HAL open science

A faster algorithm for finding minimum Tucker submatrices

Guillaume Blin, Romeo Rizzi, Stéphane Vialette

To cite this version:

Guillaume Blin, Romeo Rizzi, Stéphane Vialette. A faster algorithm for finding minimum Tucker submatrices. Theory of Computing Systems, 2012, 51 (3), pp.270-281. 10.1007/s00224-012-9388-1 . hal-00657340

HAL Id: hal-00657340

https://hal.science/hal-00657340

Submitted on 16 Mar 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A faster algorithm for finding minimum Tucker submatrices

Guillaume Blin[1], Romeo Rizzi[2], and Stéphane Vialette[1]

[1] Université Paris-Est, LIGM - UMR CNRS 8049, France.
{gblin,vialette}@univ-mlv.fr
[2] Dipartimento di Matematica ed Informatica (DIMI)
Universit degli Studi di Udine, Italy. rrizzi@dimi.uniud.it

Abstract. A binary matrix has the *Consecutive Ones Property* (C1P) if its columns can be ordered in such a way that all 1s on each row are consecutive. Algorithmic issues of the C1P are central in computational molecular biology, in particular for physical mapping and ancestral genome reconstruction. In 1972, Tucker gave a characterization of matrices that have the C1P by a set of forbidden submatrices, and a substantial amount of research has been devoted to the problem of efficiently finding such a minimum size forbidden submatrix. This paper presents a new $O(\Delta^3 m^2(m\Delta + n^3))$ time algorithm for this particular task for a $m \times n$ binary matrix with at most Δ 1-entries per row, thereby improving the $O(\Delta^3 m^2(mn + n^3))$ time algorithm of [*M. Dom, J. Guo and R. Niedermeier, Approximation and fixed-parameter algorithms for consecutive ones submatrix problems, Journal of Computer and System Sciences, 76(3-4): 204-221, 2010*]. Moreover, this approach can be used – with a much heavier machinery – to address harder problems related to Minimal Conflicting Set [*G. Blin, R. Rizzi, and S. Vialette. A Polynomial-Time Algorithm for Finding Minimal Conflicting Sets, Proc. 6th International Computer Science Symposium in Russia (CSR), 2011*].

1 Introduction

A binary matrix has the *Consecutive Ones Property* (C1P) if its columns can be ordered in such a way that all 1s on each row are consecutive. Both deciding if a given binary matrix has the C1P and finding the corresponding columns permutation can be done in linear time [9, 17, 18, 22–24, 27, 30]. The C1P of matrices has a long history and it plays an important role in combinatorial optimization, including application fields such as scheduling [6, 20, 21, 35], information retrieval [25], and railway optimization [28, 29, 32] (see [15] for a recent survey). Furthermore, algorithmic aspects of the C1P turn out to be of particular importance for physical mapping [2, 12, 26] and ancestral genome reconstruction [1, 11]. (see also [10, 3–5, 13, 31] for other applications in computational molecular biology). Actually, our main motivation for studying algorithmic aspects of the C1P comes from *minimal conflicting sets* in binary matrices in the context of ancestral genome reconstruction [**?**]. A minimal conflicting set of rows in a binary matrix is a set of rows R that does not have the C1P but such that any proper subset of R has the C1P (a similar definition applies for columns). Tucker [34] has characterized the binary matrices that have the C1P by a set of forbidden submatrices, and the aim of this paper is to lay the foundations for efficiently computing minimal conflicting sets by presenting a new efficient algorithm for finding such minimum size forbidden Tucker submatrices [8].

Recently, Dom *et al.* [16] investigated some natural problems arising when a matrix M does not have the C1P property (the C1P is indeed a desirable property than often leads to efficient algorithms): find a minimum-cardinality set of columns to delete such that the resulting matrix has the C1P, find a minimum-cardinality set of rows to delete such that the resulting matrix has the C1P, and find a minimum-cardinality set of 1-entries in the matrix that shall be flipped (that is, replaced by 0-entries) such that the resulting matrix has the C1P. All these problems are **NP**-hard even for simple instances [19, 33], and hence Dom *et al.* have focused on approximation and parameterized complexity issues. To this end, they have provided a technical solution based on efficiently detecting forbidden Tucker submatrices [34].

In this paper, we presents a new $O(\Delta^3 m^2(m\Delta + n^3))$ time algorithm for finding a a minimum size forbidden Tucker submatrix in $m \times n$ binary matrices with at most Δ 1-entries per row, thereby improving the $O(\Delta^3 m^2(mn + n^3))$ time algorithm of Dom *et al.* [16, 15].

This paper is organized as follows: we first recall basic definitions in Section 2, and we then formally introduce the considered problem. In Section 3, we briefly recall the algorithm of Dom *et a.* Section 4 is devoted to presenting out technical improvement and we consider in Section 5 matrices with unbounded Δ.

2 Preliminaries

All graphs are considered as undirected. Given a graph $G = (V, E)$, let $N(v) = \{u|(u, v) \in E\}$ denote the neighborhood of vertex v in G. The neighborhood described above does not include v itself, and is more specifically the open neighborhood of v; it is also possible to define a neighborhood in which v itself is included, called the closed neighborhood. For any subset $V' \subseteq V$ of vertices, let $G[V']$ denote the subgraph of G induced by the vertices of V' (*i.e.*, $G' = (V', \{(u,v) \in E | u, v \in V'\})$). An induced cycle is a cycle that is an induced subgraph of G; induced cycles are also called chordless cycles. An asteroidal triple is an independent set of three vertices such that each pair is joined by a path that avoids the neighborhood of the third.

A binary matrix has the Consecutive Ones Property (C1P) if its columns can be ordered in such a way that all 1s on each rows are consecutive. For a matrix M, we let r_i and c_j stand for the i^{th} row and the j^{th} column of M, respectively. Let M be a $m \times n$ binary matrix. Its corresponding bipartite graph $G(M) = (V_M, E_M)$ is defined as follows: $V_M = R \cup C$, where $R = \{r_i : 1 \leq i \leq m\}$ is the set of rows of M and $C = \{c_i : 1 \leq i \leq n\}$ is the set of columns of M, and two vertices $r_i \in R$ and $c_j \in C$ are connected by an edge if and only if $M[i,j] = 1$. Equivalently, M is the reduced adjacency matrix of $G(M)$ (*i.e.* the non-redundant portion of the full adjacency matrix for the bipartite graph). Actually, it will be convenient to define $G(M)$ as a vertex-colored bipartite graph: each $r_i \in R$ (row vertex) is colored black and each $c_j \in c$ (column vertex) is colored white. See Figure 1 for an illustration (we use capital letters for black vertices and uncapitalized letters for white vertices).

$$M = \begin{array}{c} \\ r_1 \\ r_2 \\ r_3 \\ r_4 \end{array} \begin{array}{c} \begin{array}{cccccc} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} \end{array}$$

G_M

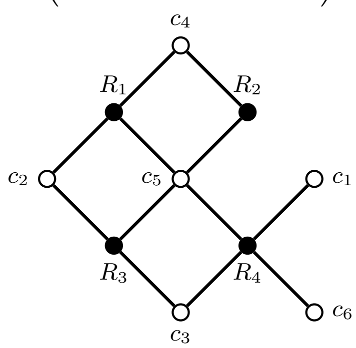

Fig. 1. A binary matrix and its corresponding vertex-colored bipartite graph.

The following result of Tucker links the C1P for binary matrices to asteroidal triples.

Theorem 1 ([34], Theorem 6). *A binary matrix has the C1P if and only if its corresponding vertex-colored bipartite graph does not contain a white asteroidal triple,* i.e. *an asteroidal triple on column vertices.*

Moreover, Tucker has characterized the binary matrices that have the C1P by a set of *forbidden submatrices* (the number of vertices in the forbidden graph defines the size of the Tucker configuration).

Theorem 2 ([34], Theorem 9). *A binary matrix has the C1P if and only if it contains none of the matrices M_{I_k}, M_{II_k}, M_{III_k} $(k \geq 1)$, M_{IV} and M_V depicted in Figure 2.*

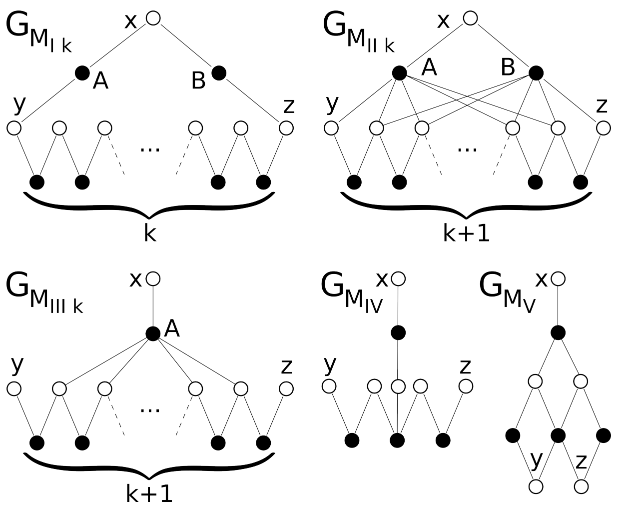

Fig. 2. Forbidden Tucker submatrices represented as vertex-colored bipartite graphs [34]. Black and white vertices correspond to rows and columns, respectively.

3 The algorithm of Dom *et al.*

In [16], Dom *et al.* provided an algorithm for finding a forbidden Tucker submatrix (*i.e.*, one of $T = \{M_{I_k}, M_{II_k}, M_{III_k}, M_{IV}, M_V\}$) in a given binary matrix. The general algorithm is as follows. For each white asteroidal triple u, v, w of $G(M)$, compute the sum of the lengths of three shortest paths connecting two by two u, v and w (each path has to avoid the closed neighborhood of the third vertex). Select an asteroidal triple u, v, w of $G(M)$ with minimum total length of the paths connecting each

	Time complexity	
Tucker configuration	Dom *et al.*	Our contribution
M_{I_k} and M_{II_k}	$O(\Delta mn^2 + n^3)$	$O(\Delta^3 m^2(n + \Delta m))$
M_{III_k}	$O(\Delta^3 m^3 n + \Delta^2 m^2 n^2)$	$O(\Delta mn^2(n + \Delta m))$
M_{IV}	$O(\Delta^3 m^2 n^3)$	Algorithm of Dom *et al.* used verbatim
M_V	$O(\Delta^4 m^2 n)$	Algorithm of Dom *et al.* used verbatim
Overall	$O(\Delta^3 m^2(mn + n^3))$	$O(\Delta^3 m^2(\Delta m + n^3))$

Table 1. Comparing our results with Dom *et al.* [16, 15].

pair of vertices and return the rows and columns of M that correspond to the vertices that occur along the three shortest paths. The authors proved that the returned submatrix does contain a forbidden Tucker submatrix of T but which is not necessarily of minimum size (for M_{III_k}, M_{IV} and M_V). Indeed, since the three shortest paths may share some vertices, the sum of the lengths of the three paths is not necessarily the number of vertices in the union of the three paths. However, Dom *et al.* showed that the returned submatrix contains at most three extra columns (resp. five extra rows) compared with a forbidden Tucker submatrix with minimum number of columns (resp. rows). To overcome this problem, they provided another algorithm devoted to M_{III_k}, M_{IV} and M_V submatrices. More precisely, they used the similarity between M_{III_k} and M_{I_k} to reduce the problem to a minimum-size chordless cycle search. For M_{IV} and M_V, they provided an exhaustive search. On the whole, Dom *et al.* provided an algorithm for finding a forbidden Tucker submatrix in a given matrix M (assuming M does not have the C1P) in $O(\Delta^3 m^2 n(m + n^2))$ time, where m is the number of rows of M, n is the number of columns of n, and Δ is the maximum number of 1-entries in a row. More precisely, the authors provided a $O(\Delta mn^2 + n^3)$ time algorithm for finding a M_{I_k} or M_{II_k} submatrix, a $O(\Delta^3 m^3 n + \Delta^2 m^2 n^2)$ time algorithm for finding a M_{III_k} submatrix, a $O(\Delta^3 m^2 n^3)$ time algorithm for finding a M_{IV} submatrix, and a $O(\Delta^4 m^2 n)$ time algorithm for finding a M_V submatrix. See Table 1.

The main contribution of this paper is a simple $O(\Delta^3 m^2\ (m\Delta + n^3))$ time algorithm for finding a minimum size forbidden Tucker submatrix. Our technical improvement on Dom *et al.* [16, 15] is based on shortest paths and two graph pruning techniques: *clean* and *complement clean* (to be defined in the next section). Graph pruning techniques were introduced by Conforti and Rao [14]. One has to note that graph pruning technique does not always succeed in the detection of induced configurations. Indeed, in [7] Bienstock gave negative results among which one can find an **NP**-completeness proof for the problem of deciding whether a graph contains an odd chordless cycle containing a given vertex. This negative result, which in attacking the perfect graph conjecture was useful in posing limits in what could have been a reasonable approach, also demonstrates that not everything can be done with the detection of induced configurations.

4 Fast detection of minimum size forbidden Tucker submatrices

In this section, we design a general algorithm that reports in polynomial-time the smallest Tucker configuration of a given matrix M if it exists. To do so, we seek for any subgraph of $G(M)$ that corresponds to a submatrix of T.

Let us introduce the `clean` and `cpl_clean` cleaning operations. Let M be a binary matrix and $G(M) = (V_M, E_M)$ be the corresponding vertex-colored bipartite graph. Let v be a vertex of $G(M)$, then $\texttt{clean}_{G(M)}(v)$ is the graph obtained by removing from $G(M)$ all neighbors of v, *i.e.*, $G(M)[V_M \setminus N(v)]$. Let v be a vertex of $G(M)$, then $\texttt{cpl\_clean}_{G(M)}(v)$ (read complement clean)) is the graph obtained by removing from $G(M)$ all vertices that do not belong to the same partition nor the neighborhood of v, *i.e.*, $G(M)[N(v) \cup \{u : \texttt{color}(u) = \texttt{color}(v)\}]$. For the sake of simplicity, we shall write $\texttt{clean}_{G(M)}(u_1, u_2, \ldots, u_k)$ for the sequence

$$\begin{aligned}
&G_1 \leftarrow \texttt{clean}_{G(M)}(u_1),\\
&G_2 \leftarrow \texttt{clean}_{G_1}(u_2),\\
&\qquad\vdots\\
&G_k \leftarrow \texttt{clean}_{G_{k-1}}(u_k),\\
&\text{Return } G_k,
\end{aligned}$$

and $\texttt{cpl\_clean}_{G(M)}(u_1, u_2, \ldots, u_k)$ for the sequence

$$\begin{aligned}
&G_1 \leftarrow \texttt{cpl_clean}_{G(M)}(u_1),\\
&G_2 \leftarrow \texttt{cpl_clean}_{G_1}(u_2),\\
&\qquad\ldots,\\
&G_k \leftarrow \texttt{cpl_clean}_{G_{k-1}}(u_k),\\
&\text{Return } G_k.
\end{aligned}$$

We now focus on the bipartite graphs that represent Tucker configurations (see Figure 2) and define our guessing functions. Define the function $\texttt{guess}_I(G(M))$ (reps. $\texttt{guess}_{II}(G(M))$) that returns all possible sets of vertices $\{x, y, z, A, B\} \subseteq V_M$ as defined for the Tucker configuration M_{I_k} (resp. M_{II_k}) in Figure 2. Furthermore, Define the function $\texttt{guess}_{III}(G(M))$ that returns all possible sets of vertices $\{x, y, z, A\} \subseteq V_M$ as defined for the Tucker configuration M_{III_k} in Figure 2. Of particular importance, in the presented algorithms, guessed vertices will never be affected (*i.e.*, deleted) by the `clean` and `cpl_clean` functions.

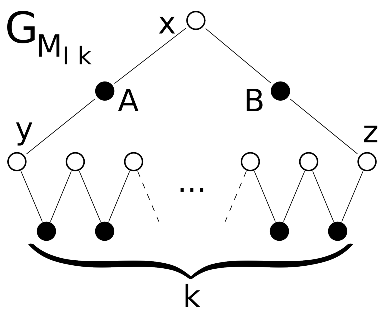

Fig. 3. M_{I_k} Tucker configuration.

Lemma 1. *Let M be $m \times n$ binary matrix with at most Δ 1-entries per row. Algorithm 1 computes the smallest submatrix $G(M_{I_k})$ in $G(M)$ in $O(m^2\Delta^3(n + \Delta m))$ time (if such a submatrix exists).*

Proof. We apply Algorithm 1 to $G(M)$. Let us first prove that if $G(M_{I_k})$ occurs in $G(M)$, then Algorithm 1 finds it. Suppose $G' = G(M_{I_k})$ occurs in $G(M)$. Then among all the guessed 5-plets x, y, z, A, B (Line 1), there should be at least one guess such that x, y, z, A, B are part of the vertices of G'. By definition, G' is a chordless cycle. Therefore, $\texttt{clean}(x, A, B)$ preserves G' since in G', (1) x is only connected to vertices A and B, (2) A is only connected to vertices x and y, and (3) B is only connected to x and z. Therefore, looking for a shortest path p in the pruned graph between y and z after having removed A and B ensures the minimality of the returned solution.

The guessing can be done in $O(m^2\Delta^3)$ time. Indeed, once A has been identified, one can (i) select x and y among the at most Δ neighbors of A, and (ii) identify B and one of its at most Δ neighbors as z such that $x \in N(B)$ and $z \notin \{x, y\}$. For each such guessing, the cleaning of x, A, B

Algorithm 1 Finding $G(M_{I_k})$ in $G(M)$

1: **for all** $\{x, y, z, A, B\} \leftarrow \texttt{guess}_I(G(M))$ **do**
2: $\quad G_1 \leftarrow \texttt{clean}_{G(M)}$(x,A,B)
3: $\quad$Remove the vertices A and B from G_1. In the resulting graph, find a shortest path p from y to z
4: $\quad$**if** p exists **then**
5: $\quad\quad S \leftarrow \{x, y, z, A, B\} \cup \{u : u \in p\}$
6: $\quad\quad$**return** the induced subgraph $G(M)[S]$
7: $\quad$**end if**
8: **end for**
9: **return** `None`

can be done in $O(\Delta + m)$ time. Finally, a shortest path between y and z can be computed in $O(n+\Delta m)$ time (the pruned graph has at most $m+n$ vertices and Δm edges). On the whole, Algorithm 1 is $O(m^2\Delta^3(n+\Delta m))$ time. $\square$

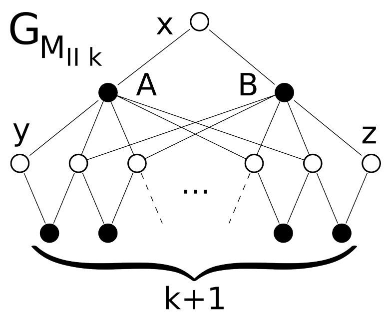

Fig. 4. M_{II_k} Tucker configuration.

Lemma 2. *Let M be a $m \times n$ binary matrix with at most Δ 1-entries per row. Algorithm 2 computes the smallest submatrix $G(M_{II_k})$ in $G(M)$ in $O(m^2\Delta^3(n + \Delta m))$ time (if such a submatrix exists).*

Proof. We apply Algorithm 2 to $G(M)$. Let us first prove that if $G(M_{II_k})$ occurs in $G(M)$, then Algorithm 2 finds it. Suppose $G' = G(M_{II_k})$ occurs in $G(M)$. Then among all the guessed 5-plets x, y, z, A, B in Line 1, there must be at least one guess such that x, y, z, A, B are part of the vertices of G'. By definition, in G' any unguessed white node is in the neighborhood of both A and B. Thus, $\texttt{cpl\_clean}(A, B)$ preserves G' since (1) y (the only white node not in the neighborhood of B) has been guessed, and (2)

z (the only white node not in the neighborhood of A) has been guessed. Moreover, in G' x should be only connected to A and B. Thus, `clean`(x) preserves G'. Finally, looking for a shortest path p in the pruned graph between y and z after having removed A and B ensures the minimality of the returned solution.

Algorithm 2 Finding $G(M_{II_k})$ in $G(M)$

for all $\{x, y, z, A, B\} \leftarrow \texttt{guess}_{II}(G(M))$ **do**
  $G_1 \leftarrow \texttt{cpl\_clean}_{G(M)}(A, B)$
  $G_2 \leftarrow \texttt{clean}_{G_1}(x)$
  Remove the vertices A and B from G_2. In the resulting graph, find a shortest path p from y to z
  if p exists **then**
    $S \leftarrow \{x, y, z, A, B\} \cup \{u : u \in p\}$
    return the induced subgraph $G(M)[S]$
  end if
end for
return `None`

The guessing can be done in $O(m^2\Delta^3)$ time. For each guessing, the cleaning/complement-cleaning of x, A, and B can be done in $O(n + m)$ time. Finally, a shortest path between y and z can be computed in $O(n+\Delta m)$ time (the pruned graph has at most $\Delta + n$ vertices and Δm edges). On the whole, Algorithm 2 is $O(m^2\Delta^3(n + \Delta m))$ time. □

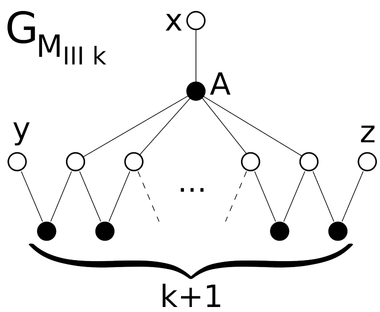

Fig. 5. M_{III_k} Tucker configuration.

Lemma 3. *Let M be a $m \times n$ binary matrix with at most Δ 1-entries in each row. Algorithm 3 computes the smallest $G(M_{III_k})$ in $G(M)$ in $O(m\Delta n^2(n + \Delta m))$ time (if such a submatrix exists).*

Algorithm 3 Finding $G(M_{III_k})$ in $G(M)$

Proof. 1: **for all** $\{x, y, z, A\} \leftarrow \texttt{guess}_{III}(G(M))$ **do**

2: $G_1 \leftarrow \texttt{cpl\_clean}_{G(M)}(A)$

3: $G_2 \leftarrow \texttt{clean}_{G_1}(x)$

4: Remove the vertex A from G_2. In the resulting graph, find a shortest path p from y to z.

5: **if** p exists **then**

6: $S \leftarrow \{x, y, z, A\} \cup \{u : u \in p\}$

7: **return** the induced subgraph $G(M)[S]$

8: **end if**

9: **end for**

10: **return** `None`

We apply Algorithm 3 to $G(M)$. Let us first prove that if $G(M_{III_k})$ occurs in $G(M)$, then Algorithm 3 finds it. Suppose $G' = G(M_{III_k})$ occurs in $G(M)$. Then among all the guessed 4-plets x, y, z, A in Line 1, there must be at least one guess such that x, y, z, A are part of the vertices of G'. By definition, in G' any unguessed white node is in the neighborhood of A. Thus, $\texttt{cpl\_clean}(A)$ preserves G' since in G' y and z (the only white nodes not in the neighborhood of A) have been guessed. Moreover, in G' x is only connected to A. Therefore, $\texttt{clean}(x)$ preserves G'. Finally, looking for a shortest path p in the pruned graph between y and z after having removed A ensures the minimality of the returned solution.

The guessing can be done in $O(m\Delta n^2)$ time. Indeed, once A has been identified, one can (i) select x among the at most Δ neighbors of A and (ii) then identify y and z. For each such guessing, the cleaning/complement-cleaning of x and A can be done in $O(n+m)$ time. Finally, a shortest path between y and z can be computed in $O(n + \Delta m)$ time (the pruned graph has at most $\Delta + n$ vertices and Δm edges). On the whole, Algorithm 3 is $O(m\Delta n^2(n + \Delta m))$ time. □

As for $G(M_{IV})$ and $G(M_V)$, a simple brute-force search yields the time complexity detailed in Lemma 4.

Lemma 4 ([16], Proposition 5.3). *Let M be a $m \times n$ binary matrix with at most Δ 1-entries per row. The smallest $G(M_{IV})$ (resp. $G(M_V)$) in $G(M)$ can be computed in $O(\Delta^3 m^2 n^3)$ (resp. $O(\Delta^4 m^2 n)$ time) if it exists.*

We are now ready to state the main result of this paper (Table 1 compares our results with Dom *et al.* [16].).

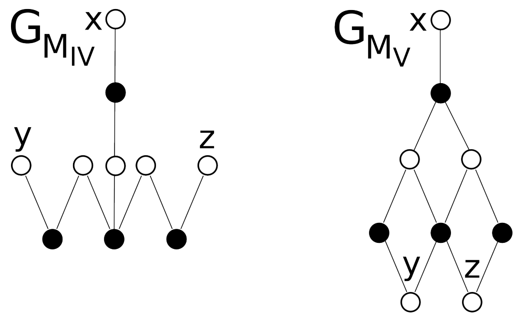

Fig. 6. M_{IV} and M_V Tucker configurations.

Theorem 3. *Let M be a $m \times n$ binary matrix that does not have the C1P with at most Δ 1-entries per row. A minimum size forbidden Tucker submatrix that occurs in M can be found in $O(\Delta^3 m^2(\Delta m + n^3))$ time.*

Notice that our results do not improve on the ones of Dom *et al.* [16] for large Δ. Furthermore, our results also do not improve on the ones of Dom *et al.* [16] for each Tucker configuration. For example, if $m = n$, our algorithm has time complexity $O(\Delta^4\, m^3)$ for M_{I_k} and M_{II_k} Tucker configurations, whereas Dom *et al.*'s algorithm has time complexity $O(\Delta\, m^3)$.

5 Matrices with unbounded Δ

As mentioned in [16], a natural question is concerned with the complexity of the problem when the number of 1s per row is unbounded. One can distinguish two subcases: the maximum number of 1s per column is bounded (say by C) or not. In the following, we prove that using a similar approach to the one used in the preceding section, one can achieve an $O(C^2n^3(m+C^2n))$ (resp. $O(n^4m^4)$) time complexity for the bounded (resp. unbounded) case.

Theorem 4. *Let M be a $m \times n$ binary matrix with at most C 1-entries per column. A minimum size forbidden Tucker submatrix that occurs in M can be found in $O(C^2n^3(m + C^2n))$ time.*

Proof. We apply Algorithms 1, Algorithms 2, and Algorithms 3 to find any forbidden submatrix of type M_{I_k}, M_{II_k} or M_{III_k}. Let us now analyze the time complexity.

For Algorithm 1, the guessing can be done in $O(n^3C^2)$ time. Indeed, once x has been identified, one can select A and B among the at most C

neighbors of x and then identify y and z such that $y \in N(A)$, $y \notin N(B)$, $z \in N(B)$, $z \notin N(A)$ and $x \neq y \neq z$. For each such guessing, the cleaning of x, A, and B can be done in $O(C+n)$ time. Finally, a shortest path between y and z can be computed in $O(m+Cn)$ time (the pruned graph has at most $m+n$ vertices and Cn edges). We conclude that Algorithm 1 is $O(C^2n^3(m+Cn))$ time.

For Algorithm 2, the guessing can also be done in $O(n^3C^2)$. For each such guessing, the cleaning/complement-cleaning of x, A and B can be done in $O(C+n)$ time. Finally, a shortest path between y and z can be computed in $O(m+Cn)$ time (the pruned graphs has at most $m+n$ vertices and Cn edges). We conclude that Algorithm 2 is $O(C^2n^3(m+Cn))$ time.

As for Algorithm 3, the guessing can be done in $O(n^3C)$ time. Indeed, once x has been identified, one can select A among the at most C neighbors of x and then identify y and z such that $y \notin N(A)$, $z \notin N(A)$ and $x \neq y \neq z$. For each such guessing, the cleaning/complement-cleaning of x and A can be done in $O(C+n)$ time. Finally, a shortest path between y and z can be computed in $O(m+Cn)$ time (the pruned graph has at most $m+n$ vertices and Cn edges). We conclude that Algorithm 3 is $O(Cn^3(m+Cn))$ time.

M_{IV} and M_V forbidden submatrices deserve careful consideration. Indeed, a direct application of Proposition 5.3 [16] for finding M_{IV} and M_V forbidden submatrices results in a $O(n^6)$ time algorithm. We use the following strategy: (i) Select the central three white nodes u, v, and w (distinct from x, y and z) among the $O(n^3)$ such triples, (ii) select four black nodes that are neighbors of at least one of u, v, or w among the $O(C^4)$ such 4-plets, and (iii) for each such combination check in $O(n)$ time whether every columns of the matrix M_{IV} appears at least once in the submatrix induced by the selected rows.

A submatrix of the type M_V can be found analogously: (i) select the two central white nodes (distinct from x, y, and z) among the $O(n^2)$ such triples, (ii) select four black nodes that are neighbors of at least one of u and v among the $O(C^4)$ such 4-plets, and (iii) for each such combination check in $O(n)$ time whether every columns of the matrix M_V appears at least once in the submatrix induced by the selected rows. ☐

Replacing C by m in the above yields the following result.

Theorem 5. *Let M be $m \times n$ binary matrix. A minimum size forbidden Tucker submatrix that occurs in M can be found in $O(n^4m^4)$ time.*

References

1. Z. Adam, M. Turmel, C. Lemieux, and D. Sankoff. Common intervals and symmetric difference in a model-free phylogenomics, with an application to streptophyte evolution. *J. Comput. Biol.*, 14:436–445, 2007.
2. F. Alizadeh, R. Karp, D. Weisser, and G. Zweig. Physical mapping of chromosomes using unique probes. *J. Comput. Biol.*, 2:159–184, 1995.
3. E. Althaus, S. Canzar, M.R. Emmett, A. Karrenbauer, A.G. Marshall, A. Meyer-Baese, and H. Zhang. Computing h/d-exchange speeds of single residues from data of peptic fragments. In ACM Press, editor, *23rd ACM Symposium on Applied Computing SAC '08*, page 12731277, 2008.
4. J.E. Atkins, E.G. Boman, and B. Hendrickson. A spectral algorithm for seriation and the consecutive ones problem. *SIAM J. Comput.*, 28(1):297310, 1998.
5. J.E. Atkins and M. Middendorf. On physical mapping and the consecutive ones property for sparse matrices. *Discrete Appl. Math.*, 71(13):2340, 1996.
6. J. J. Bartholdi, J. B. Orlin, and H. D. Ratliff. Cyclic scheduling via integer programs with circular ones. *Oper Res*, 28(5):1074–1085, 1980.
7. Dan Bienstock. On the complexity of testing for odd holes and induced odd paths. *Discrete Math.*, 90(1):85–92, 1991.
8. G. Blin, R. Rizzi, and S. Vialette. General framework for minimal conflicting set. Technical report, Université Paris Est, I.G.M., jan 2010.
9. K.S. Booth and G.S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using pq-tree algorithms. *J. Comput. System Sci.*, 13:335379, 1976.
10. C. Chauve, J. Manŭch, and M. Patterson. On the gapped consecutive ones property. In *Proc. 5th European conference on Combinatorics, Graph Theory and Applications (EuroComb), Bordeaux, France*, volume 34 of *Electronic Notes on Discrete Mathematics*, pages 121–125, 2009.
11. C. Chauve and É. Tannier. A methodological framework for the reconstruction of contiguous regions of ancestral genomes and its application to mammalian genome. *PLoS Comput. Biol.*, 4:paper e1000234, 2008.
12. T. Christof, M. Jünger, J. Kececioglu, P. Mutzel, and G. Reinelt. A branch-and-cut approach to physical mapping of chromosome by unique end-probes. *J. Comput. Biol.*, 4:433–447, 1997.
13. T. Christof, M. Oswald, and G. Reinelt. Consecutive ones and a betweenness problem in computational biology. In Springer, editor, *6th International Conference on Integer Programming and Combinatorial Optimization IPCO '98*, volume 1412 of *Lecture Notes in Comput. Sci.*, page 213228, 1998.
14. Michele Conforti and M. R. Rao. Structural properties and decomposition of linear balanced matrices. *Mathematical Programming*, 55:129–168, 1992.
15. M. Dom. Algorithmic aspects of the consecutive-ones property. *Bull. Eur. Assoc. Theor. Comput. Sci. EATCS*, 98:2759, 2009.
16. M. Dom, J. Guo, and R. Niedermeier. Approximation and fixed-parameter algorithms for consecutive ones submatrix problems. *Journal of Computer and System Sciences*, 76(3-4), 2010.
17. D.R. Fulkerson and O.A. Gross. Incidence matrices and interval graphs. *Pacific J. Math.*, 15(3):835855, 1965.
18. M. Habib, R.M. McConnell, C. Paul, and L. Viennot. Lex-bfs and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. *Theoret. Comput. Sci.*, 234(12):5984, 2000.

19. M. Hajiaghayi and Y. Ganjali. A note on the consecutive ones submatrix problem. *Information Processing Letters*, 83(3):163166, 2002.
20. Refael Hassin and Nimrod Megiddo. Approximation algorithms for hitting objects with straight lines. *Discrete Applied Mathematics*, 30(1):29 – 42, 1991.
21. Dorit S. Hochbaum and Asaf Levin. Cyclical scheduling and multi-shift scheduling: Complexity and approximation algorithms. *Discrete Optimization*, 3(4):327 – 340, 2006.
22. W.-L. Hsu. A simple test for the consecutive ones property. *J. Algorithms*, 43(1):116, 2002.
23. W.-L. Hsu and R.M. McConnell. Pc trees and circular-ones arrangements. *Theoret. Comput. Sci.*, 296(1):99116, 2003.
24. N. Korte and R.H. Mhring. An incremental linear-time algorithm for recognizing interval graphs. *SIAM J. Comput.*, 18(1):6881, 1989.
25. Lawrence T. Kou. Polynomial complete consecutive information retrieval problems. *SIAM J. Comput.*, 6(1):67–75, 1977.
26. W.-F. Lu and W.-L. Hsu. A test for the consecutive ones property on noisy data – application to physical mapping and sequence assembly. *J. Comput. Biol.*, 10:709–735, 2003.
27. R.M. McConnell. A certifying algorithm for the consecutive-ones property. In ACM Press, editor, *15th Annual ACMSIAM Symposium on Discrete Algorithms SODA '04*, page 768777, 2004.
28. S. Mecke, A. Schbel, and D. Wagner. Station location complexity and approximation. In *5th Workshop on Algorithmic Methods and Models for Optimization of Railways ATMOS '05*, Dagstuhl, Germany, 2005.
29. S. Mecke and D. Wagner. Solving geometric covering problems by data reduction. In Springer, editor, *12th Annual European Symposium on Algorithms ESA '04*, volume 3221 of *Lecture Notes in Comput. Sci.*, page 760771, 2004.
30. J. Meidanis, O. Porto, and G.P. Telles. On the consecutive ones property. *Discrete Appl. Math.*, 88:325354, 1998.
31. M. Oswald and G. Reinelt. The simultaneous consecutive ones problem. *Theoret. Comput. Sci.*, 410(2123):19861992, 2009.
32. Nikolaus Ruf and Anita Schbel. Set covering with almost consecutive ones property. *Discrete Optimization*, 1(2):215 – 228, 2004.
33. J. Tan and L. Zhang. The consecutive ones submatrix problem for sparse matrices. *Algorithmica*, 48(3):287299, 2007.
34. A.C. Tucker. A structure theorem for the consecutive 1s property. *Journal of Combinatorial Theory. Series B*, 12:153162, 1972.
35. A.F. Veinott and H.M. Wagner. Optimal capacity scheduling. *Oper Res*, 10:518–547, 1962.