# 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 $O(\mathit{\Delta}^3 m^2(m\mathit{\Delta} + n^3))$ time algorithm for this particular task for a $m \times n$ binary matrix with at most $\mathit{\Delta}$ 1-entries per row, thereby improving the $O(\mathit{\Delta}^3 m^2(mn + n^3))$ time algorithm of Dom *et al.* [17].

## 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 rows 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, 18, 19, 23–25, 28, 31]. The C1P of matrices has a long history and it plays an important role in combinatorial optimization, including application fields such as scheduling [6, 21, 22, 36], information retrieval [26], and railway optimization [29, 30, 33] (see [16] for a recent survey). Furthermore, algorithmic aspects of the C1P turn out to be of particular importance for physical mapping [2, 13, 27] and ancestral genome reconstruction [1, 12]. (see also [10, 3–5, 14, 32] 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 [11]. 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). 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 a minimum size forbidden Tucker submatrix [8].

Let us turn the C1P into an optimization problem. Recently, Dom *et al.* [17] investigated 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):

- Min-COS-C ("*Consecutive Ones Submatrix by Column Deletion*") – find a minimum-cardinality set of columns to delete such that the resulting matrix has the C1P.
- Min-COS-R ("*Consecutive Ones Submatrix by Row Deletion*") – find a minimum-cardinality set of rows to delete such that the resulting matrix has the C1P.
- Min-CO-1E ("*Consecutive Ones by Flipping 1-Entries*") – 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 [20, 34], and hence Dom *et al.* have focussed on approximation and parameterized complexity issues. To this end, they have provided a technical solution based on efficiently detecting forbidden Tucker submatrices [35]. For the sake of presentation, let us introduce these forbidden submatrices by graphs.

Let $M$ be a $m \times n$ binary matrix. Its corresponding vertex-colored bipartite graph $G(M) = (V_M, E_M)$ is defined as follows: for every row (resp. column) of $M$ there is a black (resp. white) vertex in $V_M$, and there is an edge between a black vertex $v_i$ and a white vertex $v_j$, *i.e.*, an edge between the vertices that correspond to the $i^{th}$ row and the $j^{th}$ column of $M$, if and only of $M[i, j] = 1$. Equivalently, $M$ is the reduced adjacency matrix of $G(M)$. See Figure 1 for an illustration. In the sequel, we shall speak indistinctly about binary matrices and their corresponding vertex-colored bipartite graphs. Recall now that 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. Most of the interest in this definition stems from the following theorem.

**Theorem 1 ([35], 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.*

Moreover, Tucker has characterized the binary matrices that have the C1P by a set of *forbidden submatrices*.

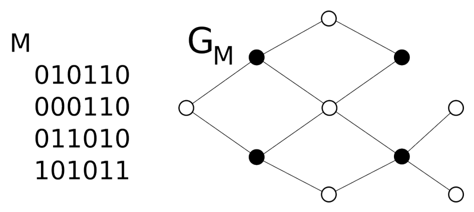

**Fig. 1.** A binary matrix and its corresponding vertex-colored bipartite graph.

**Theorem 2 ([35], 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 Figure 2.*

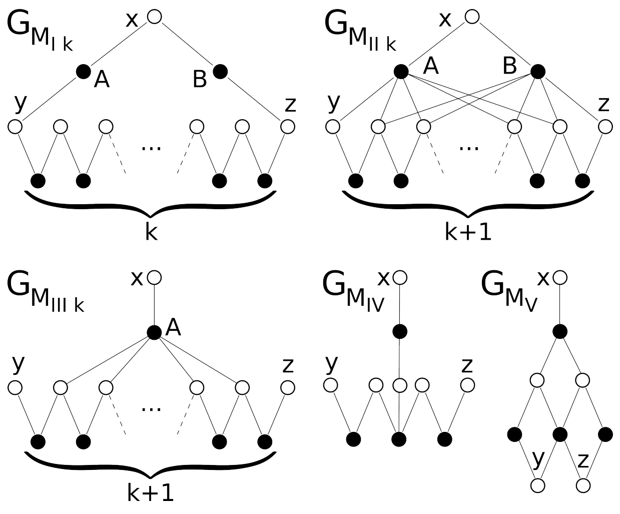

**Fig. 2.** Forbidden Tucker submatrices represented as vertex-colored bipartite graphs [35]. Black and white vertices correspond to rows and columns, respectively.

In [17], 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 sum 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 hole 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 $\Delta$ 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.

| | Dom et al. |
|---|---|
| $M_{I_k}$ and $M_{II_k}$ | $O(\Delta mn^2 + n^3)$ |
| $M_{III_k}$ | $O(\Delta^3 m^3 n + \Delta^2 m^2 n^2)$ |
| $M_{IV}$ | $O(\Delta^3 m^2 n^3)$ |
| $M_V$ | $O(\Delta^4 m^2 n)$ |
| Total | $O(\Delta^3 m^2 n(m + n^2))$ |

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 algorithm is based on shortest paths and two graph pruning techniques: *clean* and *anticlean* (to be defined in the next section). Graph pruning techniques were introduced by Conforti *et al.* [15]. One has to note that graph pruning technique 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 hole 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.

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

```
Proof. 1: guess({$x, y, z, A, B$}) and add them to $S$
2: clean($x, A, B$)
3: find a shortest path $p$ in the pruned graph between $y$ and $z$ after having removed
   $A$ and $B$
4: if $p$ exists then
5:     Add the vertices of $p$ to $S$
6:     return the induced subgraph $G(M)[S]$
7: end if
```

## 2 Fast detection of minimum size forbidden Tucker submatrices

Let us introduce the `clean` and `anticlean` cleaning operations. Let $M$ be a binary matrix and $G(M) = (V_M, E_M)$ be the corresponding vertex-colored bipartite graph. For any node $v$ of $G(M)$, `clean`$(v)$ results in the graph where any neighbor of $v$ has been deleted, *i.e.*, $G(M)[V_M \setminus N(v)]$. For any node $v$ of $G(M)$, `anticlean`$(v)$ results in the graph where any node that does not belong to the same partition nor the neighborhood of $v$ has been deleted, *i.e.*, $G(M)[V_M \setminus \{u : u \notin N(v) \textit{ and } \texttt{color}(u) \neq \texttt{color}(v)\}]$.

We now focus on the bipartite graphs that represent Tucker configurations (see Figure 2). Define the `guess`$(V \subseteq \{x, y, z, A, B\})$ operation as follows: given a Tucker configuration $\mathcal{T} \in T$, identifies by a brute-force algorithm all the vertices of $V$ among the vertices of $G(M)$. In other words, the `guess` operation tries all possible matching between vertices labeled by $x, y, z, A$ or $B$ in $\mathcal{T}$ and vertices of $G(M)$. Of particular importance, guessed vertices will never be affected (*i.e.*, deleted) by the `clean` and `anticlean` operations.

**Lemma 1.** *Let $M$ be $m \times n$ binary matrix with at most $\Delta$ 1-entries per row. One can find 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).*

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-uplets $x, y, z, A, B$ (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 hole, and hence does not have a chord. Therefore, `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 $z$. Moreover, looking for a shortest path $p$ in

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

```
Proof.  1: guess({$x, y, z, A, B$}) and add them to $S$
2: anticlean($A, B$)
3: clean($x$)
4: find a shortest path $p$ in the pruned graph between $y$ and $z$ after having removed
   $A$ and $B$
5: if $p$ exists then
6:    Add the vertices of $p$ to $S$
7:    return the induced subgraph $G(M)[S]$
8: end if
```

the pruned graph between $y$ and $z$ after having removed $A$ and $B$ ensures the minimality of the returned graph which is indeed an hole.

The guessing can be done in $O(m^2\Delta^3)$ time. Indeed, once $A$ has been identified, one can select $x$ and $y$ among the at most $\Delta$ neighbors of $A$ and then identify $B$ and one of its at most $\Delta$ neighbors as $z$ such that $x \in N(B)$ and $z \notin \{x, y\}$. For each such guessing, the cleaning of $x, A, B$ can be done in $O(\Delta + m)$ time. Finally, one can find a shortest path between $y$ and $z$ by a breadth-first search in the pruned graph after having removed $A$ and $B$ which has at most $m+n$ vertices and $\Delta m$ edges in $O(n + \Delta m)$ time. On the whole, Algorithm 1 is $O(m^2\Delta^3(n + \Delta m))$ time. □

**Lemma 2.** *Let $M$ be a $m \times n$ binary matrix with at most $\Delta$ 1-entries per row. One can find 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).*

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-uplets $x, y, z, A, B$ in Line 1, there must be at least one guess such that $x, y, z, A, B$ are indeed part of the vertices of $G'$. By definition, in $G'$, any unguessed white node is in the neighborhood of both $A$ and $B$. Thus, anticlean($A, B$) preserves $G'$ since, in $G'$, (1) $y$ which is the only white node not in the neighborhood of $B$ has been guessed and (2) $z$ which is 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 graph which is indeed $G(M_{II_k})$.

The guessing can be done in $O(m^2\Delta^3)$ time. For each such guessing, the cleaning/anticleaning of $x, A, B$ can be done in $O(n+m)$ time. Finally,

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

*Proof.*
```
1: guess($\{x, y, z, A\}$) and add them to $S$
2: anticlean($A$)
3: clean($x$)
4: find a shortest path $p$ in the pruned graph between $y$ and $z$ after having removed
   $A$
5: if $p$ exists then
6:     Add all the nodes of $p$ to $S$
7:     return the induced subgraph $G(M)[S]$
8: end if
```

one can find a shortest path between $y$ and $z$ by a breadth-first search in the pruned graph after having removed $A$ and $B$ which has at most $\Delta + n$ vertices and $\Delta m$ edges in $O(n + \Delta m)$ time. On the whole, Algorithm 2 is $O(m^2\Delta^3(n + \Delta m))$ time. □

If we compare Algorithm 1 and Algorithm 2, in both cases we are looking for a $y - z$ shortest path in the pruned graph after having removed $A$ and $B$. Moreover, if we refer to Figure 2, the final structural topology of the $y - z$ path is similar in the $M_{I_k}$ and $M_{II_k}$ matrices. Therefore, one may reasonably think that the total number of path vertices should be equal in both cases. This is not true due to different pruning techniques: cleaning in Algorithm 1 *versus* cleaning/anticleaning in Algorithm 2.

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

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-uplets $x, y, z, A$ in Line 1, there must be at least one guess such that $x, y, z, A$ are indeed part of the vertices of $G'$. By definition, in $G'$, any unguessed white node is in the neighborhood of $A$. Thus, `anticlean`$(A)$ preserves $G'$ since, in $G'$, $y$ and $z$ which are the only white nodes not in the neighborhood of $A$ have been guessed. Moreover, in $G'$, $x$ is only connected to $A$. Thus, `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 graph which is indeed $G(M_{III_k})$.

The guessing can be done in $O(m\Delta n^2)$ time. Indeed, once $A$ has been identified, one can select $x$ among the at most $\Delta$ neighbors of $A$ and then identify $y$ and $z$ among the $n$ white nodes such that $x \neq y \neq z$.

| | Dom et al. | Our contribution |
|---|---|---|
| $M_{I_k}$ and $M_{II_k}$ | $O(\Delta mn^2 + n^3)$ | $O(m^2\Delta^3(n + \Delta m))$ |
| $M_{III_k}$ | $O(\Delta^3 m^3 n + \Delta^2 m^2 n^2)$ | $O(m\Delta n^2(n + \Delta m))$ |
| $M_{IV}$ | $O(\Delta^3 m^2 n^3)$ | |
| $M_V$ | $O(\Delta^4 m^2 n)$ | |
| Overall | $O(\Delta^3 m^2(mn + n^3))$ | $O(\Delta^3 m^2(m\Delta + n^3))$ |

**Table 1.** Comparing our results with Dom *et al.* [17].

For each such guessing, the cleaning/anticleaning of $x, A$ can be done in $O(n + m)$ time. Finally, one can find a shortest path between $y$ and $z$ by a breadth-first search in the pruned graph after having removed $A$ which has at most $\Delta + n$ vertices and $\Delta m$ edges in $O(n + \Delta m)$ time. On the whole, Algorithm 3 is $O(m\Delta n^2(n + \Delta m))$ time. □

Considering $G(M_{IV})$ and $G(M_V)$, a simple brute-force search yield the following

**Lemma 4 ([17], Proposition 5.3).** *Let $M$ be a $m \times n$ binary matrix with at most $\Delta$ 1-entries per row. One can find the smallest $G(M_{IV})$ (resp. $G(M_V)$) in $G(M)$ 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.* [17].).

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

## 3 Matrices with unbounded $\Delta$

As mentioned in [17], a natural question would be to investigate the complexity of the problem when the number of 1s per row is unbounded. One can thus distinguish two subcases: the maximum number of 1s per column is bounded (say by $C$) or not. Due to space constraint, the two following results are given without proof.

**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 $O(C^2 n^3(m + C^2 n))$ time.*

**Theorem 5.** *Let $M$ be $m \times n$ binary matrix. A minimum size forbidden Tucker submatrix that occurs in $M$ can be found in $O(n^4 m^4)$ 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. Manǔch, 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, T. Stephen, U.-U. Haus, and V. You. Minimal conflicting sets for the consecutive ones property in ancestral genome reconstruction. In F. Ciccarelli and I. Miklós, editors, *Proc. 7th RECOMB Comparative Genomics Satellite Workshop (RECOMB-CG)*, volume 5817 of *Lecture Notes in Bioinformatics*, pages 48–48. Springer, 2009.
12. 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.
13. 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.
14. 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.
15. Michele Conforti and M. R. Rao. Structural properties and decomposition of linear balanced matrices. *Mathematical Programming*, 55:129–168, 1992.
16. M. Dom. Algorithmic aspects of the consecutive-ones property. *Bull. Eur. Assoc. Theor. Comput. Sci. EATCS*, 98:2759, 2009.
17. Michael Dom, Jiong Guo, and Rolf Niedermeier. Approximation and fixed-parameter algorithms for consecutive ones submatrix problems. *Journal of Computer and System Sciences*, In Press, Corrected Proof, 2009.

18. D.R. Fulkerson and O.A. Gross. Incidence matrices and interval graphs. *Pacific J. Math.*, 15(3):835855, 1965.
19. 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.
20. M. Hajiaghayi and Y. Ganjali. A note on the consecutive ones submatrix problem. *Information Processing Letters*, 83(3):163166, 2002.
21. Refael Hassin and Nimrod Megiddo. Approximation algorithms for hitting objects with straight lines. *Discrete Applied Mathematics*, 30(1):29 – 42, 1991.
22. Dorit S. Hochbaum and Asaf Levin. Cyclical scheduling and multi-shift scheduling: Complexity and approximation algorithms. *Discrete Optimization*, 3(4):327 – 340, 2006.
23. W.-L. Hsu. A simple test for the consecutive ones property. *J. Algorithms*, 43(1):116, 2002.
24. W.-L. Hsu and R.M. McConnell. Pc trees and circular-ones arrangements. *Theoret. Comput. Sci.*, 296(1):99116, 2003.
25. N. Korte and R.H. Mhring. An incremental linear-time algorithm for recognizing interval graphs. *SIAM J. Comput.*, 18(1):6881, 1989.
26. Lawrence T. Kou. Polynomial complete consecutive information retrieval problems. *SIAM J. Comput.*, 6(1):67–75, 1977.
27. 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.
28. 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.
29. 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.
30. 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.
31. J. Meidanis, O. Porto, and G.P. Telles. On the consecutive ones property. *Discrete Appl. Math.*, 88:325354, 1998.
32. M. Oswald and G. Reinelt. The simultaneous consecutive ones problem. *Theoret. Comput. Sci.*, 410(2123):19861992, 2009.
33. Nikolaus Ruf and Anita Schbel. Set covering with almost consecutive ones property. *Discrete Optimization*, 1(2):215 – 228, 2004.
34. J. Tan and L. Zhang. The consecutive ones submatrix problem for sparse matrices. *Algorithmica*, 48(3):287299, 2007.
35. A. C. Tucker. A structure theorem for the consecutive 1s property. *Journal of Combinatorial Theory. Series B*, 12:153162, 1972.
36. A.F. Veinott and H.M. Wagner. Optimal capacity scheduling. *Oper Res*, 10:518–547, 1962.