Linear-Time Algorithms for Finding Tucker Submatrices and Lekkerkerker-Boland Subgraphs[⋆]

Nathan Lindzey [⋆⋆], Ross M. McConnell [⋆⋆⋆]

Colorado State University, Fort Collins CO 80521, USA

Abstract. Tucker characterized the minimal forbidden submatrices of binary matrices that do not have the consecutive-ones property. We give a linear-time algorithm to find such a minimal one in any binary matrix that does not have the consecutive-ones property. Lekkerkerker and Boland characterized the minimal forbidden induced subgraphs for the class of interval graphs. We give a linear-time algorithm to find such a minimal one in any graph that is not an interval graph.

1 Introduction

The *intersection graph* of a collection of sets has one vertex for each set in the collection and an edge between two vertices if the corresponding sets intersect. A graph is an *interval graph* if it is the intersection graph of a collection of intervals on a line. Such a collection of intervals is known as an *interval model* of the graph. Interval graphs are an important subclass of perfect graphs [6], they have been written about extensively, and they model constraints in various combinatorial optimization and decision problems [4, 6, 14, 16]. They have a rich structure and history, and interesting relationships to other graph classes. For a survey, see [2].

For a 0-1 (binary) matrix M, let n denote the number of rows, m the number of columns, and $size(M)$ the number of rows, columns and 1's. In this paper, all matrices are binary. A sparse representation of a matrix takes $O(size(M))$ space. Such a matrix has the *consecutive-ones property* if there exists an ordering of its columns such that, in every row, the 1's are consecutive.

A *consecutive-ones matrix* is a matrix that has the consecutive-ones property, and a *consecutive-ones-ordered matrix* is a matrix where the 1's are consecutive in every row. A *clique* is a maximal induced subgraph that is a complete graph.

⋆ This work appeared in preliminary form in Lindzey, Nathan, and Ross M. McConnell. On Finding Tucker Submatrices and Lekkerkerker-Boland Subgraphs, *Lecture Notes in Computer Science*, 2013 [10].

⋆⋆ lindzey@cs.colostate.edu, Computer Science Department, Colorado State University, Fort Collins, CO, 80523-1873 U.S.A.

⋆⋆⋆ rmm@cs.colostate.edu, Computer Science Department, Colorado State University, Fort Collins, CO, 80523-1873 U.S.A.

A *clique matrix* of a graph G is a matrix that has a row for each vertex, a column for each clique, and a 1 in row i, column j if vertex i is contained in clique j. A graph is an interval graph if and only if its clique matrices have the consecutive-ones property, see, for example, [6].

In 1962, Lekkerkerker and Boland described the minimal forbidden induced subgraphs for the class of interval graphs [8], known as the *LB graphs.* These are depicted in Figure 1. Ten years later, Tucker described the minimal forbidden submatrices for consecutive-ones matrices [20]. These are the matrices that do not have the consecutive-ones property, and that satisfy the condition that deletion of any row or column results in a matrix that has the consecutive-ones property. These are depicted in Figure 2. The presence of an induced LB subgraph in a graph and the presence of a Tucker submatrix in its clique matrix are both necessary and sufficient conditions for a graph not to be an interval graph. Not surprisingly, there is a relationship between the Tucker matrices and the clique matrices of LB graphs, depicted in Figure 3.

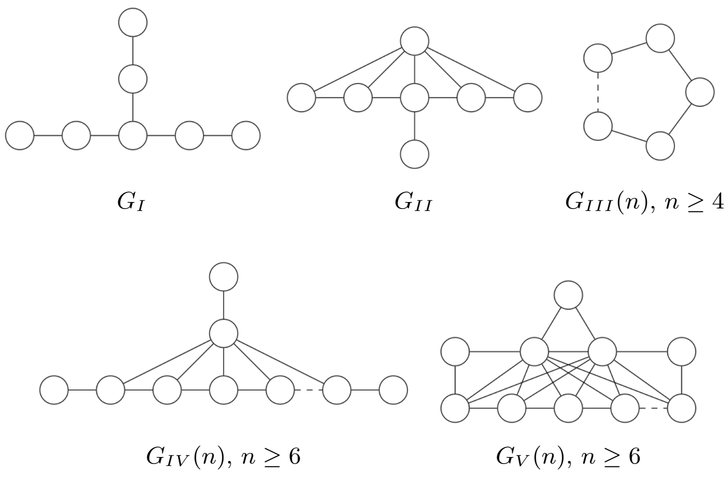


Fig. 1: The Lekkerkerker-Boland subgraphs.

In this paper, we give a linear time bound for finding an induced LB subgraph when a graph is not an interval graph. As part of our algorithm, we also give a linear-time ($O(size(M))$) bound for finding one of Tucker's submatrices in a matrix M that does not have the consecutive-ones property. This latter problem was solved previously in $O(n * size(M))$ time in [18]. An $O(\Delta^3 m^2 n(m + n^2))$ bound for finding a Tucker submatrix of minimum size is given in [5], where Δ is the maximum number of 1's in any row.

A *simplicial vertex* of a graph is a vertex that occupies a single clique; it and its neighbors are the members of the clique. In Figure 3, the square vertices are the simplicial vertices. A *chord* on a cycle or path in a graph is an edge

$M_I(k)$, $k \geq 3$

$$\begin{array}{c|cccccc|}
0 & 1 & 1 & & & & \\
1 & & 1 & 1 & & & \\
2 & & & 1 & 1 & & \\
\vdots & & & & \ddots & \ddots & \\
k{-}2 & & & & & 1 & 1 \\
k{-}1 & 1 & 0 & \cdots & 0 & 0 & 1
\end{array}$$

$M_{II}(k)$, $k \geq 4$

$$\begin{array}{c|cccccc|}
0 & 0 & 1 & 1 & \cdots & 1 & 1 \\
1 & 1 & 1 & & & & \\
2 & & 1 & 1 & & & \\
\vdots & & & \ddots & \ddots & & \\
k{-}2 & & & & 1 & 1 & 0 \\
k{-}1 & 1 & \cdots & 1 & 1 & 0 & 1
\end{array}$$

$M_{III}(k)$, $k \geq 3$

$$\begin{array}{c|ccccccc|}
0 & 1 & 1 & & & & & \\
1 & & 1 & 1 & & & & \\
2 & & & 1 & 1 & & & \\
\vdots & & & & \ddots & \ddots & & \\
k{-}2 & & & & & 1 & 1 & 0 \\
k{-}1 & 0 & 1 & \cdots & 1 & 1 & 0 & 1
\end{array}$$

M_{IV}

$$\begin{array}{c|cccccc|}
0 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 0 \\
2 & 0 & 0 & 0 & 0 & 1 & 1 \\
3 & 0 & 1 & 0 & 1 & 0 & 1
\end{array}$$

M_V

$$\begin{array}{c|ccccc|}
0 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 \\
2 & 1 & 1 & 1 & 1 & 0 \\
3 & 1 & 0 & 0 & 1 & 1
\end{array}$$

Fig. 2: The minimal forbidden submatrices for consecutive-ones matrices. Entries that have nothing in them are implicitly 0's. Since they are minimal matrices that cannot be consecutive-ones ordered, for any chosen ordering of the rows, all rows except the last can be consecutive-ones ordered.

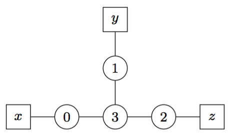
that is not on the cycle or path, but that has both of its endpoints on the cycle or path. A *chordless cycle* in a graph is a cycle on four or more vertices that has no chord, that is, it is an induced $G_{III}(n)$ for $n \geq 4$. A *chordless path* is a path that has no chord. A graph is *chordal* if it has no chordless cycle. Since $G_{III}(n)$ for $n \geq 4$ are forbidden induced subgraphs for interval graphs, interval graphs are a subclass of chordal graphs. The clique matrices of graphs can have an exponential number of columns. However, if a matrix, M, is a clique matrix of a chordal graph, then $size(M) = O(n + m)$, see [6].

An *asteroidal triple* (AT) in a graph G is a set of three vertices $\{x, y, z\}$ such that there is a path from y to z in $G - N[x]$, a path from x to z in $G - N[y]$, and a path from x to y in $G - N[z]$. Lekkerkerker and Boland showed that a chordal graph is an interval graph if and only if it has no AT. The AT's in the chordal LB subgraphs are those that are indicated by square vertices in Figure 3.

A *certifying algorithm* is an algorithm that provides, with each output, a simple-to-check proof that it has answered correctly [7, 12]. An interval model

M_{IV}

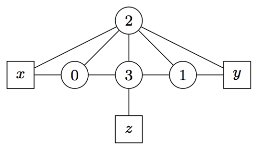
$$\begin{array}{c|cccccc} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 1 & 1 \\ 3 & 0 & 1 & 0 & 1 & 0 & 1 \\ \hline x & 1 & 0 & 0 & 0 & 0 & 0 \\ y & 0 & 0 & 1 & 0 & 0 & 0 \\ z & 0 & 0 & 0 & 0 & 1 & 0 \end{array}$$



G_I

M_V

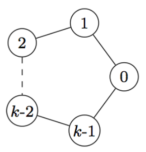
$$\begin{array}{c|ccccc} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 2 & 1 & 1 & 1 & 1 & 0 \\ 3 & 1 & 0 & 0 & 1 & 1 \\ \hline x & 0 & 1 & 0 & 0 & 0 \\ y & 0 & 0 & 1 & 0 & 0 \\ z & 0 & 0 & 0 & 0 & 1 \end{array}$$



G_{II}

$M_I(k)$

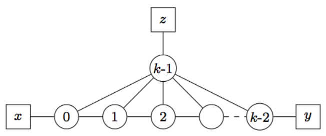
$$\begin{array}{c|ccccccc} 0 & 1 & 1 & & & & & \\ 1 & & 1 & 1 & & & & \\ 2 & & & 1 & 1 & & & \\ \vdots & & & & \ddots & \ddots & & \\ k-2 & & & & & & 1 & 1 \\ k-1 & 1 & 0 & \cdots & & 0 & 0 & 1 \end{array}$$



$G_{III}(n)$, $n \geq 4$, $k \geq 4$

$M_{III}(k-3)$

$$\begin{array}{c|ccccccc} 0 & 1 & 1 & & & & & \\ 1 & & 1 & 1 & & & & \\ 2 & & & 1 & 1 & & & \\ \vdots & & & & \ddots & \ddots & & \\ k-2 & & & & & 1 & 1 & 0 \\ k-1 & 0 & 1 & \cdots & 1 & 1 & 0 & 1 \\ \hline x & 1 & & & & & & \\ y & & & & & & 1 & \\ z & & & & & & & 1 \end{array}$$



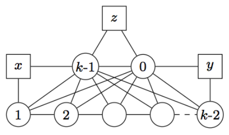
$G_{IV}(n)$, $n \geq 6$, $k \geq 3$

$M_I(3)$

$$\begin{array}{c|ccc} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 2 & 1 & 0 & 1 \\ \hline x & 0 & 1 & 0 \\ y & 0 & 0 & 1 \\ z & 1 & 0 & 0 \end{array}$$

$M_{II}(k-3)$

$$\begin{array}{c|cccccc} 0 & 0 & 1 & 1 & \cdots & 1 & 1 \\ 1 & 1 & 1 & & & & \\ 2 & & 1 & 1 & & & \\ \vdots & & & \ddots & \ddots & & \\ k-2 & & & & 1 & 1 & 0 \\ k-1 & 1 & \cdots & 1 & 1 & 0 & 1 \\ \hline x & 1 & & & & & \\ y & & & & & 1 & \\ z & & & & & & 1 \end{array}$$



$G_V(n)$, $n \geq 6$, $k \geq 4$

Fig. 3: The relationship between clique matrices of the LB graphs and the Tucker matrices. To the left of each graph is a corresponding clique matrix. The square vertices of the graphs are those that occupy a single clique; these are the *simplicial* vertices. In each case, the submatrix obtained by excluding rows corresponding to simplicial vertices is a Tucker matrix, a fact observed by Tucker [20]. $G_V(6)$ is a special case that has $M_I(3)$ as a submatrix of its clique matrix; $G_V(n)$ for $n \geq 7$ has $M_{II}(n-3)$ as a submatrix of its clique matrix.

gives a certificate that a graph is an interval graph, and an LB subgraph gives one if the graph is not an interval graph. A certifying algorithm for recognition of interval graphs was given previously in [7]. The ability to give a consecutive-ones ordering or a Tucker submatrix in linear time gives a linear-time certifying algorithm for recognizing consecutive-ones matrices. One was given previously in [11], but the certificates of that paper are neither minimal nor uniquely characterized. The presentation in that paper gives the certificate in an especially easy format for authenticating, and the LB subgraphs can easily be given in this format. Checking that it is also minimal would be more complicated and unnecessary for certifying that the graph is not an interval graph.

Therefore, interest in the algorithm of this paper will likely be motivated by the theoretical importance of the LB subgraphs, rather than by certification. Results such as those in this paper can have unanticipated algorithmic uses. For example, the algorithm of Rose, Tarjan and Lueker [15] recognizes whether a graph is a chordal graph, but it does not return a chordless cycle if it is not. The addendum given by [19] was a response to demand for an algorithm to find chordless cycles in arbitrary non-chordal graphs. Though this is useful for certification, it does not appear to have been the motivation for the addendum.

The paper also makes extensive use of generic techniques whose usefulness, to our knowledge, has not been previously recognized. An example is the extensive use of Lemma 1 in various parts of the algorithm.

Tucker recognized a relationship between the LB graphs and the Tucker matrices: a Tucker submatrix occurs in the clique matrix of every LB graph. The relationship between Tucker submatrices of clique matrices of graphs and their induced LB subgraphs, however, has a much richer structure than has been previously recognized, which is shown in the last section of the paper.

An open question is whether a minimal, unique, especially simple, or otherwise interesting special case of the certificate from [11] can be obtained by applying the algorithm of that paper to a Tucker submatrix obtained by the algorithm of the present paper. Tucker submatrices may be useful in heuristics for finding large submatrices that have the consecutive-ones property, small Tucker matrices, or identifying errors in biological data [17, 3]. Our techniques provide new tools for such heuristics.

2 Preliminaries

Given a graph G, let V denote the set of vertices and E denote the set of edges. Let n denote $|V|$ and m denote $|E|$. If $\emptyset \subset X \subseteq V$, let $G[X]$ denote the subgraph induced by X. By $G-x$ we denote $G[V \setminus \{x\}]$ and by $G-X$ we denote $G[V \setminus X]$.

Given an ordered sequence $(a_1, a_2, \ldots, a_k)$, let a *prefix* of the sequence denote a consecutive subsequence $(a_1, a_2, \ldots, a_i)$ for some i such that $0 \leq i \leq k$. (If $i = 0$, the prefix is empty.) Similarly, let a *suffix* denote $(a_i, a_{i+1} \ldots a_k)$ for some i such that $1 \leq i \leq k+1$.

We treat the rows as *sets*, where each row R is the set of columns in which the row has a 1. Consistent with conventional set notation, we will use a capital letter to denote a row and calligraphic font for a collection of rows. Since columns are elements of these sets, we will use a small letter to denote a column and a capital letter to denote a set of columns.

For a set $\mathcal{R}$ of rows and a set C of columns of a matrix, let $\mathcal{R}[C]$ denote the restriction $\{R \cap C | R \in \mathcal{R}\}$ of $\mathcal{R}$ to columns in C. Suppose $\mathcal{R}$ is the set of rows of a consecutive-ones ordered matrix and $(c_1, c_2, \ldots, c_m)$ is the ordering of the columns. In linear time, we can find, for each row, the leftmost and rightmost columns in the row. Let us call these the *left endpoint* and *right endpoint* of the row. Every graph G is the intersection graph of rows of the clique matrix, since two vertices of G are adjacent if and only if they are members of a common clique. Therefore, a consecutive-ones ordering of the clique matrix of a graph gives an interval model of the graph; the intervals are the consecutive blocks of 1's in the rows.

That interval graphs are a subclass of the class of chordal graphs follows from inclusion of the G_{III}'s among the LB subgraphs.

When a graph is chordal, the problem of deciding whether it is an interval graph reduces to the problem of deciding whether its clique matrix has the consecutive-ones property. When G is chordal, the algorithm of Rose, Tarjan and Lueker produces its cliques, hence a sparse representation of its clique matrix, in linear time [15]. Booth and Lueker gave an algorithm for finding a consecutive-ones ordering of an arbitrary matrix or else determining that it does not have the consecutive-ones property, which yielded a linear-time bound for interval graph recognition [1].

Booth and Lueker's algorithm actually achieves a stronger result, of which we make extensive use in this paper:

Lemma 1. *[1] Given a matrix M, it takes $O(size(M))$ time to find the maximal prefix of the rows of M that has the consecutive-ones property.*

Definition 1. *Let $\mathcal{R} = \{R_1, R_2, \ldots, R_p\}$ be a subset of rows of a matrix. Two rows* overlap *if their intersection is nonempty but neither is a subset of the other. The* overlap graph *of $\mathcal{R}$ is the undirected graph whose vertices are the members of $\mathcal{R}$, and where $R_i, R_j \in \mathcal{R}$ are adjacent if and only if R_i and R_j overlap. By an* overlap component *of $\mathcal{R}$, we denote the elements of $\mathcal{R}$ that make up a connected component of the overlap graph.*

Definition 2. *Suppose the overlap graph of a set $\mathcal{R}_Q$ of rows is connected. Then two columns of M are in the same* Venn class *of $\mathcal{R}_Q$ if they are elements of the same set of members of $\mathcal{R}_Q$. The* unconstrained Venn class *consists of those columns that are not in any member of $\mathcal{R}_Q$; all others are* constrained.

Lemma 2. *[13] (See Figure 5.) If the overlap graph of a set $\mathcal{R}_Q$ of rows is connected, then in all consecutive-ones orderings of $\mathcal{R}_Q$, each constrained Venn class is consecutive, the union of constrained Venn classes is consecutive, and the sequence of constrained Venn classes in the ordering is invariant, up to reversal.*

$$M_V \quad \begin{array}{c|ccccc} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 2 & 1 & 1 & 1 & 1 & 0 \\ 3 & 1 & 0 & 0 & 1 & 1 \end{array}$$

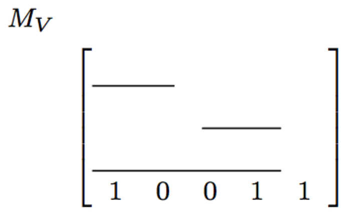


Fig. 4: When the 1's in some of the rows are consecutive, we will often represent the intervals occupied by the 1's in these rows with line segments.

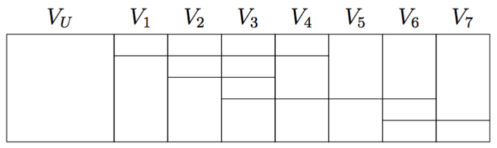


Fig. 5: A set $\mathcal{R}_Q$ of rows of a consecutive-ones matrix whose overlap graph is connected, and its Venn classes, $\{V_U, V_1, V_2, \ldots, V_7\}$. Each element of $\{V_1, V_2, \ldots, V_7\}$ is consecutive, their union is consecutive, and the order of V_1 through V_7 is uniquely constrained in any consecutive-ones ordering of $\mathcal{R}_Q$, up to reversal. These are the *constrained* classes. Columns in the unconstrained class, V_U, can go on either end of this sequence.

In this paper, we will focus on sets of columns whose overlap graphs are paths. Abusing notation slightly, when $\mathcal{P}$ is the set of rows on a path in the overlap graph, we will alternatingly treat it as an ordered sequence or as a collection of sets. For example, a *prefix* of $\mathcal{P}$ is a subpath of $\mathcal{P}$ that is either empty or contains its first row, and a *suffix* is a subpath that is either empty or contains its last row. On the other hand, we can treat it as an unordered collection of sets, as in the expression $\mathcal{P} \cup \{Z\}$, where Z is a row not in $\mathcal{P}$.

3 Breadth-first search on the overlap graph of the rows of a matrix, given a consecutive-ones ordering

One step of our algorithm is to find a shortest path of the overlap graph of the rows of a consecutive-ones ordered matrix. The difficulty in performing breadth-first search (BFS) on the overlap graph within the $O(size(M))$ time bound is that the size of the overlap graph is not $O(size(M))$. A simple example of this is when M has three columns, $n/2$ rows with 1's in the first and second column and $n/2$ rows with 1's in the second and third column. Then $size(M) = O(n)$ but its overlap graph is a complete bipartite graph with $n^2/4 = \Omega(n^2)$ edges. The algorithm is given as Algorithm 1.

Algorithm 1: `OverlapBFS`(M, S)

```
Data: A consecutive-ones ordered matrix $M$ and a starting row $S$.
Result: A BFS tree in the overlap graph on rows of $M$, rooted at $S$.
Let $\mathcal{Q}$ be an empty queue of rows;
Enqueue $S$ to $\mathcal{Q}$;
Let $\mathcal{R}'$ be the rows of $M$ other than $S$ ;
for each column $c_i$ do
    Let $\mathcal{R}_i$ be a doubly-linked list of rows in $\mathcal{R}'$ whose right endpoint is in $c_i$,
        sorted in monotonically ascending order of left endpoint;
    Let $\mathcal{L}_i$ be a doubly-linked list of rows in $\mathcal{R}'$ whose left endpoint is in $c_i$,
        sorted in monotonically descending order of right endpoint;
while $\mathcal{Q}$ is not empty do
    Dequeue a row $R$;
    Let $(c_j, c_{j+1}, \ldots, c_k)$ be the columns of $R$;
    for $i = j$ to $k - 1$ do
        $R' \longleftarrow$ the first row in $\mathcal{R}_i$ or null if $\mathcal{R}_i$ is empty;
        while $R'$ is not null and the left endpoint of $R'$ is to the left of $c_j$ do
            Remove $R'$ from the front of $\mathcal{R}_i$;
            Let $h$ be the left endpoint of $R'$;
            Remove $R'$ from $\mathcal{L}_h$;
            Assign $R$ as the parent of $R'$;
            Enqueue $R'$ to $\mathcal{Q}$;
            $R' \longleftarrow$ the first row in $\mathcal{R}_i$ or null if $\mathcal{R}_i$ is empty;
    // Left-right mirror image of previous for loop .. ;
    for $i = j + 1$ to $k$ do
        $R' \longleftarrow$ the first row in $\mathcal{L}_i$ or null if $\mathcal{L}_i$ is empty;
        while $R'$ is not null the right endpoint of $R'$ is to the right of $c_k$ do
            Remove $R'$ from the front of $\mathcal{L}_i$;
            Let $h$ be the right endpoint of $R'$;
            Remove $R'$ from $\mathcal{R}_h$;
            Assign $R$ as the parent of $R'$;
            Enqueue $R'$ to $\mathcal{Q}$;
            $R' \longleftarrow$ the first row in $\mathcal{L}_h$ or null if $\mathcal{L}_h$ is empty;
```

Lemma 3. *The parent relation assigned by* `OverlapBFS`(M, S) *(Algorithm 1) is a BFS tree, rooted at S, in the overlap graph of rows of M.*

Proof. In BFS, each time a vertex v is dequeued, the vertices that are enqueued are those that have not previously been enqueued and that are neighbors of v. It suffices to show that this is what `OverlapBFS` does.

An invariant is that for each column c_i, the rows in $\mathcal{L}_i$ are those rows that have not been enqueued and whose left endpoints are in c_i, and the rows in $\mathcal{R}_i$ are those that have not been enqueued and whose right endpoints are in c_i. This is true initially, and when a row is enqueued, it is removed from the two lists $\mathcal{R}_i$ and $\mathcal{L}_h$ or $\mathcal{L}_i$ and $\mathcal{R}_h$ that it occupies, maintaining the invariant.

A row R' that has not been enqueued properly overlaps row $R = (c_j, c_{j+1}, \ldots, c_k)$ if and only if one of the following conditions applies:

1. The left endpoint of R' is in $\{c_{j+1}, c_{j+2}, \ldots, c_k\}$ and its right endpoint is to the right of c_k;
2. The right endpoint of R' is in $\{c_j, c_{j+1}, \ldots, c_{k-1}\}$ and its left endpoint is to the left of c_j.

The unenqueued rows meeting the first condition are prefixes of $\{\mathcal{L}_{j+1}, \mathcal{L}_{j+1}, \ldots, \mathcal{L}_k\}$ because of the way these lists are sorted. Similarly, the rows meeting the second condition are prefixes of $\{\mathcal{R}_j, \mathcal{R}_{j+1}, \ldots, \mathcal{R}_{k-1}\}$. The inner `while` loops enqueue these prefixes. Thus, when R is dequeued, the algorithm enqueues precisely those rows that have not previously been enqueued and that properly overlap R, that is, the unenqueued neighbors of R in the overlap graph of rows of M.

Lemma 4. `OverlapBFS` *can be implemented so that it can find the overlap components of rows of a consecutive-ones ordered matrix M, and, for each component, a BFS tree, in $O(size(M))$ time.*

Proof. In linear time, we may label each row of a consecutive-ones ordered matrix with its left and right endpoints. Let $\mathcal{L}$ be a set of m k-tuples of integers, each in the range $\{1, 2, \ldots, n\}$. It takes $O(k(n + m))$ time to sort $\mathcal{L}$ lexicographically using radix sort [4]. We may thus radix sort the rows with right endpoint as primary sort key and left endpoint as secondary sort key to obtain the sorted lists $\mathcal{R}_i$, in $O(n + m) = O(size(M))$ time. Similarly, we may obtain the lists $\mathcal{L}_i$ in $O(size(M))$ time. Since the lists are doubly-linked, when a row is removed from $\mathcal{R}_i$, it can be removed from the list $\mathcal{L}_h$ corresponding to its left endpoint, c_h, in $O(1)$ time.

When R is dequeued, the first inner while loop takes $O(1+k)$ time to process each list $\mathcal{R}_i$, where k is the number of elements of $\mathcal{R}_i$ that get enqueued by the step, and similarly for $\mathcal{L}_i$. Processing R therefore takes $O(|R| + q)$ time, where q is the number of vertices that get enqueued when R is processed. Over all rows in the set $\mathcal{R}$ of rows of the overlap component that contains a row S, this takes time proportional to the sum of cardinalities of members of $\mathcal{R}$, since each member of $\mathcal{R}$ is enqueued once. Iteratively restarting it on a new unenqueued row S each time it finds an overlap component gives all overlap components in $O(size(M))$ time.

4 Finding a Tucker submatrix in a matrix that does not have the consecutive-ones property

Our algorithm for finding a Tucker submatrix in a matrix M that does not have the consecutive-ones property is summarized in Algorithm 2.

```
Algorithm 2: TuckerSubmatrix ($M$)
  Data: A matrix $M$ that does not have the consecutive-ones property.
  Result: The rows and columns of a Tucker submatrix.
  $M$ = TuckerRows($M$, 4) // Algorithm 3;
  if $M$ has $i \leq 4$ rows then
      // The rows of $M$ are the rows of a Tucker submatrix ;
      Find the set $C$ of column vectors that make up a Tucker submatrix;
  else
      // Every Tucker submatrix of $M$ contains the first five rows of $M$ ;
      $M \longleftarrow$ FindRows($M$) // Algorithm 4 ;
      $C \longleftarrow$ FindColumns($M$) // Algorithm 6 ;
  return $M[C]$;
```

`TuckerRows`, which it calls, takes as parameters a matrix M that does not have the consecutive-ones property and an integer k, which is 4 in this case. It returns a matrix that is an ordering of a subset of rows of M and that does not have the consecutive-ones property. If it has at most $k = 4$ rows, these are the rows of every Tucker submatrix of its returned matrix, and finding the columns in linear time is trivial. Otherwise, every Tucker submatrix of this matrix contains the first five rows, and possibly other rows.

The next procedure, `FindRows` takes as a parameter a matrix M that fails to have the consecutive-ones property and such that every Tucker submatrix of it contains the first five rows. It returns a matrix M' consisting of a subset of rows of M, where every Tucker submatrix of M' contains all rows of M'. Moreover, excluding the last row of M', its overlap graph is a path.

This matrix is then passed to `FindColumns`, which has as a precondition that its parameter satisfy the aforementioned conditions that M' satisfies. It returns a set C of columns, such that $M'[C]$ is a Tucker matrix.

4.1 `TuckerRows`

In this section, we define `TuckerRows` (Algorithm 3), which is called from Algorithm 2. It takes as a parameter a matrix M that does not have the consecutive-ones property, and a parameter k. It returns a matrix M' that is an ordering of a subset of rows of M, such that M' does not have the consecutive-ones property. If the number i of rows of M' is at most k, then every Tucker submatrix of M'

contains all i rows. Otherwise, every Tucker submatrix of M' contains the first $k+1$ rows of M', and possibly additional rows.

It runs in $O(k * size(M))$ time. We could find the rows of a Tucker submatrix in every case by calling `TuckerRows` with parameter $k = n$, but that would take $O(n * size(M))$ time, which is not linear. Since Algorithm 2 calls it with with $k = 4$, this call takes $O(size(M))$ time. The reason for setting $k = 4$ is that M_{IV} and M_V each have four rows. If the returned submatrix has five or more rows, then since every Tucker submatrix of it contains at least five rows, this excludes the possibility that any of them are M_{IV} or M_V, simplifying the problem. If the returned submatrix has four rows, this also simplifies the task of finding one within the linear time bound, since $4 = O(1)$.

The strategy of the algorithm is based on the following lemma:

Lemma 5. *If a set $\mathcal{R}'$ of rows has the consecutive-ones property and Z is a row such that $\mathcal{R} = \mathcal{R}' \cup \{Z\}$ does not, then Z is one of the rows of every Tucker submatrix in $\mathcal{R}$.*

Proof. Suppose there exists a Tucker submatrix M_T whose rows are contained in rows of $\mathcal{R}'$. Then $\mathcal{R}'$ does not have the consecutive-ones property, a contradiction.

Algorithm 3: `TuckerRows`(M, k)

```
Data: A matrix $M$ that does not have the consecutive-ones property, $k \geq 1$.
Result: Postconditions are given by Lemma 6.
$i \longleftarrow 1$;
while $i \leq k$ and $M$ has at least $i$ rows do
    $(R_1, R_2, \cdots, R_r, Z) \longleftarrow$ the minimal prefix of rows of $M$ that does not have
                                   the consecutive-ones property (Lemma 1);
    $M \longleftarrow (Z, R_1, R_2, \cdots, R_r)$;
    $i \longleftarrow i + 1$;
return $M$;
```

Lemma 6. *Suppose* `TuckerRows` *(Algorithm 3) is run with parameter k and a matrix M that does not have the consecutive-ones property. The returned submatrix M' does not have the consecutive-ones property. If M' has at most k rows, then these are the rows of every Tucker submatrix in M'. Otherwise, every Tucker submatrix in M' contains the first $k+1$ rows of M' (and possibly some other rows).*

Proof. By induction on i, M does not have the consecutive-ones property at the end of iteration i. Also by induction on i, using Lemma 5, at the end of iteration i, either M has at least i rows and every Tucker submatrix in M contains the first i rows of M, or else M has only $i-1$ rows and every Tucker submatrix in M contains these $i-1$ rows, in which case M is returned before another iteration takes place.

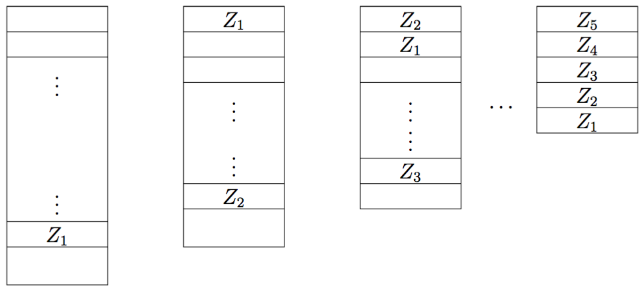


Fig. 6: The first iteration of Algorithm 3 finds the minimal prefix of the rows of M that does not have the consecutive-ones property, ending at some row Z_1. After moving Z_1 to the beginning of the matrix, the second iteration finds the minimal prefix that does not have the consecutive-ones property, ending at some row Z_2. Iterating this operation, after i iterations, we see by induction using Lemma 5 that every Tucker submatrix in the remaining rows contains every row of $\{Z_i, Z_{i-1}, \ldots, Z_1\}$. If that is all of the remaining rows, the algorithm can return them as the rows of a Tucker submatrix. If it halts after $k+1$ iterations, every Tucker submatrix in the remaining rows of M contains the first $k+1$ rows.

Lemma 7. *Algorithm 3 takes $O(k * size(M))$ time.*

Proof. It has at most $k+1$ iterations of the loop, each of which takes $O(size(M))$ time by Lemma 1.

4.2 `FindRows`

Since Algorithm 2 only calls the remaining procedures if `TuckerRows` returns a submatrix with at least five rows, we may assume the following henceforth, by Lemma 6:

- M is a matrix that does not have the consecutive-ones property and where every Tucker submatrix contains the first five rows of M.

The purpose of `FindRows` is to find the rows of a Tucker submatrix in a matrix meeting this condition. Since M_{IV} and M_V have only four rows, we may exclude them from consideration. The only Tucker submatrices we need to consider henceforth are M_I, M_{II}, and M_{III}.

Proposition 1. *The overlap graphs of $M_I(k)$, $M_{II}(k)$, and $M_{III}(k)$ are simple cycles.*

Let $\{Z_1, Z_2, \ldots, Z_5\}$ be the first five rows of M. Let us choose $Z \in \{Z_1, Z_2, \ldots, Z_5\}$, and let $\mathcal{R}'$ be the remaining rows of M, excluding Z. Since every Tucker submatrix contains Z, $\mathcal{R}'$ has the consecutive-ones property, but $\mathcal{R}' \cup \{Z\}$ does not.

Removal of one element from a chordless cycle gives a chordless path. Therefore, we seek a chordless path in the overlap graph of $\mathcal{R}'$ that has the consecutive-ones property, and such that when we add Z to the rows on the path, they no longer have the consecutive-ones property.

Definition 3. *Let $\mathcal{R}'$ be a set of rows of M such that $\mathcal{R}'$ has the consecutive-ones property, but $\mathcal{R}' \cup \{Z\}$ does not. Rows $A, B \in \mathcal{R}'$ are a* suitable pair *for Z if they are members of the same overlap component $\mathcal{R}_Q$ of $\mathcal{R}'$, each of A and B contains a 1 of Z, and in a consecutive-ones ordering of $\mathcal{R}_Q$, a 0 of Z lies in between A and B. (See Figure 7.)*

Our strategy is to find a suitable pair $\{A, B\}$ for some Z and find a shortest, hence chordless, path $\mathcal{P} = (A = R_1, R_2, \ldots, R_k = B)$ between rows A and B in the overlap graph. $\mathcal{P}$ must exist, since A and B are members of the same overlap component $\mathcal{R}_Q$. (See Figures 7 and 8.) It is easy to see by Lemma 2 that every consecutive-ones ordering of $\mathcal{R}_Q$ forces a 0 of Z between two 1's, and since these three columns are in distinct Venn classes of $\mathcal{P}$, this is true of the rows of $\mathcal{P}$ also. Therefore, $\mathcal{P}$ has the consecutive-ones property but $\mathcal{P} \cup \{Z\}$ does not, and the overlap graph of $\mathcal{P}$ is a path.

Unfortunately, the introduction of Z may introduce chords in the overlap graph between Z and rows of $\mathcal{P}$ other than its endpoints, A and B. Therefore, Proposition 1 does not imply that $\mathcal{P} \cup \{Z\}$ is the set of rows of a Tucker submatrix. We could find a smaller chordless cycle $\mathcal{P}' \cup \{Z\}$ in the overlap graph of $\mathcal{P} \cup \{Z\}$, but then we would run the risk that $\mathcal{P}' \cup \{Z\}$ would have the consecutive-ones property, defeating our effort to find a minimal set of rows that does not have the consecutive-ones property.

Our solution is to show that if we find the minimal prefix $\mathcal{P}_1$ of $\mathcal{P}$ such that $\mathcal{P}_1 \cup \{Z\}$ does not have the consecutive-ones property, and then the minimal suffix $\mathcal{P}_2$ of $\mathcal{P}_1$ such that $\mathcal{P}_2 \cup \{Z\}$ does not have the consecutive-ones property, then $\mathcal{P}_2 \cup \{Z\}$ is a minimal set of rows that does not have the consecutive-ones property. Therefore, it must be the rows of a Tucker submatrix.

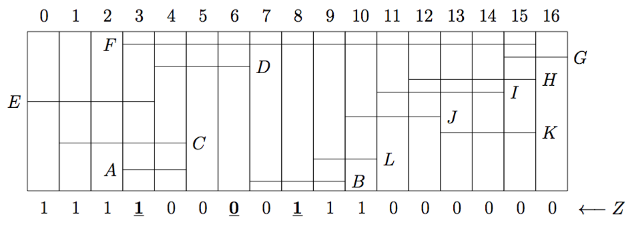


Fig. 7: Example of finding a minimal set of rows that does not have the consecutive-ones property. The set $\mathcal{R}_Q$ of rows, excluding Z, has the consecutive-ones property, but $\mathcal{R}_Q \cup \{Z\}$ does not. Rows A and B are a suitable pair for Z, as shown by the boldface 1, 0, and 1.

Example: *In Figure 7, $\mathcal{P} = (A, E, F, G, H, J, L, B)$ is a shortest path from A to B in the overlap graph of $\mathcal{R}_Q$. In a consecutive-ones ordering of $\mathcal{P}$ (Figure 8), the 0 in column 6 is forced to go between the 1's in columns 3 and 8 because of where A and B must be placed in a consecutive-ones ordering of $\mathcal{P}$. Therefore, $\mathcal{P} \cup \{Z\}$ does not have the consecutive-ones property.*

Next, (A, E, F, G, H, J) is the smallest prefix $\mathcal{P}_1$ of $\mathcal{P}$ such that $\mathcal{P}_1 \cup \{Z\}$ does not have the consecutive-ones property, and (F, G, H, J) is the smallest suffix $\mathcal{P}_2$ of $\mathcal{P}_1$ such that $\mathcal{P}_2 \cup \{Z\}$ does not have the consecutive-ones property. $\mathcal{P}_2 \cup \{Z\} = \{F, G, H, J, Z\}$ gives the rows of a Tucker submatrix (Figure 9).

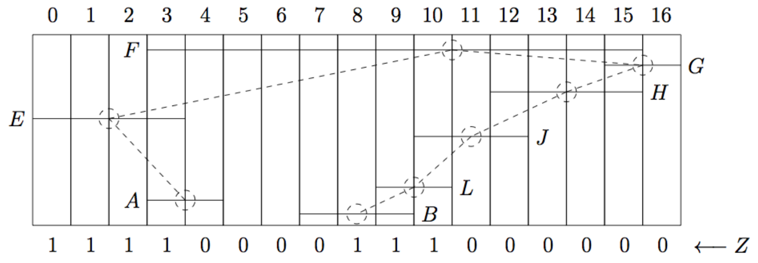


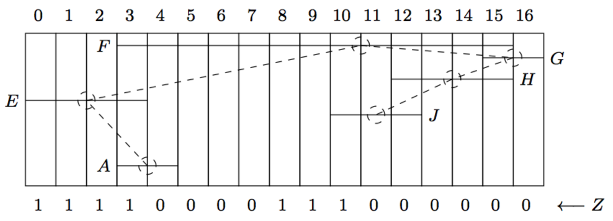
Fig. 8: A shortest, hence chordless, path $\mathcal{P}$ in the overlap graph of Figure 7 between A and B.

A shortest path $(R_1, R_2, \ldots, R_k)$ between a suitable pair A and B can be found efficiently using `OverlapBFS`. Finding a minimal prefix whose union with $\{Z\}$ does not have the consecutive-ones property reduces to finding the minimal prefix of $(Z, R_1, R_2, \ldots, R_k)$ that does not have the consecutive-ones property. Finding a minimal suffix whose union with $\{Z\}$ does not have the consecutive-ones property is solved similarly. These problems can be solved efficiently by Lemma 1. We give an algorithm for finding a suitable pair below. The procedure is summarized as Algorithm 4, where $\mathcal{P} = (R_1, R_2, \ldots, R_k)$, $\mathcal{P}_1 = (R_1, R_2, \ldots, R_j)$, and $\mathcal{P}_2 = (R_i, R_{i+1}, \ldots, R_j)$.

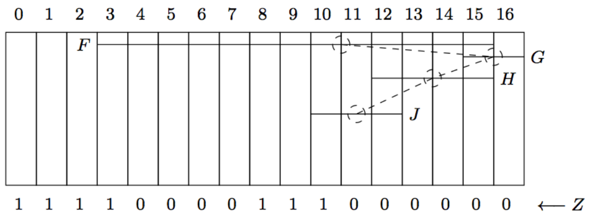
Lemma 8. `FindRows` *returns a submatrix M' that does not have the consecutive-ones property and where every Tucker submatrix in M' contains all rows of M'. If p is the number of rows of M', then the overlap graph of the first $p - 1$ rows is a path.*

Proof. In the proof of correctness of `SuitablePair` (Algorithm 5), below, we show that (A, B, Z) exists. Let $\mathcal{P} = (A = R_1, R_2, \ldots, R_k = B)$. Since A and B lie in the same overlap component, $\mathcal{R}_Q$, $\mathcal{P}$ exists. It is a shortest path, hence a chordless path in the overlap graph. The first $p-1$ rows of the returned submatrix is a subpath of $\mathcal{P}$, so their overlap graph is a path.

Since $\mathcal{R}_Q$ has the consecutive-ones property, so does $\mathcal{P}$, and any consecutive-ones ordering of $\mathcal{R}_Q$ is a consecutive-ones ordering of $\mathcal{P}$. Every consecutive-ones



$$\begin{array}{c|cccccc} & 2 & 7 & 10 & 12 & 15 & 16 \\ \hline F & 0 & 1 & 1 & 1 & 1 & 0 \\ G & 0 & 0 & 0 & 0 & 1 & 1 \\ H & 0 & 0 & 0 & 1 & 1 & 0 \\ J & 0 & 0 & 1 & 1 & 0 & 0 \\ Z & 1 & 0 & 1 & 0 & 0 & 0 \end{array}$$



$$\begin{array}{c|cccccc} & 16 & 15 & 12 & 10 & 2 & 7 \\ \hline G & 1 & 1 & 0 & 0 & 0 & 0 \\ H & 0 & 1 & 1 & 0 & 0 & 0 \\ J & 0 & 0 & 1 & 1 & 0 & 0 \\ Z & 0 & 0 & 0 & 1 & 1 & 0 \\ F & 0 & 1 & 1 & 1 & 0 & 1 \end{array}$$

Fig. 9: In Figure 8, (A, E, F, G, H, J) is the minimal prefix $\mathcal{P}_1$ of $\mathcal{P}$ such that $\mathcal{P}_1 \cup \{Z\}$ does not have the consecutive-ones property, and (F, G, H, J) is the minimal suffix $\mathcal{P}_2$ of $\mathcal{P}_1$ such that $\mathcal{P}_2 \cup \{Z\}$ does not have the consecutive-ones property. These can be found efficiently, by Lemma 1. Then $\mathcal{P}_2$ is a minimal subpath of $\mathcal{P}$ whose union with Z fails to have the consecutive-ones property, hence $\mathcal{P}_2 \cup \{Z\}$ gives the rows of a Tucker submatrix. The matrix at the top right is the submatrix we obtain after identifying the columns (Algorithm 6, below), and the permutation of rows and columns given at the bottom right reveals that it is an instance of $M_{III}(5)$.

Algorithm 4: `FindRows`(M)

```
Data: A matrix $M$ that does not have the consecutive-ones property and such
      that every Tucker submatrix contains the first five rows of $M$.
Result: A submatrix $M'$ of rows that does not have the consecutive-ones
        property, every Tucker submatrix of $M'$ contains all rows of $M'$, and,
        excluding the last row, the overlap graph of rows of $M'$ is a path.
Let $(A, B, Z) \longleftarrow$ SuitablePair$(M)$ (Algorithm 5);
Let $\mathcal{R}'$ be the rows of $M$ excluding $Z$;
$(A = R_1, R_2, \cdots, R_k = B) \longleftarrow$ a shortest path from $A$ to $B$ in the overlap
    graph of $\mathcal{R}'$ (Algorithm 1);
$(Z, R_1, R_2, \cdots, R_j) \longleftarrow$ the minimal prefix of $(Z, R_1, R_2, \cdots, R_k)$ that does not
    have the consecutive-ones property (Lemma 1);
$(Z, R_j, R_{j-1}, \ldots, R_i) \longleftarrow$ the minimal prefix of $(Z, R_j, R_{j-1}, \cdots, R_1)$ that
    does not have the consecutive-ones property (Lemma 1);
Return $(R_i, R_{i+1}, \cdots, R_j, Z)$;
```

ordering of $\mathcal{R}_Q$ forces the 0 that lies between A and B to lie between the two 1's in A and B, by Lemma 2. This is true of at least one consecutive-ones ordering of $\mathcal{P}$. The order of Venn classes of $\mathcal{P}$ is uniquely determined up to reversal, and the 0 that lies between A and B and the 1's in A and B are in different constrained Venn classes of $\mathcal{P}$, so this 0 lies between the two 1's in every consecutive-ones ordering of $\mathcal{P}$. $\mathcal{P} \cup \{Z\}$ does not have the consecutive-ones property.

Since $\mathcal{P} \cup \{Z\}$ does not have the consecutive-ones property, the shortest prefix $\mathcal{P}_1$ of $\mathcal{P}$ such that $\mathcal{P}_1 \cup \{Z\}$ does not have the consecutive-ones property exists. By the definition of $(Z, R_1, R_2, \ldots, R_j)$, $\mathcal{P}_1 = (R_1, R_2, \ldots, R_j)$. Since $\mathcal{P}_1 \cup \{Z\}$ does not have the consecutive-ones property, the shortest suffix $\mathcal{P}_2$ of $\mathcal{P}_1$ such that $\mathcal{P}_2 \cup \{Z\}$ does not have the consecutive-ones property exists. By the definition of $(Z, R_j, R_{j-1}, \ldots, R_i)$, $\mathcal{P}_2 = (R_i, R_{i+1}, \ldots, R_j)$.

Suppose there is a proper subset $\mathcal{R}'$ of the rows on $\mathcal{P}_2$ such that $\mathcal{R}' \cup \{Z\}$ does not have the consecutive-ones property. Since $\mathcal{P}_2$ is a shortest path, it is a chordless path, so $\mathcal{R}'$ is a subpath of $\mathcal{P}_2$ by Proposition 1. Let $\mathcal{R}_1 = (R_1, R_2, \ldots, R_{j-1})$. This is the result of removing the last row from $\mathcal{P}_1$. Let $\mathcal{R}_2 = (R_{i+1}, R_{i+2}, \ldots, R_j)$. This is the result of removing the first row from $\mathcal{P}_2$. By the minimality of $\mathcal{P}_1$ and $\mathcal{P}_2$, $\mathcal{R}_1 \cup \{Z\}$ and $\mathcal{R}_2 \cup \{Z\}$ have the consecutive-ones property. Since $\mathcal{R}'$ is a proper subpath of $\mathcal{P}_2$, $\mathcal{R}' \subseteq \mathcal{R}_1$ or $\mathcal{R}' \subseteq \mathcal{R}_2$, so $\mathcal{R}' \cup \{Z\}$ has the consecutive-ones property, contradicting our assumption that it does not. Therefore, $\mathcal{P}_2 \cup \{Z\}$ is a minimal set of rows that does not have the consecutive-ones property. It must be a minimal set of rows that contains a Tucker submatrix.

4.3 Finding a suitable pair

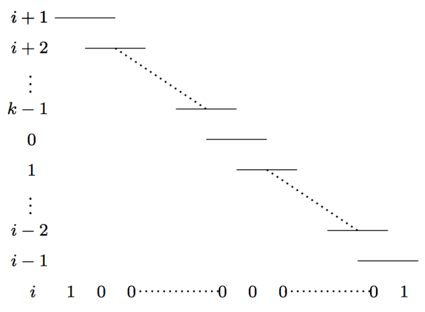
The procedure `SuitablePair`, which is called from `FindRows` (Algorithm 4), takes as a parameter a matrix M that does not have the consecutive-ones property and such that every Tucker submatrix in M contains the first five rows.

Lemma 9. *Suppose a matrix M fails to have the consecutive-ones property, and every Tucker submatrix contains the rows $\{Z_1, Z_2, Z_3, Z_4, Z_5\}$. For each $Z_i \in \{Z_1, Z_2, Z_3, Z_4, Z_5\}$, let $\mathcal{R}_i$ be the rows of M, excluding Z_i. For at least one of the five choices of Z_i, $\mathcal{R}_i$ has an overlap component with a suitable pair A, B for Z_i.*

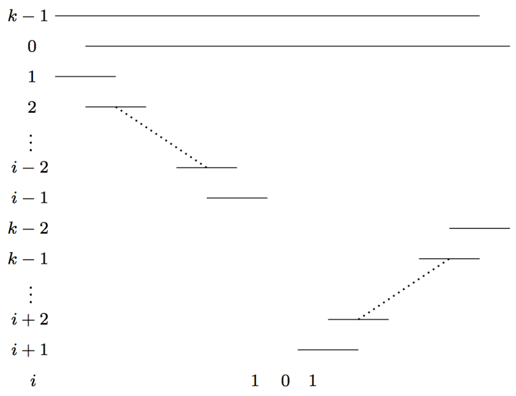
Proof. (See Figure 10.) Let M_T be a Tucker submatrix in M. Since M_T must have at least five rows, it is an instance of $M_I(k)$, $M_{II}(k)$ or $M_{III}(k)$ for $k \geq 5$.

For each choice of Z_i, $\mathcal{R}_i$ has the consecutive-ones property, since every Tucker submatrix contains Z_i as one of its rows. Let $\mathcal{R}_T$ be the rows of M_T, excluding Z_i. A consecutive-ones ordering of $\mathcal{R}_i$ imposes a consecutive-ones ordering on $\mathcal{R}_T$. Since the overlap graph of $\mathcal{R}_T$ is connected, by Proposition 1, the ordering of Venn classes of $\mathcal{R}_T$ is unique, up to reversal, by Lemma 2.

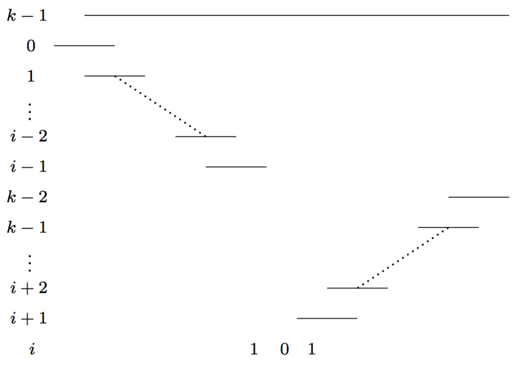
Let the rows of M_T be numbered as in Figure 2. As shown in Figure 10, unless Z_i contains row $i \in \{0, 1, k-2, k-1\}$ of M_T, rows $i-1$ and $i+1$ of M_T are a suitable pair for Z_i. There are at most four of the five choices of Z_i that can fail to have a suitable pair in $\mathcal{R}_i$.



M_I



M_{II}



M_{III}

Fig. 10: Consecutive-ones orderings of all but row i for any $i \in \{2, 3, \ldots, k-3\}$ in each of M_I, M_{II} and M_{III} gives rows $i-1$ and $i+1$ as a suitable pair for row i by Definition 3

Lemma 10. *Suppose a set $\mathcal{R}'$ of rows is consecutive-ones ordered, but but $\mathcal{R}' \cup \{Z\}$ does not have the consecutive-ones property. Let $\mathcal{R}_Q$ be the rows of an overlap component of $\mathcal{R}'$. Let A' be a row with the leftmost right endpoint such that A' contains a column that has a 1 in row Z, and let B' be a row with the rightmost left endpoint such that B' contains a column that has a 1 of Z. Then A' and B' are a suitable pair for Z if A' and B' are disjoint, and a column that has a 0 in row Z occurs in between A' and B'. Otherwise no suitable pair for Z exists in $\mathcal{R}_Q$.*

Proof. If A' and B' satisfy the conditions, they are a suitable pair for Z, by definition. Conversely, suppose there exist rows $A, B \in \mathcal{R}_Q$ that are a suitable pair for Z. Suppose without loss of generality that A is to the left of B. Then A' can be substituted for A and B' can be substituted for B, and the 0 of row Z that lies between A and B will also lie between A' and B', hence $\{A', B'\}$ is a suitable pair for Z.

The procedure is given as Algorithm 5.

Algorithm 5: `SuitablePair`(M)

```
Data: A matrix $M$ that does not have the consecutive-ones property, and in
      which every Tucker submatrix contains the first five rows of $M$
Result: A row $Z$ and a suitable pair $A, B$ for $Z$ (Definition 3).
Let $(Z_1, Z_2, \ldots, Z_5)$ be the first five rows of $M$ ;
for $i = 1$ to 5 do
    Let $\mathcal{R}_i$ be the rows of $M$, excluding $Z_i$;
    Let $M'$ be a consecutive-ones ordering of $\mathcal{R}_i$;
    Use OverlapBFS to find the overlap components of $M'$ (Algorithm 1);
    for each overlap component $\mathcal{R}_Q$ of $M'$ do
        Let $A'$ be the member of $\mathcal{R}_Q$ with the leftmost right endpoint among
            rows that contain a 1 of $Z_i$ ;
        Let $B'$ be the member of $\mathcal{R}_Q$ with the rightmost left endpoint among
            rows that contain a 1 of $Z_i$;
        if a 0 of $Z_i$ occurs in between $A'$ and $B'$ then
            return $(A', B', Z_i)$;
```

Lemma 11. `SuitablePair`(M) *returns a triple (A, B, Z) such that $\{A, B\}$ are a suitable pair for Z.*

Proof. By Lemma 9, a suitable pair exists for one of the five choices $Z_i \in \{Z_1, Z_2, \ldots, Z_5\}$. By Lemma 10, the procedure finds a suitable pair $\{A', B'\}$ for Z_i returning (A', B', Z_i).

Lemma 12. `SuitablePair`(M) *takes $O(size(M))$ time.*

Proof. On each iteration i, we may find a consecutive-ones ordering of $\mathcal{R}_i$ in $O(size(M))$ time. We may then label all columns of M with the value of Z_i in the column in $O(size(M))$ time. The calls to `OverlapBFS` take $O(size(M))$ time by Lemma 4. Since each column is labeled with the value of Z that it contains, it takes $O(|R|)$ time to find whether a set R of columns contains a 1 in Z or a 0 in Z. It takes time proportional to the sum of cardinalities of rows in an overlap component to find A' and B', and to determine whether the columns in the interval between A' and B' contain a 0 in Z. Over all connected components, this check takes $O(size(M))$ time.

Therefore, over the five iterations, the algorithm takes $O(5 * size(M)) = O(size(M))$ time.

Lemma 13. `FindRows`(M) *(Algorithm 4) takes* $O(size(M))$ *time.*

Proof. The call to `SuitablePair` takes $O(size(M))$ time by Lemma 12. It takes $O(size(M))$ time to find a shortest path from A to B by Lemma 4. It takes $O(size(M))$ time to find each of $(Z, R_1, R_2, \ldots, R_j)$ and $(Z, R_j, R_{j-1}, \ldots, R_i)$ by Lemma 1.

4.4 `FindColumns`

In this section, we develop `FindColumns` (Algorithm 6), which is called from Algorithm 2. Since it is called on a matrix M that is returned by `FindRows`, we may assume henceforth that M satisfies the conditions of Lemma 8, that is, that it fails to have the consecutive-ones property, every Tucker submatrix of M contains all rows of M, and, excluding the last row, Z, the overlap graph of the rows of M is a path, which we will denote by $\mathcal{P}'$.

Definition 4. *(See Figure 11.) Let $\mathcal{R}_Q$ be a set of rows such that the overlap graph of $\mathcal{R}_Q$ is connected, let $(c_1, c_2, \ldots, c_k)$ be the left-to-right order of columns in a consecutive-ones ordering of $\mathcal{R}_Q$, and let Z be a row that is not in $\mathcal{R}_Q$. A* 1-0-1 configuration *is a sequence of three columns (c_h, c_i, c_j) in three separate constrained Venn classes of $\mathcal{R}_Q$ such that $h < i < j$, c_h contains a 1, c_i contains a 0, and c_j contains a 1 in row Z. A* 0-1-0 configuration *for Z is defined in the same way, except that c_h contains a 0, c_i contains a 1, and c_j contains a 0 in row Z.*

The sufficiency of the following is implicit in Booth and Lueker's algorithm. The necessity in the case where the overlap graph of the rows is observed in [11].

Lemma 14. *If $\mathcal{R}_Q$ is a set of rows that has the consecutive-ones property and whose overlap graph is connected, and Z is a row not in $\mathcal{R}_Q$, then $\mathcal{R}_Q \cup \{Z\}$ fails to have the consecutive-ones property if and only if one of the following cases applies:*

1. *$\mathcal{R}_Q$ has a 1-0-1 configuration for Z;*
2. *$\mathcal{R}_Q$ has a 0-1-0 configuration for Z and Z has a 1 in the unconstrained Venn class of $\mathcal{R}_Q$.*

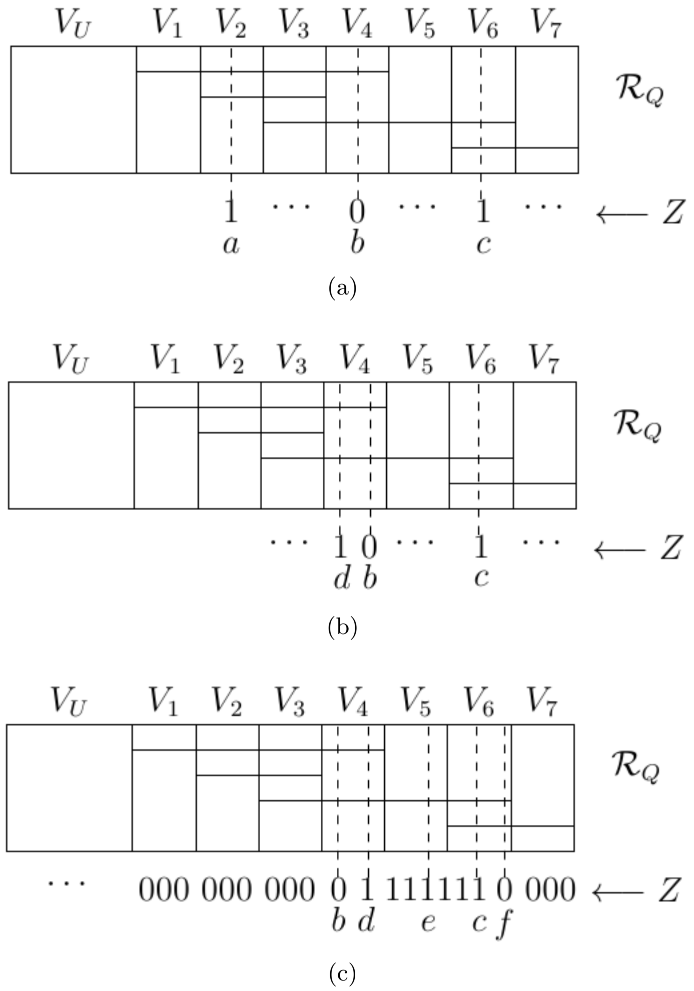

V_U V_1 V_2 V_3 V_4 V_5 V_6 V_7
$\mathcal{R}_Q$
1 ⋯ 0 ⋯ 1 ⋯ ⟵ Z
a b c
(a)
V_U V_1 V_2 V_3 V_4 V_5 V_6 V_7
$\mathcal{R}_Q$
⋯ 1 0 ⋯ 1 ⋯ ⟵ Z
d b c
(b)
V_U V_1 V_2 V_3 V_4 V_5 V_6 V_7
$\mathcal{R}_Q$
⋯ 000 000 000 0 1 111111 0 000 ⟵ Z
b d e c f
(c)

Fig. 11: (See Lemma 14). $\mathcal{R}_Q$ is a set of rows that has the consecutive-ones property and whose overlap graph is connected, and Z is an additional row. A 1-0-1 configuration for Z is three columns in a consecutive-ones ordering of $\mathcal{R}_Q$, such as (a, b, c) in the top figure. They must reside in three separate constrained Venn classes of $\mathcal{R}_Q$. The 1-0-1 configuration proves that $\mathcal{R}_Q \cup \{Z\}$ does not have the consecutive-ones property, since the 0 in b is forced to be between the two 1's in a and c in every consecutive-ones ordering of $\mathcal{R}_Q$, by Lemma 2. Columns (d, b, c) in the middle figure are not a 1-0-1 configuration, since they do not reside in three separate Venn classes. Since a Venn class can be reordered to give a new consecutive-ones ordering, they do not exclude a consecutive-ones ordering such as one that is consistent with the bottom figure. A 0-1-0 configuration is defined similarly, and (b, e, f) is an example in the bottom figure. The 0-1-0 configuration proves that $\mathcal{R}_Q \cup \{Z\}$ does not have the consecutive-ones property if if V_U contains a 1 of Row Z: the 1 in e is separated from it by 0's in b and f in every consecutive-ones ordering of $\mathcal{R}_Q$.

Proof. Let $(V_1, V_2, \ldots, V_k)$ be the ordering of the constrained Venn classes in a consecutive-ones ordering of $\mathcal{R}_Q$. Let C be the columns of M and let $C_Q = \bigcup \mathcal{R}_Q = \bigcup_{i=1}^{k} V_i$. Note that $C \setminus C_Q$ is the unconstrained Venn class. By Lemma 2, the ordering of $(V_1, V_2, \ldots, V_k)$ unique, up to reversal, C_Q is consecutive, and the columns within each Venn class can be reordered arbitrarily to obtain a consecutive-ones ordering of $\mathcal{R}_Q$.

Suppose one of the conditions applies for Z. If the first condition applies, then, since the three columns of the 1-0-1 configuration lie in distinct members of $\{V_1, V_2, \ldots, V_k\}$, the 0 of the configuration is forced between the two 1's in every consecutive-ones ordering of $\mathcal{R}_Q$, hence there can be no consecutive-ones ordering of $\mathcal{R}_Q \cup \{Z\}$. Similarly, if the second condition applies, the 1 of the 0-1-0 configuration is forced between the two zeros in every consecutive-ones ordering of $\mathcal{R}_Q$, so in every consecutive-ones ordering of $\mathcal{R}_Q$, it is separated from the 1 in the unconstrained class by one of the two zeros in the configuration. Therefore, $(\mathcal{R}_Q \cup \{Z\})$ does not have the consecutive-ones property.

Conversely, suppose neither of the two conditions applies. If C_Q contains no 1's of Z, then the columns in $C \setminus C_Q$ can be freely ordered to give a consecutive-ones ordering. Assume henceforth that C_Q contains a 1 of Z.

If $|\mathcal{R}_Q| = 1$, then $(\mathcal{R}_Q \setminus Z, \mathcal{R}_Q \cap Z, Z \setminus \mathcal{R}_Q)$ is a consecutive-ones ordering. If $|\mathcal{R}_Q| > 1$, then since two rows that are adjacent in the overlap graph have three Venn classes, the number k of Venn classes is at least 3. To avoid a 1-0-1 configuration for Z, the Venn classes where Z has 1's must be a consecutive block $(V_i, V_{i+1}, \ldots, V_j)$ in $(V_1, V_2, \ldots, V_k)$, and only V_i and V_j can have both 0's and 1's in Z. If $V_i = V_j$, V_i can be freely ordered to give a consecutive-ones ordering of $(\mathcal{R}_Q \cup \{Z\})[C_Q]$. If $i < j$, the columns in V_i can be freely ordered so that its its 1's in Z are consecutive with V_{i+1} and the columns in V_j can be freely ordered so that its 1's in Z are consecutive with V_{j-1}, giving a consecutive-ones ordering of $(\mathcal{R}_Q \cup \{Z\})[C_Q]$.

If Z has no 1 in $C \setminus C_Q$, this is a consecutive-ones ordering of $\mathcal{R}_Q \cup \{Z\}$. Therefore, suppose there is no 1-0-1 configuration and Z has a 1 in some column $c' \in C \setminus C_Q$. Because of c' and because the second condition of Lemma 14 does not apply, there is no 0-1-0 configuration for Z. To avoid a 0-1-0 configuration, it must be that $(V_i, V_{i+1}, \ldots, V_j)$ is a prefix or a suffix of $(V_1, V_2, \ldots, V_k)$. Without loss of generality, suppose it is a prefix, namely, $(V_1, V_2, \ldots V_j)$. The columns in V_j can be freely ordered so that those that contain 1's of Z are to the left of those that contain 0's, and the columns containing 1's in unconstrained class can be placed so that they are consecutive with V_1, giving a consecutive-ones ordering of $\mathcal{R}_Q \cup \{Z\}$.

Lemma 15. *Let M be a matrix that does not have the consecutive-ones property, where every Tucker submatrix contains all rows of M, and where, excluding the last row Z of M, the overlap graph of rows of M is a path, $\mathcal{P}'$. Let C be be a subset of the columns of M. Then $M[C]$ is a Tucker submatrix of M if and only if C is a minimal set of columns such that the following conditions apply:*

1. *The overlap graph of $\mathcal{P}'[C]$ is connected;*

2. $(\mathcal{P}' \cup \{Z\})[C]$ *satisfies one of the conditions of Lemma 14 for* Z*.*

Proof. Since the overlap graph of $\mathcal{P}'$ is a path, it is connected. Since $\mathcal{P}' \cup \{Z\}$ does not have the consecutive-ones property, it must satisfy one of the conditions of Lemma 14 for Z. If $\mathcal{P}'[C'']$ is connected and $(\mathcal{P}' \cup \{Z\})[C'']$ satisfies one of the two conditions of the lemma, then $(\mathcal{P}' \cup \{Z\})[C'']$ contains a Tucker submatrix since it does not have the consecutive-ones property. Conversely, let M_T be a Tucker submatrix contained in $\mathcal{P}' \cup \{Z\}$. By Lemma 8, $\mathcal{P}' \cup \{Z\}$ is a minimal set of rows that contains a Tucker submatrix, so the rows of M_T are $\mathcal{P}' \cup \{Z\}$. Let C' be the columns of M_T. Since the overlap graph of a Tucker submatrix with at least five rows is a chordless cycle, by Proposition 1, the overlap graph of $\mathcal{P}'[C']$ is a chordless path. Therefore, C' satisfies the conditions of Lemma 14, and so does every set C'' of columns such that $C' \subset C''$.

Therefore, necessary and sufficient conditions for $(\mathcal{P}' \cup \{Z\})[C'']$ to contain a Tucker submatrix is for the overlap graph of $\mathcal{P}'[C'']$ to be connected and for one of the conditions of Lemma 14 to apply for Z in $(\mathcal{P}' \cup \{Z\})[C'']$. A minimal such set C is a minimal set of columns that contains the columns of a Tucker submatrix in $\mathcal{P}' \cup \{Z\}$. Since $\mathcal{P}' \cup \{Z\}$ is its set of rows, $(\mathcal{P}' \cup \{Z\})[C]$ is a Tucker submatrix.

Our algorithm for finding the columns of a Tucker submatrix removes columns one at a time from the set C of columns, except when doing so would undermine the requirements of Lemma 15 on rows of $(\mathcal{P}' \cup \{Z\})[C]$. To obtain a linear time bound, we must describe an efficient test of whether the removal of a column would undermine one of the requirements of Lemma 15.

Definition 5. *Let* $(R_1, R_2, \ldots, R_h)$ *be the sequence rows of* $\mathcal{P}'$*. For* $i \in \{1, 2, \ldots, h-1\}$*, let* $\mathcal{A}_i = \{R_i \setminus R_{i+1}, R_i \cap R_{i+1}, R_{i+1} \setminus R_i\}$*. Let* $\mathcal{A} = \bigcup_{i=1}^{h-1} \mathcal{A}_i$*.*

Lemma 16. $\sum_{i=1}^{h-1} (|R_i \setminus R_{i+1}| + |R_i \cap R_{i+1}| + |R_{i+1} \setminus R_i|) = O(size(M))$*.*

Proof. For each $i \in \{1, 2, \ldots, h-1\}$, the members of $\{R_i \setminus R_{i+1}, R_i \cap R_{i+1}, R_{i+1} \setminus R_i\}$ are disjoint and each is a subset of R_i or of R_{i+1}. The sum is at most twice the sum of cardinalities of members of $\{R_1, R_2, \ldots, R_h\}$.

Lemma 17. *Let* $C \subseteq C_Q$*. The overlap graph of* $\mathcal{P}'[C]$ *is connected if and only if every element of* $\mathcal{A}$ *contains an element of* C*.*

Proof. For $i \in \{1, 2, \ldots, h-1\}$, $R_i[C]$ and $R_{i+1}[C]$ overlap if and only if each member of $\mathcal{A}_i$ contains an element of C. Since the overlap graph of $\mathcal{P}' = (R_1, R_2, \ldots, R_h)$ is a chordless path, the result follows.

Lemma 18. *If the overlap graph of* $\mathcal{R}_Q$ *is connected,* $\mathcal{R}_Q$ *is consecutive-ones ordered, and* Z *is a row not in* $\mathcal{R}_Q$*, it takes* $O(size(M))$ *time either to find a 1-0-1 configuration for* Z *in* $\mathcal{R}_Q \cup \{Z\}$ *or else to determine that no such configuration exists. Similarly, it takes* $O(size(M))$ *time either to find a 0-1-0 configuration for* Z *in* $\mathcal{R}_Q \cup \{Z\}$*, or else to determine that no such configuration exists.*

Proof. Let $(c_1, c_2, \ldots, c_k)$ be the consecutive-ones ordering of $\mathcal{R}_Q$. It takes $O(size(M))$ time to partition $(c_1, c_2, \ldots, c_k)$ into intervals corresponding to its constrained Venn classes; the boundaries of the partition classes occur between consecutive columns such that the first column is the right endpoint or the second column is a left endpoint of a member of $\mathcal{R}_Q$.

Label each column of $(c_1, c_2, \ldots, c_k)$ with the value of Z in the column. Suppose there exists a 1-0-1 configuration (c_h, c_i, c_j) for Z. If c_p is the leftmost column of $(c_1, \ldots, c_k)$ that contains a 1 of Z, then (c_p, c_i, c_j) is a 1-0-1 configuration where $p \leq h$. Find the next column c_q to the right of c_p that resides in a different Venn class from c_p and contains a 0 of Z. Because (c_p, c_i, c_j) is a 1-0-1 configuration, c_q exists and $q \leq i$. Then (c_p, c_q, c_j) is a 1-0-1 configuration. Find the next column c_r to the right of c_q that resides in a different Venn class from c_q and contains a 1 of Z. Because (c_p, c_q, c_j) is a 1-0-1 configuration, c_r exists. Then (c_p, c_q, c_r) is a 1-0-1 configuration. The procedure always succeeds in producing a 1-0-1 configuration if one exists. Conversely, if c_p, c_q and c_r exist, then a 1-0-1 configuration exists, since (c_p, c_q, c_r) is an example of one.

The procedure takes $O(k)$ time to find c_p and scan rightward looking for c_q and c_r, once $(c_1, c_2, \ldots, c_k)$ and its partition into constrained Venn classes is known. By symmetry of the treatment of 0's and 1's of Z, and the second statement of the lemma also follows.

The implementation of `FindColumns` is based on these tests, and is given as Algorithm 6.

Lemma 19. `FindColumns`(M) *returns a Tucker submatrix of* M.

Proof. The initial matrix contains a Tucker submatrix. By Lemma 17, the tests applied before a column is removed ensure that the conditions of Lemma 15 continue to be satisfied by the final submatrix $(\mathcal{P}' \cup \{Z\})[C]$ returned by the procedure. Therefore, $(\mathcal{P}' \cup \{Z\}[C]$ contains a Tucker submatrix.

We now show that C is minimal with respect to this property. If the overlap graph of $\mathcal{P}'[C \setminus \{c\}]$ is not connected, then $(\mathcal{P}' \cup \{Z\})[C \setminus \{c\}]$ does not satisfy the conditions of Lemma 15. Thus, all columns of $C \cap C_Q \setminus \{c_{p(1)}, c_{p(2)}, c_{p(3)}\}$ are necessary for $(\mathcal{P}' \cup \{Z\})[C]$ to satisfy the conditions of Lemma 15.

Suppose $(c_{p(1)}, c_{p(2)}, c_{p(3)})$ is a 1-0-1 configuration. Due to the absence of any element of the unconstrained class in C, after the first `if` statement, only condition 1 of Lemma 14 is satisfied. Therefore, each element $\{c_{p(1)}, c_{p(2)}, c_{p(3)}\}$ that is retained in C is necessary for $(\mathcal{P}' \cup \{Z\})[C]$ to satisfy the requirements of Lemma 15, either because its removal would undermine all 1-0-1 configurations or would undermine the connectivity of the overlap graph of $\mathcal{P}'$.

If $(c_{p(1)}, c_{p(2)}, c_{p(3)})$ is a 0-1-0 configuration, it is because condition 1 of Lemma 14 is not satisfied. Removal of c' from C would undermine condition 2 of Lemma 14, hence the conditions of Lemma 15. Each element of $\{c_{p(1)}, c_{p(2)}, c_{p(3)}\}$ that is retained in C is necessary for $(\mathcal{P}' \cup \{Z\})[C]$ to satisfy the requirements of Lemma 15, either because its removal would undermine all 0-1-0 configurations or would undermine the connectivity of the overlap graph of $\mathcal{P}'$.

By Lemma 15, the returned submatrix, $(\mathcal{P}' \cup \{Z\})[C]$, is a Tucker submatrix.

Algorithm 6: `FindColumns`(M)

```
Data: A matrix $M$ that does not have the consecutive-ones property and such
      that every Tucker submatrix in $M$ contains all rows of $M$, and such that,
      excluding the last row of $M$, the overlap graph of the rows is a path, $\mathcal{P}'$.
Result: A Tucker submatrix of $\mathcal{P}' \cup \{Z\}$.
Let $C$ be the columns of $M$;
$C_Q \longleftarrow \bigcup \mathcal{P}'$;
Let $(c_1, c_2, \cdots, c_k)$ be the left-to-right ordering of elements of $C_Q$ in a
    consecutive-ones ordering of $\mathcal{P}'$;
Compute the members of $\mathcal{A}$;
if there is a 1-0-1 configuration then
    // The first condition of Lemma 14 applies
    $(c_{p(1)}, c_{p(2)}, c_{p(3)}) \longleftarrow$ a 1-0-1 configuration for $Z$;
    $C \longleftarrow C_Q$;
else
    // The second condition of Lemma 14 applies
    $C \longleftarrow C_Q \cup \{c'\}$;
    $(c_{p(1)}, c_{p(2)}, c_{p(3)}) \longleftarrow$ a 0-1-0 configuration for $Z$;
for $i \in (1, 2, \ldots, k)$ do
    if $c_i \notin \{c_{p(1)}, c_{p(2)}, c_{p(3)}\}$ and $c_i$ is not the only element of $C$ in any
    member of $\mathcal{A}$ then
        $C \longleftarrow C \setminus \{c_i\}$;
if $(c_{p(1)}, c_{p(2)}, c_{p(3)})$ is a 1-0-1 configuration for $Z$ then
    for $i = 1$ to 3 do
        if $c_{p(i)}$ is not the only element of $C$ in any member of $\mathcal{A}$ and $C \setminus \{c_{p(i)}\}$
        has a 1-0-1 configuration for $Z$ then
            $C \longrightarrow C \setminus \{c_{p(i)}\}$;
else
    for $i = 1$ to 3 do
        if $c_{p(i)}$ is not the only element of $C$ in any member of $\mathcal{A}$ and $C \setminus \{c_{p(i)}\}$
        has a 0-1-0 configuration then
            $C \longrightarrow C \setminus \{c_{p(i)}\}$;
return $(\mathcal{P}' \cup \{Z\})[C]$;
```

Lemma 20. `FindColumns`(M) *can be implemented to take* $O(size(M))$ *time.*

Proof. A consecutive-ones ordering of $\mathcal{P}'$ takes $O(size(M))$ time by [1].

By Lemma 16, it takes $O(size(M))$ time to list the members of each instance of $R_i \setminus R_{i+1}$, $R_i \cap R_{i+1}$, and $R_{i+1} \setminus R_i$ for $i \in \{1, 2, \ldots, h-1\}$. This gives the members of $\mathcal{A}$, some of them possibly more than once. Let $\mathcal{L}$ be this collection of lists. Give each column a list of members of $\mathcal{L}$ to which it belongs. Initialize a *cardinality counter* on each list in $\mathcal{L}$, indicating the number of columns of C that it contains. These operations take $O(size(M))$ time using elementary methods, given that the sum of lengths of the lists in $\mathcal{L}$ is $O(size(M))$.

In the first `for` loop, each column c_i is tested to see whether it one of the lists of $\mathcal{L}$ that contains it has a cardinality counter of 1. If not, c_i is removed from C and the cardinality counters of the lists of $\mathcal{L}$ that contain it are decremented. The total time for this loop is bounded by summing, over all $c_i \in \{c_1, c_2, \ldots, c_k\}$, the number of members of $\mathcal{L}$ that contain c_i. This is just the sum of cardinalities of lists in $\mathcal{L}$, hence $O(size(M))$.

The procedure needs to test for or find a 1-0-1 or a 0-1-0 configuration on at most five occasions: two when it initially determines whether there is a 1-0-1 or 0-1-0 configuration, thereby finding $(c_{p(1)}, c_{p(2)}, c_{p(3)})$, and three in one of the last two `for` loops, when it tests whether removal of $c_{p(1)}$, $c_{p(2)}$, and $c_{p(3)}$ leave a 1-0-1 or 0-1-0 configuration. Each of these occasions requires $O(size(M))$ time by Lemma 18.

Example: *We give an illustration of how Algorithm 6 works on the example of Figure 9 (bottom).* $\mathcal{A}$ *consists of the following sets:* $\{3, 4, \ldots, 14\}$, $\{15\}$ *and* $\{16\}$, *which ensure the overlap relation of* F *and* G, $\{16\}$, $\{15\}$, $\{12, 13, 14\}$, *which ensure the overlap relation of* G *and* H, $\{13, 14, 15\}$, $\{12\}$ *and* $\{10, 11\}$, *which ensure the overlap relation of* H *and* J. *Some of these sets are redundant, but there is no need to detect this. We put a cardinality counter on each of these sets, and decrement it whenever a column in the set is deleted. At least one column from each of these sets must be retained to maintain the connectivity of the overlap graph of* $\mathcal{P}'$.

Initially, $C = \{0, 1, \ldots, 16\}$, *and* $\mathcal{C}_Q = \{3, 4, \ldots, 16\}$. *There is no 1-0-1 configuration, so the algorithm retains a column* c' *in the unconstrained class that contains a 1 in row* Z. *Suppose it selects* $c' = 2$. *It eliminates the remaining columns* 0 *and* 1 *of the unconstrained class from* C. *It finds the 0-1-0 configuration* $(c_{p(1)}, c_{p(2)}, c_{p(3)}) = (4, 10, 12)$.

The first `for` *loop eliminates all but columns* $\{2, 4, 10, 12, 15, 16\}$*; 2 is skipped because it is a member of the unconstrained Venn class, 4, 10, and 12 are skipped because they are* $c_{p(1)}, c_{p(2)}$ *and* $c_{p(3)}$*, and 15 and 16 are retained because they are the only remaining elements of some member of* $\mathcal{A}$ *when they are reached. The final* `for` *loop determines that elimination of 4 or 10 would undermine all 0-1-0 configurations, and elimination of 12 would remove the last remaining element of the member* $\{12\}$ *of* $\mathcal{A}$.

The resulting Tucker submatrix is that depicted on the righthand side of Figure 9.

That the algorithm is incomplete without the last two `for` loops is illustrated by the following example. Let $R_1 = \{c_1, c_2\}$, $R_2 = \{c_2, c_3, c_4\}$, $R_3 = \{c_4, c_5\}$, $Z = \{c_1, c_2, c_5\}$, and $(c_1, c_2, c_3, c_4, c_5)$ be a consecutive-ones ordering of $\mathcal{R}_Q = \{R_1, R_2, R_3\}$. Let $(c_{p(1)}, c_{p(2)}, c_{p(3)}) = (c_1, c_3, c_5)$. None of the columns is removed by the first `for` loop, since each of c_2 and c_4 is the sole element of a member of $\mathcal{A}$. However, $c_{p(2)}$ can be eliminated without undermining Lemma 15. Although $c_{p(1)}$, $c_{p(2)}$, and $c_{p(3)}$ are in separate Venn classes, there is no member of $\mathcal{A}$ that contains $c_{p(2)}$ and excludes both $c_{p(1)}$ and $c_{p(3)}$. The second `for` loop is required to eliminate it. The third `for` loop handles analogous situations when $(c_{p(1)}, c_{p(2)}, c_{p(3)})$ is a 0-1-0 configuration.

Theorem 1. *It takes $O(size(M))$ time to find a Tucker submatrix in any matrix that does not have the consecutive-ones property.*

Proof. The calls to `TuckerRows`, `FindRows` and `FindColumns` take $O(size(M))$ time by Lemmas 7, 13 and 20. This gives the result if `TuckerRows` returns a submatrix with greater than four rows. If `TuckerRows` returns a submatrix with $i \leq 4$ rows, then we cannot use `FindColumns` to find the columns of a Tucker submatrix. However, by Lemma 6, every Tucker submatrix contains all i rows. One way to find one is to generate all $i! \leq 24 = O(1)$ orderings of rows, and for each, to check for the existence of the column vectors of length i, as depicted in Figure 2. One of the Tucker submatrices has been found if all of its depicted column vectors are found in one of the orderings. This takes $O(m) = O(size(M))$ time.

5 Finding a Lekkerkerker-Boland Subgraph

Henceforth, we let n denote the number of vertices and m the number of edges in a graph. In [7], an algorithm is given that finds an AT in a non-interval chordal graph in $O(n + m)$ time. However, it does not follow from this result that there is a linear-time algorithm for finding an LB subgraph. One reason is that the algorithm of that paper can produce AT's that are not the AT of any induced LB subgraph. Some of the difficulties posed by this and other pitfalls are explained below.

Let M be the clique matrix of a graph G. For notational convenience, in this section, we will treat the rows of M as interchangeable with the corresponding vertices of G, rather than as sets of columns, as we have denoted them up to this point. This allows us to refer to the subgraph of G induced by a set X of rows of M, for example. Since G is the intersection graph of the rows of its clique matrix, $G[X]$ is the intersection graph of the rows in X. We will also treat the columns interchangeably with the cliques they represent. This allows us to refer to the intersection of two columns, for example.

Given that the clique matrices of chordal graphs have a consecutive-ones ordering if and only if they are interval graphs, it follows that the clique matrix of every non-interval chordal graph must contain a Tucker submatrix. As observed above, the Tucker submatrices other than $M_I(k)$ for $k > 3$ are obtained by deleting the simplicial (square) vertices in Figure 3 from the clique matrices of chordal LB graphs. Each of these vertices is simplicial, so each is a member of single column of the clique matrix. Let the *incomplete columns* of the Tucker submatrices be the three columns from which these simplicial vertices are removed. Other than for $M_{III}(k)$ for $k > 3$, this uniquely defines the three incomplete columns for every Tucker submatrix. By *completing a Tucker matrix*, let us denote the inverse operation, namely, for each incomplete column, adding a row that contains a 1 in that column and in no other column of the Tucker submatrix.

One could mistakenly believe that, since a Tucker submatrix in the clique matrix of G proves that G is not an interval graph, and an induced LB subgraph proves the same thing, all that is required to complete an induced LB subgraph of G is to find the rows that complete the submatrix.

The first pitfall is that proof is required that a Tucker submatrix can always be completed using rows of the clique matrix of G. It is easy to believe that this is self-evident and that no proof of it is required. However, this is not true when the graph is not chordal; where the clique matrix has a Tucker submatrix that cannot be extended in this way [9]. The example gives no obvious insight into why a similar phenomenon cannot happen when the graph is chordal.

Fortunately, a Tucker submatrix of the clique matrix of G can always be completed if G is chordal, as we show below. However, it still does not follow that completing the Tucker matrix in this way yields an LB subgraph. The problem is that that the intersection graph of rows of the Tucker submatrix may not faithfully represent the subgraph of G that these rows induce.

Figure 12 illustrates this pitfall. Suppose the depicted instance of M_{IV} is found to be a submatrix of the clique matrix M of G. Without loss of generality, suppose that the depicted ordering of M_{IV} is consistent with the ordering of the clique matrix, M, and that the 1's in each of rows 0, 1 and 2 are consecutive in M. If the intersection graph of these rows is the depicted instance M_{IV}, then the completion of this Tucker matrix yields the first graph to its right, which is G_I. However, it may be that rows 0 and 1 or that rows 1 and 2 intersect in one of the columns that do not form part of the submatrix. In G, the pairs $\{0, 1\}$ and $\{1, 2\}$ might be adjacent.

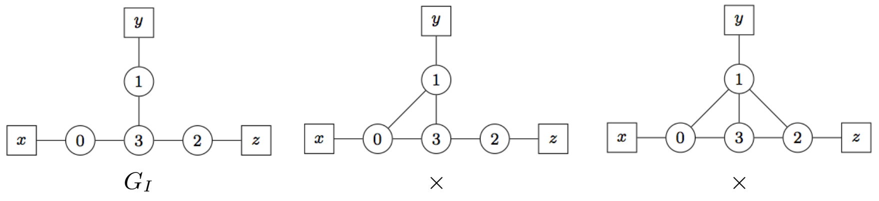
If 0 and 1 are adjacent but 1 and 2 are not, then completing the M_{IV} yields the clique matrix of G_I. However, the subgraph induced by rows in G is second graph on the right. This is not an LB subgraph. When z is deleted from it, however, it yields an LB subgraph, $G_{IV}(6)$. The case where 0 and 1 are nonadjacent and 1 and 2 are adjacent is symmetric with this case.

If 0 and 1 are adjacent and 1 and 2 are adjacent, the completion still gives the clique matrix of G_I, but the subgraph of G induced by its rows is the third graph on the right, which is an instance of $G_{IV}(7)$.

Similarly, if an instance of M_V is found, its completion yields a submatrix of the clique matrix of G that is the clique matrix of G_{II}. However, the subgraph of G induced by the rows of the instance could be the second graph on the

$$M_{IV}$$

$$\begin{array}{c|cccccc}
0 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 0 \\
2 & 0 & 0 & 0 & 0 & 1 & 1 \\
3 & 0 & 1 & 0 & 1 & 0 & 1 \\
x & 1 & 0 & 0 & 0 & 0 & 0 \\
y & 0 & 0 & 1 & 0 & 0 & 0 \\
z & 0 & 0 & 0 & 0 & 1 & 0
\end{array}$$



$$M_V$$

$$\begin{array}{c|ccccc}
0 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 \\
2 & 1 & 1 & 1 & 1 & 0 \\
3 & 1 & 0 & 0 & 1 & 1 \\
x & 0 & 1 & 0 & 0 & 0 \\
y & 0 & 0 & 1 & 0 & 0 \\
z & 0 & 0 & 0 & 0 & 1
\end{array}$$

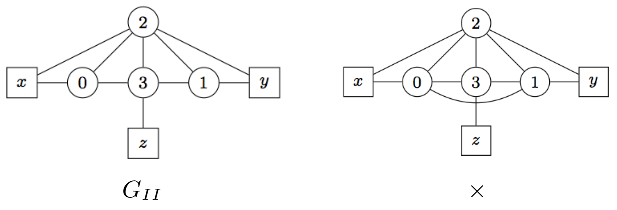


Fig. 12: Even when a Tucker submatrix of a clique matrix can be completed using rows of the clique matrix M of a graph G, this completion does not necessarily give the vertices of an LB subgraph. The intersection graph of rows of the completed submatrix may not accurately reflect the subgraph of G that they induce. Rows that do not intersect in the submatrix may intersect in M. The graphs below each matrix are the possible subgraphs induced by the rows in the completion of the Tucker submatrices M_{IV} and M_V. The middle graph below M_{IV} is not an LB subgraph, though it contains one, a $G_{IV}(6)$ induced by $\{x, y, 0, 1, 2, 3\}$. The second graph below M_V is not an LB subgraph, though it contains a $G_{IV}(6)$ induced by $\{x, y, z, 0, 1, 3\}$. Similar issues arise in the completion of Tucker submatrices that are examples of $M_{II}(k)$ and $M_{III}(k)$.

right, which is not an LB graph, though it contains an induced LB subgraph, an instance of G_{IV}.

Similar issues arise with the completion of instances of $M_I(k)$, $M_{II}(k)$ and $M_{III}(k)$.

To get around these problems, we make use of a stronger version of Theorem 1:

Lemma 21. *When M does not have the consecutive-ones property, it takes $O(size(M))$ time to find a Tucker submatrix whose rows are a minimal set of rows of M that contain a Tucker submatrix of M.*

Proof. If `TuckerSubmatrix` (Algorithm 2) returns a submatrix with at most four rows, this follows from Lemma 6. Otherwise, it follows from Lemma 8.

The key result of this section is the following:

Lemma 22. *Suppose G is chordal, M_T is a Tucker submatrix in the clique matrix M of G, and that the rows of M_T are a minimal set of rows that contain a Tucker submatrix of M. Then M_T is not an instance of $M_I(k)$ for $k \geq 4$, hence it has three incomplete columns. For every choice of rows x, y, z that complete the three incomplete columns of M_T, x, y, z and the rows of M_T induce an LB subgraph of G.*

We prove the lemma below. This gives the strategy for finding an LB subgraph in any graph that is not an interval graph, which is summarized as Algorithm 7.

Algorithm 7: `FindLBSubgraph`(G)

```
Data: $G$ is not an interval graph.
Result: The vertices of $G$ inducing an LB subgraph.
1. Test whether $G$ is chordal using the algorithm of [15].
2. If it is not chordal, return an instance of $G_{III}(k)$ for $k \geq 4$, using the algorithm
   of [19].
3. Otherwise, let $M$ be the clique matrix of $G$ produced by the algorithm of [15].
4. Using a call to TuckerSubmatrix($M$), find a Tucker submatrix $M_T$ in $M$. Let $X$
   be its rows in $M$.
5. Find three rows $\{x, y, z\}$ of $M$ that complete $M_T$.
6. Return $X \cup \{x, y, z\}$.
```

It remains to prove Lemma 22. We begin by showing that the three incomplete columns in any Tucker submatrix in the clique matrix of a chordal graph can always be completed with three additional rows.

A *clique tree* of a chordal graph is a tree $\mathcal{T}$ that has one node for each clique, and with the property that, for each vertex v of G, the cliques that contain

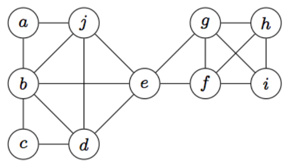
v induce a connected subtree. Every chordal graph has a clique tree, see for example [6]. There is not necessarily a unique clique tree.

Observe that if $\mathcal{K}$ is the set of cliques that contain some vertex v and a clique C does not contain v, C cannot lie on the path between any pair of members of $\mathcal{K}$ in any clique tree. Otherwise, the subtree induced by cliques containing it would not be connected, contradicting the definition of a clique tree.

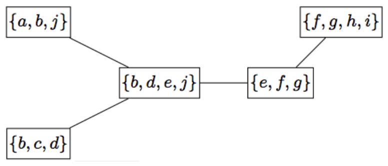
Generalizing from this insight, we obtain the following:

Lemma 23. *Let G be a connected chordal graph and let $\mathcal{T}$ be a clique tree for G. Let $\mathcal{K}$ be a set of cliques of G and let C be a clique such that $C \notin \mathcal{K}$. Let $\mathcal{K}'$ be the multiset obtained by removing the members of C from every clique in $\mathcal{K}$. If the intersection graph of $\mathcal{K}'$ is connected, then C does lie on the path in $\mathcal{T}$ between any two members of $\mathcal{K}$.*

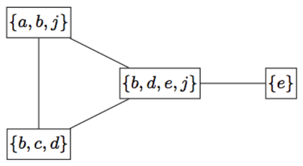
Proof. Suppose to the contrary that C lies on the path between two members of $\mathcal{K}$ in $\mathcal{T}$. Removal of C from $\mathcal{T}$ leaves a set of two or more trees that partition $\mathcal{K}$. Since the intersection graph of $\mathcal{K}'$ is connected, there exist two of these trees, one with $K_1 \in \mathcal{K}'$ and the other with $K_2 \in \mathcal{K}'$ such that K_1 intersects K_2 on a vertex $v \notin C$. Since C lies on the path from K_1 to K_2 in $\mathcal{T}$, the subtree of the clique tree induced by cliques containing v is not connected, contradicting the definition of a clique tree.



A chordal graph G



G's clique tree



Intersection graph of $\mathcal{K} \setminus C$

Fig. 13: Let C be the clique $\{f, g, h, i\}$ and let $\mathcal{K}$ be the remaining cliques of G. To the lower right is the intersection graph of the members of $\mathcal{K}$ after elements of C have been removed from them. This graph is connected, which means by Lemma 23 that $\{f, g, h, i\}$ cannot lie on the path between any pair of members of $\mathcal{K}$ in any clique tree of G.

Figure 13 illustrates the idea. The following lemma is immediate from results that appear in [6].

Lemma 24. *Let $\mathcal{T}$ be a clique tree for a chordal graph G and let K be a leaf. Then K contains a simplicial vertex of G. Let S be the simplicial vertices of K and let $\mathcal{T}'$ be the result of deleting leaf K from $\mathcal{T}$. Deleting S from G yields an induced subgraph that has $\mathcal{T}'$ as a clique tree. (See Figure 14.)*

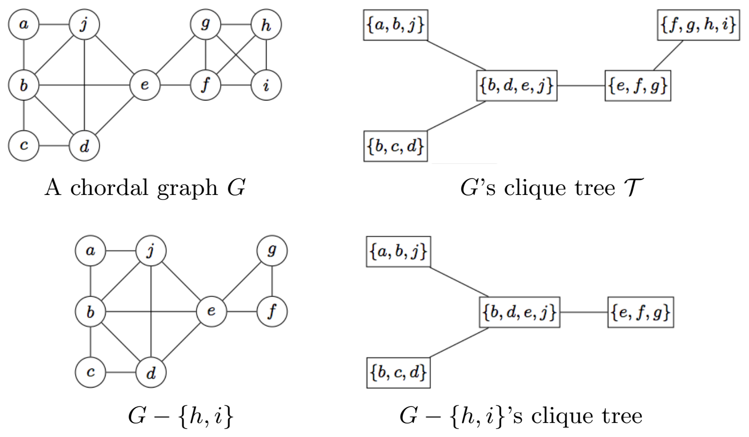


Fig. 14: “Shrinking” a clique tree. If G is a chordal graph and $\mathcal{T}$ is a clique tree for it, then deleting the simplicial vertices $\{h,i\}$ in a leaf $\{f,g,h,i\}$ of $\mathcal{T}$ gives a smaller graph that has as a clique tree the tree obtained by deleting K from $\mathcal{T}$.

Definition 6. *By* shrinking a clique tree $\mathcal{T}$, *let us denote the operation of deleting the set S of simplicial vertices in a leaf K of $\mathcal{T}$, yielding a smaller graph G' that has $\mathcal{T}-K$ as a clique tree.*

Lemma 25. *Let G be a chordal graph and let M_T be a Tucker submatrix of a clique matrix M of G. Then M_T can be completed using rows of M.*

Proof. Let C be the column of M that contains an incomplete column of a Tucker submatrix M_T, and let $\mathcal{K}$ be the columns of M that contain the the remaining columns of M_T. Iteratively shrink the clique tree of G subject to the constraint that we do not delete C or any member of $\mathcal{K}$. Let $\mathcal{T}'$ be the resulting clique tree when no more cliques can be deleted without violating the constraint, and let G' be the corresponding induced subgraph of G.

All leaves of $\mathcal{T}'$ are members of $\mathcal{K}$; otherwise they could be deleted through further shrinking without violating the constraint. Therefore, any member of $\mathcal{K}$ that is not a leaf lies on the path in $\mathcal{T}'$ between two other members of $\mathcal{K}$.

By inspection on each of M_I, M_{II}, $M_{III}(3)$, M_{IV} and M_V, each choice of C satisfies the requirements of Lemma 23 that prevent it from lying on the path between any two members of $\mathcal{K}$. Therefore, each incomplete column of M_T is a leaf in $\mathcal{T}'$, hence contains a simplicial vertex v in G'. The row corresponding to

v has a single 1 in the clique matrix of G'. Therefore, the addition of v to M_T completes the column.

Lemma 26. *If three vertices complete three columns of a Tucker submatrix of the clique matrix of a chordal graph G, then they are pairwise nonadjacent.*

Proof. Let X be the vertices of the Tucker submatrix. For each pair a, b of the vertices that complete it, the adjacencies of a and b to members of X are known, by definition. Also, a has a neighbor $a' \in X \setminus N(b)$ and b has a neighbor $b' \in X \setminus N(a)$, and there is a path P in X from a' to b' that avoids the common neighbors of a and b. Even if there are unknown chords on P, a, a', b', b and zero or more members of P form a chordless cycle, contradicting the chordality of G.

Proof of Lemma 22. The cases for M_{IV} and M_V are illustrated in Figure 12. By assumption in each case, $\{0, 1, 2, 3\}$ is a minimal set of rows of M that contain a Tucker matrix. Also, assume without loss of generality that the depicted the ordering of columns is consistent with a consecutive-ones ordering of rows $\{0, 1, 2\}$ of M. By Lemma 26, the only possible adjacencies among $\{0, 1, 2, 3\}$ that are not reflected by the intersection graph of M_T are $\{0, 1\}$ and $\{1, 2\}$. If only 0 and 1 are adjacent, $\{0, 1, 3, x, y, z\}$ induces an instance of $G_{IV}(6)$. Since the clique matrix of $G_{IV}(6)$ consists of three rows for its simplicial vertices, plus a Tucker matrix, rows $\{0, 1, 3\}$ induce a Tucker matrix in the clique matrix of $G_{IV}(6)$. Thus, it is a Tucker matrix in the clique matrix of G, contradicting the minimality of $\{0, 1, 2, 3\}$. This case cannot occur. By symmetry, it cannot be the case that only 1 and 2 are adjacent. Therefore, x, y and z, together with the rows of the M_V, induce an instance of G_I or of G_{IV}.

Similarly, for M_V, by Lemma 26, the case where 0 and 1 are adjacent yields an $M_{II}(3)$ on $\{0, 1, 3\}$, contradicting the minimality of $\{0, 1, 2, 3\}$, hence, the completion with x, y and z induces a G_{II}.

For $M_T \in \{M_I(k), M_{II}(k), M_{III}(k)\}$, the rows of M corresponding to $\{0,$ $1,$ $2, \ldots, k-1\}$ are a minimal set $\mathcal{R}$ of rows of M that contain a Tucker submatrix, by assumption. Let X be the corresponding vertices of G. Assume that the rows of $\mathcal{R}$ are ordered as shown in Figure 2, so that only the last row of $\mathcal{R}$ is not consecutive-ones ordered.

Every pair of nonadjacent vertices in the intersection graph of M_T contains some $i \in \{1, 2, \ldots, k-4\}$ if M_T is an instance of $M_{II}(k+1)$, or $i \in \{0, 1, \ldots, k-4\}$ if M_T is an instance of $M_{III}(k)$ for $k \geq 4$. Without loss of generality, we may assume that i is the lower-numbered member of the adjacent pair. In each case, row $i+1$ contains the right endpoint of row i in M. Therefore, $i+2$ is the only possible higher-numbered neighbor of i in G that is not a neighbor in the intersection graph of M_T. Suppose $i+2$ is a neighbor of i. Then M has a column that contains both i and $i+2$. In M_T, we may replace the column that contains i and $i+1$ and the column that contains $i+1$ and $i+2$ with this column, and then delete row $i+1$ from M_T, yielding a smaller instance of M_{II} or of M_{III}. This contradicts the assumed minimality of the rows of M_T. It follows that the intersection graph of rows of M_T faithfully represents $G[X]$. The completion of M_T with three vertices $\{x, y, z\}$ induces an induced LB subgraph by Lemma 26.

An identical argument applies to $i \in \{0, 1, \ldots, k-4\}$ for $M_I(k)$ and $k \geq 4$. By the cyclic symmetry of M_I, it therefore applies for all rows of M_I, and the intersection graph of M_I faithfully represents $G[X]$. Since this is a chordless cycle, no instance of $M_I(k)$ for $k \geq 4$ satisfies the conditions of Lemma 21. The intersection graph of $M_I(3)$ is complete, so it faithfully represents $G[X]$, and its completion is an induced LB subgraph by Lemma 26.

Summarizing the results of this section, we obtain the following:

Theorem 2. *Given an adjacency-list representation of an arbitrary graph G, it takes $O(n+m)$ time to find either an interval model of G or an induced LB subgraph.*

Proof. If G is an interval graph, an interval model can be produced in linear time by [1]. Otherwise, in a call to `FindLBSubgraph`(G), the cited algorithms run in $O(n+m)$ time. When G is not chordal, it returns an instance of $G_{III}(k)$ for $k \geq 4$, which is an LB subgraph. When G is chordal, the clique matrix M returned by the algorithm of [15] has $size(M) = O(n+m)$. Since G is not an interval graph, M does not have the consecutive-ones property, so it contains a Tucker submatrix. By Lemma 21, it takes in $O(size(M)) = O(n+m)$ time to find one that satisfies the conditions of Lemma 22. Denote it by M_T.

Let X be the vertices corresponding to rows of M_T. For each incomplete column of M_T, make a list of the rows that have a 1 in the column. For each vertex of G other than those in X, check whether $N(v) \cap X$ is the set of rows in one of these lists. This takes $O(1 + |N(v)|)$ by marking and counting neighbors of v in X, checking each list for an unmarked vertex, then unmarking them, for a total of $O(n+m)$ over all vertices of G. This gives the set of rows that complete incomplete columns of M_T, since a row that has a 1 in any other column of M_T has neighbors in X that are not in any incomplete column. Selecting one row from each of these sets gives an LB subgraph by Lemma 22.

References

[1] G. S. Booth and K. S. Lueker, *Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms*, J. Comput. Syst. Sci., 13 (1976), pp. 335–379.

[2] A. Brandstädt, V.B. Le, and J.P. Spinrad, *Graph Classes: A Survey*, SIAM Monographs on Discrete Mathematics, Philadelphia, 1999.

[3] C. Chauve, U.-U. Haus, T. Stephen, and V. P. You, *Minimal conflicting sets for the consecutive ones property in ancestral genome reconstruction*, Journal of Computational Biology, 17 (2010), pp. 1167–1181.

[4] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, Cambridge, Massachusetts, 2009.

[5] M. Dom, J. Guo, and R. Niedermeier, *Approximation and fixed-parameter algorithms for consecutive ones submatrix problems*, Journal of Computer and System Sciences, 76 (2010), pp. 204–221.

[6] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.

[7] D. Kratsch, R.M. McConnell, K. Mehlhorn, and J.P. Spinrad, *Certifying algorithms for recognizing interval graphs and permutation graphs*, SIAM Journal on Computing, 36 (2006), pp. 326–353.

[8] C. Lekkerkerker and D. Boland, *Representation of finite graphs by a set of intervals on the real line*, Fund. Math., 51 (1962), pp. 45–64.

[9] N. Lindzey and R. M. McConnell, *Linear-time algorithms for finding Tucker matrices and Lekkerkerker-Boland subgraphs*, Tech. Report arXiv:1401.0224v1 [cs.DM], http://arxiv.org/abs/1401.0224, 2013.

[10] ———, *On finding Tucker submatrices and Lekkerkerker-Boland subgraphs*, Proceedings of the Thirty-ninth International Workshop on Graph-Theoretic Concepts in Computer Science, June 19, 2013, Lubeck, Germany, Lecture Notes in Computer Science, 8165 (2013), pp. 345–357.

[11] R. M. McConnell, *A certifying algorithm for the consecutive-ones property*, Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA04), 15 (2004), pp. 761–770.

[12] R. M. McConnell, K. Mehlhorn, S. Näher, and P. Schweitzer, *Certifying algorithms*, Computer Science Reviews, 5 (2011), pp. 119–161.

[13] J. Meidanis, O. Porto, and G.P. Telles, *On the consecutive ones property*, Discrete Applied Mathematics, 88 (1998), pp. 325–354.

[14] Fred S. Roberts, *Graph Theory and Its Applications to Problems of Society*, Society for Industrial and Applied Mathematics, Philadelphia, 1978.

[15] D. Rose, R. E. Tarjan, and G. S. Lueker, *Algorithmic aspects of vertex elimination on graphs*, SIAM J. Comput., 5 (1976), pp. 266–283.

[16] J. Spinrad, *Efficient Graph Representations*, American Mathematical Society, Providence RI, 2003.

[17] J Stoye and R. Wittler, *A unified approach for reconstructing ancience gene clusters*, IEEE/ACM Transactions on Computational Biology and Bioinformatics, 6 (2009), pp. 387–400.

[18] M. Tamayo, *Algorithms for finding tucker patterns*, master's thesis, Simon Fraser University, 2013.

[19] R. E. Tarjan and M. Yannakakis, *Addendum: Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs*, SIAM Journal on Computing, 14 (1985), pp. 254–255.

[20] A. Tucker, *A structure theorem for the consecutive 1's property*, Journal of Combinatorial Theory, Series B, 12 (1972), pp. 153–162.