12.21	13.47
LC1-size9	8019.4	289.25	197	207.0	1674.0	22.47	11.43
LC1-size11	4610.8	344.60	197	211.9	1625.4	77.73	7.80
LC2	6820.6	312.52	197	181.0	1514.7	36.85	9.38
LC3	7440.0	321.18	197	164.1	1404.9	34.17	9.38
LC4	8056.7	328.98	196	154.7	1328.4	32.77	9.38
LC5	8639.3	338.44	196	148.2	1275.0	31.94	9.38
TC0	5630.8	1304.07	149	64.3	508.2	520.93	21.80
TC1	3536.5	1446.79	134	105.1	773.5	810.98	21.80
TC1-primal	4106.4	1494.44	136	97.3	615.8	780.95	24.95
TC1-size8	3535.4	1446.72	134	105.0	773.5	810.57	21.80
TC1-size9	3536.2	1448.37	134	105.1	773.8	811.89	21.79
TC5	5380.4	1315.75	148	71.9	587.9	553.85	21.79

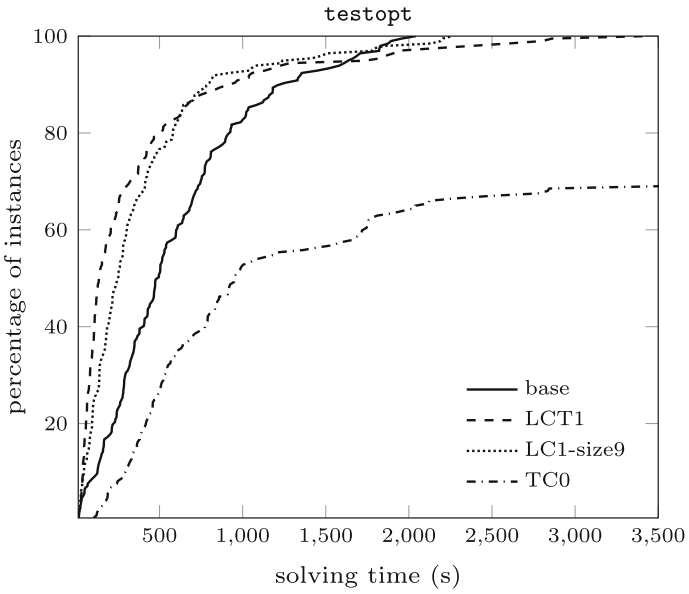

Fig. 6 Solving time diagram for different local cut variants on `testopt`: The x-axis depicts solving time, the y-axis shows the percentage of instances solved within the given time

time in seconds, the number of instances that could be solved within the time-limit (among 197), the geometric mean of the number of calls to the separation routine, the geometric mean of the number of cuts added to the LP, the shifted geometric mean of the total time used for separation, and the average gap of the root node. For the base settings, the last three numbers refer to the basic separation via Tucker minors, while for all other variants, the numbers refer to local cuts only.

We can draw the following conclusions:

- Variant `LCT1-size9` and `LCT1-size11` are the overall best variants in terms of the average solving time, very closely followed by `LCT1`, which produces less nodes in comparison to the first two.
- All variants of local cuts with tilting (`LCT...`), with the exception of variant `LCT1-size12`, are faster than the other local cut variants, see also Fig. 6.
- Comparing `LCT1` and `LC1` shows that tilting significantly improves the performance of local cuts.
- Target cuts are clearly the slowest: all target cuts are slower than the variants of any other type. This is because the separation needs too many LP-solves in order to converge to a possibly violated cut. Consequently, using target cuts only in the root node (`TC0`) is faster than the other target cut variants, because it limits this slow-down.
- When varying the separation frequency of local cuts with tilting, separation in every node (`LCT1`) is the best option in terms of average performance, but `LCT2` is closely behind and has a slightly smaller maximal solving time, see Fig. 7.
- Primal separation is not successful, but `LC1-primal` slightly improves on `LC1`.

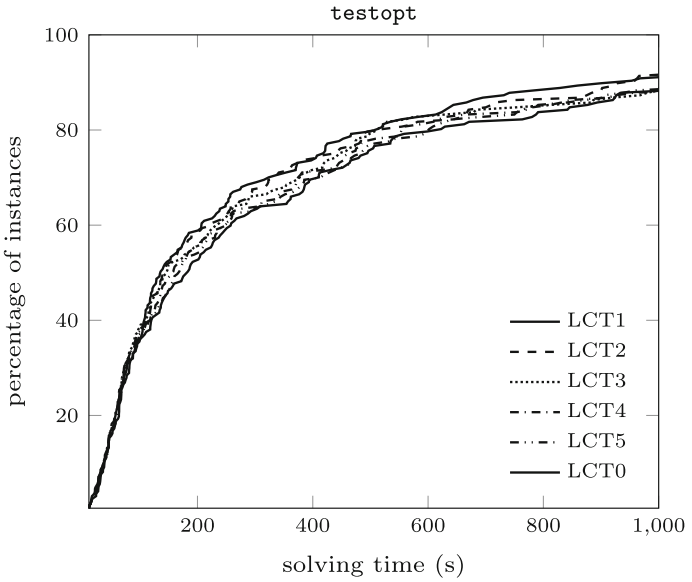

Fig. 7 Solving time diagram for local cuts with tilting with different separation frequencies on `testopt`: The x-axis depicts solving time and is truncated at 1000 seconds for better visibility, the y-axis shows the percentage of instances solved within the given time. LCT1 and LCT0 correspond to the left and right most solid line, respectively

- Turning off the submatrix reduction does not significantly increase the run times.
- Increasing or decreasing the size of the submatrix has different effects for the different variants: For `LCT1`, the right choice seems to be unclear, but 9, 10 or 11 columns produce excellent results. For `LC1`, smaller sizes seem to be better. For `TC1`, the size does not significantly change the results.

6.4 Results for unsolved instances

The previous computations show the improvement of the additional cutting planes over `RS-base`. Most variants solved all 197 instances in the `testopt` testset. We now consider the instances that could not be solved by the `RS-base` settings within one hour in order to see the additional effect of cutting planes and local cuts in particular. Moreover, we consider the influence of the heuristics. Thus, we do not initialize the optimization runs with an optimal solution. The result on the corresponding testsets `testgap0-10`, `testgap10-20`, `testgap20-30` are displayed in Table 2.

The results show that `LCT1` is able to solve a significant number of instances within the time limit that cannot be solved by `RS-base`. In fact, `LCT1` solves about 93 % of the instances in `testgap0-10` and about 59 % in `testgap10-20`. However, the solution becomes significantly more difficult for the instances in which `RS-base` had a large gap. For example, `LCT1` can only solve about 7 % of the instances in

Table 2 Comparison of local cuts with tilting on unsolved instances

Setting	#nodes	Time	#opt	#calls	#cuts	septime	rootgap
`testgap0-10` (58 instances)							
LCT1	8656.8	484.58	54	201.4	2002.3	46.15	10.03
`testgap10-20` (271 instances)							
LCT1	24 054.1	1294.97	160	233.0	2600.1	64.71	10.79
`testgap20-30` (387 instances)							
LCT1	43 295.6	3274.47	30	256.3	3038.7	84.06	16.54

Table 3 Comparison of the effect of heuristics on the `testopt` testset

Setting	#nodes	time	#opt	#calls	#found	heurtime
base	28,232.7	423.09	197	–	–	–
base-heur	28,171.1	483.99	197	983.8	2.6	61.42
LCT1	2436.9	176.20	197	–	–	–
LCT1-heur	2371.8	195.84	197	68.9	2.3	11.39

`testgap20-30`. Nevertheless, these results show the strength of the approach via local cuts with tilting.

6.5 Results of the heuristic

In this section, we investigate the effect of applying the heuristic explained in Sect. 5.2. To this end, we run `base` and `LCT1` on the `testopt` testset with and without initializing with an optimal solution. The results are shown in Table 3. The last three columns display the geometric mean of the number of calls to the heuristic, the geometric mean of the number of solutions found, and the shifted geometric mean of the time spent in the heuristic.

Not initializing with an optimal solution shows a slowdown by about 12 % for `base` and by about 10 % for for `LCT1`. Surprisingly, the number of nodes even slightly decreases for the heuristic variant in shifted geometric mean. However, this is an artifact of the mean, since the total number of nodes slightly increases. In any case, the time difference essentially comes from the time needed for running the heuristic.

7 Conclusions

We considered the weighted positive C1P patching problem, as a variant of the weighted C1P problem. The problem is NP-hard and it has several applications, specially defined on weight matrices with nonnegative entries. In the paper, we exploited the polyhedral properties of the positive patching polytope $P^+(M)$ in order to design a new branch-and-cut algorithm to solve the problem to optimality. In particular, we

first extended some facet defining inequalities to $P^+(M)$ that where known for the C1P polytope, we gave sufficient conditions for the 0-lifting procedure to produce facet defining inequalities, and we presented polyhedral properties of the dominant polyhedron of $P^+(M)$.

Then we defined separation procedures for a large set of families of valid inequalities that we used as cutting planes in our implementation of a branch-and-cut algorithm. Among these separation procedures, we in particular focused on oracle-based methods for the *on-line* generation of valid inequalities.

We finally tested the overall solution algorithm via extensive computational experiments on instances taken from the literature. The results clearly show that the oracle-based methods are very effective. This good performance also results from the right choice of parameters, e.g., frequency and submatrix size. In general, this approach seems to be well suited for optimization problems for which it is difficult to obtain a polyhedral description like the weighted positive C1P patching problem.

Acknowledgements We thank Sebastian Leipert for supplying his PQ-tree code. Moreover, we thank the three reviewers for their valuable comments and helpful suggestions.

Funding Open Access funding enabled and organized by Projekt DEAL.

Appendix

See Table 4.

Table 4 Statistics for preprocessing of `testopt`: Given are the number of rows m, columns n and the density δ (number of 1s/$(m \cdot n)$) in the original input matrix and after preprocessing. Moreover, we report the number of Tucker minors that are found: the number found as a submatrix (# sub), the number found by considering asteriodial triples (# AT), the number of copies found (# copies), and the total number (# total). Finally, we show the total number of small instance cuts generated

Instance	Original			Preproc.			Tucker minors				# cuts
	m	n	δ %	m	n	δ %	# sub	# AT	# copies	# total	
Warwick_0144	10	20	40.0	10	20	40.0	16	260	834	1110	10 311
Warwick_0150	10	20	40.0	10	19	38.0	15	195	609	819	13 128
Warwick_0243	10	20	30.0	10	20	30.0	14	171	295	480	109,794
Warwick_0248	10	20	30.0	10	18	27.0	13	122	174	309	47,707
Warwick_0250	10	20	30.0	10	19	28.5	12	115	236	363	37,266
Warwick_0257	10	20	40.0	10	20	40.0	17	296	932	1245	44,315
Warwick_0261	10	20	50.0	10	20	50.0	16	359	1536	1911	13,221
Warwick_0265	10	20	50.0	10	20	50.0	18	407	1537	1962	13,504
Warwick_0271	10	30	20.0	10	24	16.0	15	307	51	373	65,957
Warwick_0283	10	30	30.0	10	26	26.0	21	461	1006	1488	334,644
Warwick_0285	10	30	30.0	10	26	26.0	22	386	893	1301	101,011
Warwick_0288	10	30	30.0	10	26	26.0	23	394	961	1378	214,199
Warwick_0292	10	30	40.0	10	28	37.3	25	693	2631	3349	105,534
Warwick_0294	10	30	40.0	10	28	37.3	24	660	2499	3183	105,457
Warwick_0296	10	30	40.0	10	26	34.7	23	612	2191	2826	74,113
Warwick_0298	10	30	40.0	10	28	37.3	24	697	2674	3395	256,831
Warwick_0301	10	30	50.0	10	29	48.3	25	1184	5521	6730	53,994
Warwick_0302	10	30	50.0	10	28	46.7	25	1349	5437	6811	42,292
Warwick_0305	10	30	50.0	10	28	46.7	23	1174	4971	6168	42,272
Warwick_0308	10	30	50.0	10	28	46.7	24	1203	4636	5863	32,353
Warwick_0312	10	30	20.0	10	27	19.0	13	118	176	307	55,075

Table 4 continued

Instance	Original			Preproc.			Tucker minors				# cuts
	m	*n*	δ %	*m*	*n*	δ %	# sub	# AT	# copies	# total	
Warwick_0313	10	30	20.0	10	27	19.0	16	82	194	292	39,132
Warwick_0315	10	30	20.0	10	24	17.0	12	59	106	177	34,737
Warwick_0317	10	30	20.0	10	28	19.0	15	116	221	352	48,796
Warwick_0323	10	30	23.3	10	26	21.0	16	146	298	460	12,367
Warwick_0325	10	30	23.3	10	26	22.0	16	151	367	534	77,149
Warwick_0326	10	30	23.3	10	26	22.0	21	148	431	600	14,695
Warwick_0331	10	30	26.7	10	29	26.0	23	254	676	953	42,862
Warwick_0332	10	30	26.7	10	30	26.7	25	293	827	1145	59,290
Warwick_0339	10	30	26.7	10	29	26.0	22	353	790	1165	228,958
Warwick_0341	10	30	30.0	10	29	29.7	21	365	1068	1454	45,630
Warwick_0342	10	30	30.0	10	28	29.0	22	442	1054	1518	74,053
Warwick_0344	10	30	30.0	10	29	29.7	24	404	1145	1573	199,725
Warwick_0347	10	30	30.0	10	29	29.0	21	331	998	1350	24,180
Warwick_0351	10	30	33.3	10	29	32.3	22	471	1341	1834	173,215
Warwick_0353	10	30	33.3	10	30	33.3	24	531	1740	2295	71,233
Warwick_0355	10	30	33.3	10	30	33.3	24	568	1787	2379	59,148
Warwick_0356	10	30	33.3	10	30	33.3	24	504	1646	2174	159,551
Warwick_0357	10	30	33.3	10	29	32.7	22	513	1640	2175	31,746
Warwick_0358	10	30	33.3	10	29	32.7	24	525	1666	2215	23,551
Warwick_0360	10	30	33.3	10	28	32.0	20	484	1204	1708	126,883
Warwick_0363	10	30	36.7	10	28	35.0	23	542	1891	2456	25,859
Warwick_0367	10	30	36.7	10	28	33.7	23	744	2416	3183	29,733
Warwick_0369	10	30	36.7	10	30	36.7	24	561	2091	2676	38,301

Table 4 continued

Instance	Original			Preproc.			Tucker minors				# cuts
	m	n	δ %	m	n	δ %	# sub	# AT	# copies	# total	
Warwick_0375	10	30	40.0	10	29	38.3	22	736	2527	3285	62,184
Warwick_0376	10	30	40.0	10	30	40.0	24	800	2875	3699	34,677
Warwick_0377	10	30	40.0	10	30	40.0	24	657	2057	2738	62,489
Warwick_0378	10	30	40.0	10	29	38.3	24	823	2567	3414	52,825
Warwick_0380	10	30	40.0	10	30	40.0	25	803	3023	3851	45,290
Warwick_0382	10	30	43.3	10	30	43.3	25	987	3968	4980	48,066
Warwick_0389	10	30	43.3	10	30	43.3	25	807	3027	3859	55,825
Warwick_0390	10	30	43.3	10	30	43.3	24	844	3419	4287	27,473
Warwick_0391	10	30	46.7	10	30	46.7	25	1078	4597	5700	28,566
Warwick_0394	10	30	46.7	10	30	46.7	25	1266	5159	6450	91,616
Warwick_0398	10	30	46.7	10	30	46.7	27	956	3643	4626	30,724
Warwick_0401	10	30	50.0	10	30	50.0	25	1826	7300	9151	43,714
Warwick_0403	10	30	50.0	10	30	50.0	25	1231	4723	5979	35,220
Warwick_0405	10	30	50.0	10	29	48.7	25	1174	5241	6440	51,971
Warwick_0407	10	30	50.0	10	30	50.0	25	1241	5494	6760	36,075
Warwick_0409	10	30	50.0	10	30	50.0	25	1481	5852	7358	45,591
Warwick_0411	10	30	20.0	10	22	14.7	13	236	42	291	44,529
Warwick_0415	10	30	20.0	10	25	16.7	16	378	34	428	68,867
Warwick_0419	10	30	20.0	10	23	15.3	14	263	35	312	47,564
Warwick_0420	10	30	20.0	10	24	16.0	16	317	61	394	740,09
Warwick_0421	10	30	30.0	10	27	27.0	21	402	1113	1536	384,962
Warwick_0422	10	30	30.0	10	26	26.0	21	404	1294	1719	540,215
Warwick_0426	10	30	30.0	10	29	29.0	23	566	1555	2144	603,869

Table 4 continued

Instance	Original			Preproc.			Tucker minors				# cuts
	m	n	δ %	m	n	δ %	# sub	# AT	# copies	# total	
Warwick_0429	10	30	30.0	10	25	25.0	20	360	944	1324	372,070
Warwick_0440	10	30	40.0	10	27	36.0	23	693	2596	3312	135,404
Warwick_0635	15	30	13.3	15	27	12.0	13	193	55	261	28,021
Warwick_0640	15	30	13.3	13	28	12.0	15	279	67	361	43,948
Warwick_0691	15	30	13.3	15	28	12.9	13	72	94	179	13,828
Warwick_0694	15	30	13.3	15	26	12.4	13	56	93	162	13,317
Warwick_0695	15	30	13.3	15	24	11.8	12	26	46	84	12,388
Warwick_0700	15	30	13.3	15	27	12.7	10	43	103	156	9020
Warwick_1031	20	20	15.0	17	20	14.2	12	63	45	120	17,740
Warwick_1034	20	20	15.0	16	20	14.0	12	59	87	158	24,927
Warwick_1047	20	20	20.0	17	20	19.2	16	64	186	266	25,987
Warwick_1048	20	20	20.0	19	20	19.8	14	47	142	203	20,542
Warwick_1049	20	20	20.0	16	20	19.0	15	100	175	290	49,725
Warwick_1056	20	20	25.0	18	20	24.5	16	125	395	536	125,714
Warwick_1057	20	20	25.0	20	20	25.0	16	79	304	399	93,977
Warwick_1059	20	20	25.0	20	20	25.0	16	106	354	476	38,913
Warwick_1121	20	20	15.0	20	20	15.0	8	30	29	67	6293
Warwick_1122	20	20	15.0	20	20	15.0	12	20	54	86	18,877
Warwick_1123	20	20	15.0	20	20	15.0	12	28	52	92	10,907
Warwick_1124	20	20	15.0	20	19	14.8	11	21	33	65	6038
Warwick_1125	20	20	15.0	20	20	15.0	10	24	39	73	3744
Warwick_1127	20	20	15.0	20	20	15.0	12	26	46	84	8644
Warwick_1130	20	20	15.0	20	20	15.0	10	44	56	110	17,366

Table 4 continued

Instance	Original			Preproc.			Tucker minors				# cuts
	m	n	δ %	m	n	δ %	# sub	# AT	# copies	# total	
Warwick_1150	20	20	25.0	20	20	25.0	15	100	358	473	97,613
Warwick_1155	20	20	30.0	20	20	30.0	16	129	688	833	28,544
Warwick_1217	20	20	15.0	20	20	15.0	12	31	64	107	4828
Warwick_1719	30	30	6.7	19	26	5.1	3	7	6	16	132
Warwick_1853	30	30	6.7	29	28	6.4	6	5	0	11	35
Shaw_11	20	20	30.0	20	20	30.0	14	109	685	808	236,766
Shaw_12	20	20	30.0	20	20	30.0	18	115	582	715	44,055
Shaw_15	20	20	30.0	19	20	29.8	16	157	765	938	36,562
Shaw_16	20	20	30.0	19	20	29.8	15	134	605	754	24,214
Shaw_18	20	20	30.0	18	20	29.5	17	176	940	1133	52,981
Shaw_1	20	20	30.0	19	20	29.8	16	111	562	689	95,086
Shaw_20	20	20	30.0	20	20	30.0	16	109	525	650	85,954
Shaw_22	20	20	30.0	20	20	30.0	15	131	622	768	55,104
Shaw_24	20	20	30.0	19	20	29.8	15	125	527	667	108,777
Shaw_3	20	20	30.0	20	20	30.0	17	127	585	729	113,551
Shaw_4	20	20	30.0	20	20	30.0	14	121	554	689	186,518
Shaw_6	20	20	30.0	20	20	30.0	17	125	589	731	23,581
Shaw_8	20	20	30.0	20	20	30.0	16	128	656	800	65,876
HS_problem_2204	20	20	22.2	20	20	22.2	14	72	198	284	25,280
HS_problem_2209	20	20	19.2	18	20	18.8	13	75	182	270	74,316
HS_problem_2211	20	20	20.5	19	20	20.2	12	58	180	250	25,304
HS_problem_2212	20	20	18.8	19	20	18.5	12	56	153	221	42,949
HS_problem_2214	20	20	20.8	19	20	20.5	16	42	142	200	23,361

Table 4 continued

Instance	Original			Preproc.			Tucker minors				# cuts
	m	n	δ %	m	n	δ %	# sub	# AT	# copies	# total	
HS_problem_2219	20	20	21.8	20	20	21.8	12	47	214	273	24,892
HS_problem_2224	20	20	27.5	20	20	27.5	14	94	426	534	48,110
HS_problem_2225	20	20	24.0	20	20	24.0	12	67	256	335	54,845
HS_problem_2228	20	20	23.0	18	20	22.5	13	82	288	383	21,592
HS_problem_2234	20	20	28.0	19	20	27.8	17	107	475	599	22,749
HS_problem_2235	20	20	25.2	19	19	24.2	12	144	459	615	75,335
HS_problem_2236	20	20	25.8	20	20	25.8	16	78	344	438	34,939
HS_problem_2248	20	20	30.8	20	20	30.8	17	111	580	708	21,093
HS_problem_2251	20	20	28.0	20	20	28.0	17	128	464	609	12,309
HS_problem_2304	20	20	45.0	20	20	45.0	18	327	2108	2453	14,824
HS_problem_2307	20	20	45.2	20	20	45.2	17	285	1941	2243	19,864
HS_problem_2317	20	20	46.5	20	20	46.5	17	250	1653	1920	13,091
HS_problem_2318	20	20	46.2	20	20	46.2	17	306	2325	2648	22,944
HS_problem_2341	20	20	56.2	20	20	56.2	18	410	3327	3755	21,665
HS_problem_2343	20	20	55.2	20	20	55.2	18	406	3158	3582	18,968
HS_problem_2344	20	20	57.5	20	20	57.5	18	661	5148	5827	24,467
HS_problem_2346	20	20	56.8	20	20	56.8	18	406	3089	3513	16,428
HS_problem_2352	20	20	60.5	20	20	60.5	18	632	5131	5781	22,204
HS_problem_2356	20	20	60.5	20	20	60.5	18	448	3587	4053	19,994
HS_problem_2361	20	20	60.2	20	20	60.2	17	464	3685	4166	16,936
HS_problem_2362	20	20	58.2	20	20	58.2	16	468	3759	4243	14,633
HS_problem_2363	20	20	59.0	20	20	59.0	18	475	3664	4157	19,910
HS_problem_2365	20	20	56.2	20	20	56.2	18	529	4518	5065	17,218

Table 4 continued

Instance	Original			Preproc.			Tucker minors				# cuts
	m	n	δ %	m	n	δ %	# sub	# AT	# copies	# total	
HS_problem_2366	20	20	61.8	20	20	61.8	18	536	4020	4574	23,970
HS_problem_2368	20	20	58.2	20	20	58.2	18	639	5344	6001	15,461
HS_problem_2370	20	20	57.5	20	20	57.5	18	446	3787	4251	20,268
HS_problem_2372	20	20	56.2	20	20	56.2	18	520	4060	4598	20,112
HS_problem_2379	20	20	63.0	20	20	63.0	18	822	6642	7482	33,776
HS_problem_2381	20	20	65.2	20	20	65.2	18	593	4866	5477	14,729
HS_problem_2384	20	20	64.2	20	20	64.2	18	600	4932	5550	14,309
HS_problem_2385	20	20	66.2	20	20	66.2	18	648	5269	5935	24,781
HS_problem_2386	20	20	65.2	20	20	65.2	18	607	5063	5688	13,564
HS_problem_2387	20	20	68.5	20	20	68.5	18	747	5972	6737	20,736
HS_problem_2389	20	20	66.2	20	20	66.2	17	844	7027	7888	25,728
HS_problem_2393	20	20	66.0	20	20	66.0	18	752	6010	6780	30,345
HS_problem_2394	20	20	68.5	20	20	68.5	18	736	6040	6794	20,824
HS_problem_2397	20	20	61.2	20	20	61.2	18	564	4434	5016	19,477
HS_problem_2398	20	20	65.8	20	20	65.8	18	607	5119	5744	14,852
HS_problem_2400	20	20	65.5	20	20	65.5	18	710	5412	6140	17,577
HS_problem_2402	20	20	71.2	20	20	71.2	18	804	6267	7089	12,555
HS_problem_2404	20	20	69.2	20	20	69.2	17	772	6586	7375	20,714
HS_problem_2408	20	20	69.2	20	20	69.2	17	781	6373	7171	16,641
HS_problem_2409	20	20	67.8	20	20	67.8	18	759	6197	6974	19,238
HS_problem_2410	20	20	69.5	20	20	69.5	18	801	6457	7276	17,549
HS_problem_2412	20	20	69.5	20	20	69.5	18	785	6175	6978	21,208
HS_problem_2415	20	20	72.0	20	20	72.0	18	807	6224	7049	14,874

Table 4 continued

Instance	Original			Preproc.			Tucker minors				# cuts
	m	n	δ %	m	n	δ %	# sub	# AT	# copies	# total	
HS_problem_3191	15	30	22.0	15	30	22.0	21	271	857	1149	155,151
HS_problem_3195	15	30	19.8	14	30	19.6	22	187	596	805	101,935
HS_problem_3354	15	30	62.2	15	30	62.2	27	2453	16,147	18,627	63,347
HS_problem_3371	15	30	64.9	15	30	64.9	28	2230	14,014	16,272	49,366
HS_problem_3373	15	30	64.0	15	30	64.0	27	2297	15,604	17,928	43,592
HS_problem_3374	15	30	64.4	15	30	64.4	28	2376	16,319	18,723	47,759
HS_problem_3376	15	30	69.8	15	30	69.8	28	2371	15,939	18,338	30,382
HS_problem_3379	15	30	63.6	15	30	63.6	27	2330	14,916	17,273	57,897
HS_problem_3383	15	30	64.0	15	30	64.0	28	2278	16,352	18,658	39,859
HS_problem_3386	15	30	68.4	15	30	68.4	27	2870	19,949	22,846	55,877
HS_problem_3387	15	30	67.6	15	30	67.6	28	2276	15,482	17,786	37,765
HS_problem_3391	15	30	70.4	15	30	70.4	27	2747	18,811	21,585	36,171
HS_problem_3395	15	30	70.7	15	30	70.7	27	2568	16,577	19,172	38,641
HS_problem_3396	15	30	68.9	15	30	68.9	27	2429	14,817	17,273	40,014
HS_problem_3397	15	30	68.4	15	30	68.4	28	2371	15,484	17,883	47,683
HS_problem_3398	15	30	63.8	15	30	63.8	27	2302	15,896	18,225	58,132
HS_problem_3399	15	30	69.6	15	30	69.6	27	2983	21,791	24,801	46,926
HS_problem_3400	15	30	69.8	15	30	69.8	27	2530	17,916	20,473	44,614
HS_problem_3401	15	30	71.1	15	30	71.1	26	2687	17,320	20,033	37,090
HS_problem_3402	15	30	68.2	15	30	68.2	27	2799	18,916	21,742	53,270
HS_problem_3404	15	30	71.3	15	30	71.3	27	2562	16,478	19,067	37,743
HS_problem_3406	15	30	70.9	15	29	67.8	25	2363	16,322	18,710	31,911
HS_problem_3407	15	30	67.1	15	30	67.1	27	2371	15,538	17,936	42,502

Table 4 continued

Instance	Original			Preproc.			Tucker minors				# cuts
	m	n	δ %	m	n	δ %	# sub	# AT	# copies	# total	
HS_problem_3410	15	30	71.6	15	30	71.6	28	2592	16,341	18,961	39,601
p2020n0	18	20	18.1	15	19	16.1	13	65	92	170	45,469
p2020n3	19	20	17.6	17	20	17.1	14	65	183	262	64,629
p2020n6	18	20	18.3	15	20	17.5	12	71	140	223	43,098
p2030n1	30	20	12.3	21	20	10.8	13	35	93	141	24,047
p2030n7	30	20	12.2	20	20	10.5	12	38	77	127	25,752
p2040n3	40	20	10.0	23	20	7.9	10	26	33	69	5104
p2040n4	40	20	9.8	21	20	7.4	11	43	54	108	21,747
p2040n8	40	20	9.5	22	20	7.2	9	34	33	76	10,085
p2040n9	40	20	9.8	20	20	7.2	9	28	37	74	16,821
p2520n6	20	25	16.4	18	25	16.0	16	131	285	432	94,445
p3010n2	10	30	31.0	9	29	29.7	24	509	1446	1979	99,068
p3010n4	10	30	30.7	9	28	28.3	21	550	1096	1667	159,883
p3010n5	10	30	31.3	9	29	30.0	22	608	1578	2208	197,298
p3010n6	10	30	29.3	9	27	26.7	21	433	1241	1695	294,235

References

1. Applegate, D., Bixby, R., Chvátal, V., Cook, W.: The Traveling Salesman Problem: A Computational Study. Princeton University Press, Princeton (2006)
2. Baptiste, P.: Simple MIP formulations to minimize the maximum number of open stacks. In: Smith, B.M., Gent, I.P. (eds.) Proceedings of IJCAI'05—Constraint Modelling Challenge 2005, pp. 9–13. Edimburgh (2005)
3. Bentley, J.: Programming Pearls. Addison-Wesley, Reading (1986)
4. Booth, K.S.: PQ-tree algorithms. Ph.D. thesis, University of California Berkeley (1975)
5. Booth, K.S., Lueker, G.S.: Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. J. Comput. Syst. Sci. **13**, 335–379 (1976)
6. Buchheim, C., Liers, F., Oswald, M.: Local cuts revisited. Oper. Res. Lett. **36**(4), 430–433 (2008)
7. Christof, T., Jüunger, M., Kececioglu, J., Mutzel, P., Reinelt, G.: A branch-and-cut approach to physical mapping of chromosomes by unique end-probes. J. Comput. Biol. **4**, 433–447 (1997)
8. Chvátal, V., Cook, W., Espinoza, D.: Local cuts for mixed-integer programming. Math. Program. Comput. **5**, 171–200 (2013)
9. De Giovanni, L., Brentegani, L., Festa, M.: New facets for the consecutive ones polytope. Tech. rep., Optimization Online (2018). http://www.optimization-online.org/DB_HTML/2018/06/6674.html
10. De Giovanni, L., Massi, G., Pezzella, F., Pfetsch, M., Rinaldi, G., Ventura, P.: A heuristic and an exact method for the gate matrix connection cost minimization problem. Int. Trans. Oper. Res. **20**(5), 627–643 (2013). https://doi.org/10.1111/itor.12025
11. de la Banda, M.G., Stuckey, P.J.: Dynamic programming to minimize the maximum number of open stacks. INFORMS J. Comput. **19**(4), 607–617 (2007)
12. Eisenbrand, F., Rinaldi, G., Ventura, P.: Primal separation for 0/1 polytopes. Math. Program. **95**, 475–491 (2003)
13. Faggioli, E., Bentivoglio, C.A.: Heuristic and exact methods for the cutting sequencing problem. Eur. J. Oper. Res. **110**, 564–575 (1998)
14. Fink, A., Voß, S.: Applications of modern heuristic search methods to pattern sequencing problems. Comput. Oper. Res. **26**, 17–34 (1999)
15. Fukuda, K.: `cdd` home page. http://www.cs.mcgill.ca/~fukuda/soft/cdd_home/cdd.html
16. Fulkerson, D., Gross, O.: Incidence matrices and interval graphs. Pac. J. Math. **15**(3), 835–855 (1965)
17. Gawrilow, E., Joswig, M.: `polymake`: a framework for analyzing convex polytopes. In: Kalai, G., Ziegler, G.M. (eds.) Polytopes—Combinatorics and Computation, DMV Seminar, vol. 29, pp. 43–74. Birkhäuser, Basel (2000)
18. Gawrilow, E., Joswig, M.: `polymake`: an approach to modular software design in computational geometry. In: Proceedings of the 17th Annual Symposium on Computational Geometry, pp. 222–231. ACM (2001)
19. Gawrilow, E., Joswig, M.: `polymake`: Version 2.1.0. http://www.math.tu-berlin.de/polymake (2007)
20. GNU Multiple Precision Arithmetic Library. http://gmplib.org
21. Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization. Algorithms and Combinatorics, vol. 2, 2nd edn. Springer, Heidelberg (1993)
22. Kendall, D.: Incidence matrices, interval graphs and seriation in archaeology. Pac. J. Math. **28**, 565–570 (1969)
23. Kreher, D., Stinson, D.: Combinatorial Algorithms: Generation, Enumeration, and Search, vol. 7. CRC Press, Cambridge (1998)
24. Lekkerkerker, C., Boland, J.: Representation of a finite graph by a set of intervals on the real line. Fund. Math. **51**, 45–64 (1962)
25. Linhares, A., Yanasse, H.H.: Connections between cutting-pattern sequencing, VLSI design, and flexible machines. Comput. Oper. Res. **29**, 1759–1772 (2002)
26. Maher, S.J., Fischer, T., Gally, T., Gamrath, G., Gleixner, A., Gottwald, R.L., Hendel, G., Koch, T., Lübbecke, M.E., Miltenberger, M., Müller, B., Pfetsch, M.E., Puchert, C., Rehfeldt, D., Schenker, S., Schwarz, R., Serrano, F., Shinano, Y., Weninger, D., Witt, J.T., Witzig, J.: The SCIP optimization suite 4.0. Tech. rep., Optimization Online (2017). http://www.optimization-online.org/DB_HTML/2017/03/5895.html
27. Oswald, M.: Weighted consecutive ones problems. Ph.D. thesis, University of Heidelberg (2003)
28. Oswald, M., Reinelt, G.: Constructing new facets of the consecutive ones polytope. In: Jünger, M., Reinelt, G., Rinaldi, G. (eds.) Combinatorial Optimization—Eureka, You Shrink! Papers Dedicated to

Jack Edmonds, 5th International Workshop, Aussois, 2001, *LNCS*, vol. 2570, pp. 147–157. Springer, Berlin (2003)
29. Oswald, M., Reinelt, G.: Computing optimal consecutive ones matrices. In: Grötschel, M. (ed.) The Sharpest Cut, The Impact of Manfred Padberg and His Work, Optimization, pp. 173–184. MPS/SIAM (2004)
30. Papadimitriou, C.H.: The NP-completeness of the bandwidth minimization problem. Computing **16**(3), 263–270 (1976)
31. Roberts, F.: Discrete Mahematical Models with Applications to Social, Biological, and Environmental Problems. Prenctice-Hall, Englewood Cliff (1976)
32. SCIP: Solving Constraint Integer Programs. http://scip.zib.de/
33. Tucker, A.: A structure theorem for the consecutive 1's property. J. Comb. Theory Ser. B **12**, 153–162 (1972)