# A Theory for the Consecutive Ones Property

João Meidanis[1 3] Erasmo G. Munuera[2]

[1] Institute of Computing, University of Campinas, Brazil.
[2] National Institute for Space Research, Brazil.
[3] Supported by grants from FAPESP and CNPq.

**Abstract.** We propose a new, set-theoretical approach to the study of the consecutive ones property for binary matrices. A matrix $M$ has this property if there is a permutation of its rows that leaves the ones consecutive in each column. In some applications, it is important to know all such permutations. Booth and Lueker devised in 1975 a data structure called PQ-tree to efficienlty store this information. We generalize this structure into what we call the PQR-tree, which exists for all binary matrices. The new structure gives more information on the problem by pointing out the obstructions to the property, namely, the R-nodes. We outline a quadratic time algorithm to construct the PQR-tree given a matrix in list form.

## 1 Introduction

The consecutive ones property appears in many applications, ranging from data retrieval to literature and archeology [5, 7]. In particular, it appears in several problems in molecular biology, including DNA physical mapping, sequence assembly, and other problems involving interval graphs. Testing for the consecutive ones property is part of the standard procedure for recognizing interval graphs and proper interval graphs, although other methods exist for this task [6, 4, 3].

The consecutive ones property is a property of binary matrices, that is, two-dimensional matrices with entries 0 and 1. We say that a binary matrix $M$ has the **consecutive ones property** (which will be abbreviated to C1P from now on) when the rows of $M$ can be permuted so that, in each column of $M$, the 1's appear consecutively. Figure 1(a) shows a matrix $M$ with the C1P, and Figure 1(b) shows a permutation of the rows of $M$ that leaves consecutive ones in all columns. Such permutations will be called **valid** permutations.

$$\begin{array}{c} a \\ b \\ c \\ d \end{array} \begin{pmatrix} 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \qquad\qquad \begin{array}{c} b \\ a \\ c \\ d \end{array} \begin{pmatrix} 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

(a) (b)

**Fig. 1.** (a) Binary matrix $M$ with the C1P; (b) permutation of the rows of $M$ so that all columns have consecutive ones.

Not all matrices have the C1P. Figure 2 shows a binary matrix $M$ that fails to comply to the C1P conditions. There are no valid permutations of the rows of $M$.

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

**Fig. 2.** Binary matrix without the C1P.

In 1975, Booth and Lueker devised an elegant data structure that is able to compactly represent **all** valid permutations of the rows of $M$, provided $M$ has the C1P [2]. This structure is highly valuable in applications in which it is important to record all solutions compatible with current data, as in DNA physical mapping. Further experiments can then refine the structure, if needed.

These structures are called **PQ-trees**. They are rooted trees of unbounded arity in which the leaves are identified with the rows of $M$, and each internal node is of type P or type Q. A node can have an arbitrary number of children, but to avoid ambiguity we require that P-nodes have at least two children and Q-nodes have at least three. In addition, the children of a node form an ordered set, that is, changing this order gives a distinct tree. Sometimes this reordering of children yields what is called an **equivalent** tree. Type P nodes allow arbitrary reordering of their children to yield equivalent trees. Type Q nodes are more rigid, and a reversal of their children is the only allowed operation to keep the tree equivalent to the original one. Any number of such operations can be applied on several nodes, still yielding equivalent trees.

Booth and Lueker showed that for each matrix $M$ with the C1P there is a PQ-tree such that its equivalence class records all valid permutations of the rows of $M$. In addition, they described a linear time algorithm to construct such a tree given $M$. However, only matrices with the C1P admit PQ-trees. The time bound of this algorithm is linear in three parameters: $n =$ the number of rows of $M$, $m =$ the number of columns of $M$, and $r =$ the total number of ones in $M$. Thus, if $M$ is sparse, the algorithm runs in subquadratic time. Throughtout this paper whenever we say "linear" we mean "$O(n + m + r)$."

This algorithm is $O(n + m + r)$ but its implementation is cumbersome. Recently, Hsu proposed a novel approach, based on the concept of *module* of a graph, and described another linear time algorithm for the problem. However, it is not clear how to construct the PQ-tree or equivalent structure that encodes all valid permutations with this approach.

Other work has focused on the related problem of recognizing interval graphs [8, 6]. Although there is a standard way of reducing the recognition of interval graphs to testing the C1P, no one knows how to do the opposite reduction in linear time. A parallel algorithm for testing the C1P has appeared in the literature as well [1].

Our goal is to design a simple, fast algorithm that tests for the C1P and constructs an appropriate PQ-tree when the property holds. This paper is a step toward this goal. Here we:

- propose a new, elegant theory underlying the problem
- create the notion of a PQR-tree, a generalization of PQ-trees, so that the new structure applies to all matrices
- characterize the node sets of such a structure, and present an algorithm to construct the PQR-tree of a binary matrix.

Our algorithm is quadratic in $s = n + m + r$. We believe this time bound can be reduced to linear with a more elaborate algorithm, which will still be much simpler than the original PQ-tree algorithm of Booth and Lueker. We are currently pursuing research in this direction.

Other generalizations of PQ-trees have appeared in the literature. Korte and Möring [8] define a modified form of PQ-tree to deal specifically with the problem of recognizing interval graphs. Novick [10] proposed a generalization which solves a related problem, that of finding all orthogonal sets (in the language of Section 3).

The rest of this paper is organized as follows. Section 2 sets the notation and basic definitions. Section 3 introduces the novel theory and presents an important decomposition theorem. Section 4 characterizes the indecomposable instances, which give rise to P-, Q-, and R-nodes, and presents the algorithm. Finally, Section 5 contains our concluding remarks. Most proofs are omitted in this extended abstract, due to space limitations.

## 2 Definitions and notation

For any set $U$, $\mathcal{P}(U)$ denotes the set of all subsets of $U$. We deal exclusively with finite sets. A **permutation** of a finite set $U$ of size $n$ is a bijection $\alpha : \{1, 2, \ldots, n\} \mapsto U$. A permutation will be indicated by $\alpha(1)\alpha(2)\ldots\alpha(n)$.

For a set $U$, define the **trivial** subsets of $U$ as being the empty set, the singletons, and $U$ itself. So, $\mathcal{T}(U) = \{\emptyset, U\} \cup \{\{a\} | a \in U\}$ is the collection of the trivial subsets of $U$.

Given a finite set $U$, a permutation $\alpha$ of the elements of $U$, and a subset $A$ of $U$, we say that $A$ is **consecutive** in $\alpha$ when the elements of $A$ appear consecutively in $\alpha$. For instance, let $U = \{a, b, c, d, e, f\}$ and $\alpha = becfda$; then $A = \{b, c, e\}$ is consecutive in $\alpha$ while $B = \{e, d\}$ is not.

Given a pair $(U, \mathcal{C})$ with $\mathcal{C} \subseteq \mathcal{P}(U)$ we say that a permutation $\alpha$ of $U$ is **valid** (with respect to $\mathcal{C}$) if all sets $A \in \mathcal{C}$ are consecutive in $\alpha$.

We will denote by *valid*$(U, \mathcal{C})$ or sometimes just by *valid*$(\mathcal{C})$ the set of all valid permutations with respect to $(U, \mathcal{C})$.

Similarly, *consec*$(\alpha)$ is the collection of all sets $A \subseteq U$ that are consecutive in $\alpha$. The term **collection** is just a synonym of set, but we will reserve it for sets of sets.

The informal definition of PQ-trees given in the introduction will be used throughout the paper. In addition, we will be speaking of PQR-trees, which are just like PQ-trees but with an additional type of node, the R-node. For equivalence purposes, the children of an R-node can be arbitrarily permuted, just like the children of a P-node.

## 3 A new theory

Let $M$ be a binary matrix, and let $U$ be the set of its rows. Each column $j$ of $M$ can be seen as the set $A \subseteq U$ formed by the rows $i$ such that $M[i,j] = 1$. The matrix $M$ can be seen then as a pair $(U, \mathcal{C})$ where $\mathcal{C}$ is the collection of the subsets of $U$ that correspond to its columns.

Our goal is to find all valid permutations in this context. The pair $(U, \mathcal{C})$ has the **consecutive ones property** (C1P) if there is at least one valid permutation.

A crucial observation is that if the sets of $\mathcal{C}$ are to be consecutive, many other sets must be consecutive as well. These include the following.

1. the intersection $A \cap B$ of two sets in $\mathcal{C}$
2. the union $A \cup B$ of two sets in $\mathcal{C}$, provided that $A \cap B \neq \emptyset$
3. the relative complement $A \setminus B$ of two sets in $\mathcal{C}$, provided that $B \not\subseteq A$

In addition, note that the singletons $\{a\}$ for $a \in U$ and $U$ itself are always consecutive in any permutation. We also make the convention that the **empty set is always consecutive**. Hence, the trivial subsets are always consecutive.

**Definition 1.** A collection $\mathcal{C}$ of subsets of $U$ is called **complete** if it contains all the trivial subsets of $U$ and is closed under operations (1), (2), and (3) above.

**Definition 2.** Given a collection $\mathcal{C} \subseteq \mathcal{P}(U)$, define $\overline{\mathcal{C}}$ as being the smallest complete supercollection of $\mathcal{C}$.

The collection $\overline{\mathcal{C}}$ is well defined since it is the intersection of all complete supercollections of $\mathcal{C}$:

$$\overline{\mathcal{C}} = \bigcap_{\text{complete } \mathcal{D} \supseteq \mathcal{C}} \mathcal{D}.$$

The following result shows that we might as well consider $\overline{\mathcal{C}}$ for the sake of obtaining all valid permutations.

**Theorem 3.** *For any collection* $\mathcal{C}$

$$\mathit{valid}(\mathcal{C}) = \mathit{valid}(\overline{\mathcal{C}}).$$

The proof is simple and will be omitted. More interesting for our purposes is the fact that any set $A$ that is consecutive in all permutations of $valid(\mathcal{C})$ is in $\overline{\mathcal{C}}$. While we do not prove this fact here, it is a consequence of what follows.

We are interested in certain special subsets of $U$ that allow us to decompose the problem into two smaller ones. These subsets are in $\overline{\mathcal{C}}$, but not all $\overline{\mathcal{C}}$ sets are suitable for our purpose. The notion of *orthogonal* set, defined below, gives the additional condition for an element of $\mathcal{C}$ to be what we need.

**Definition 4.** Given two subsets $A$ and $B$ of $U$, we say that $A$ and $B$ are mutually **orthogonal**, denoted by $A \perp B$, when either:

- $A \subseteq B$, or
- $B \subseteq A$, or else
- $A \cap B = \emptyset$.

**Definition 5.** Given a subset $A$ of $U$ and a collection $\mathcal{C} \subseteq \mathcal{P}(U)$, we write $A \perp \mathcal{C}$ when $A \perp B$ for every $B \in \mathcal{C}$. Such a set $A$ is said to be **orthogonal** to $\mathcal{C}$. The collection of all sets orthogonal to $\mathcal{C}$ is denoted by $\mathcal{C}^{\perp}$.

An aesthetically pleasing result whose proof is straightforward from the definitions, but perhaps a bit tedious, is given below.

**Theorem 6.** *For any collection* $\mathcal{C}$*,*

$$\mathcal{C}^{\perp} = \overline{\mathcal{C}^{\perp}} = (\overline{\mathcal{C}})^{\perp}.$$

We now proceed to describe a nice way of breaking the problem into two smaller ones, using sets in $\overline{\mathcal{C}} \cap \mathcal{C}^{\perp}$.

Given a pair $(U, \mathcal{C})$ with $\mathcal{C} \subseteq \mathcal{P}(U)$ and a subset $H$ of $U$, we can define a subinstance of the problem by taking the pair $(H, \mathcal{C} \cap \mathcal{P}(H))$.

Similarly, if $H$ is orthogonal to $\mathcal{C}$, we can construct another subinstance $(U/H, \mathcal{C}/H)$, where $U/H$ is the set $(U \setminus H) \cup \{H\}$, that is, the set obtained from $U$ by replacing all the elements of $H$ by a single element, which will be called $H$ for simplicity. The collection $\mathcal{C}/H$ is defined accordingly:

$$\mathcal{C}/H = \{A/H | A \in \mathcal{C}, A \not\subset H\},$$

where we make the convention that $A/H = A$ if $A \cap H = \emptyset$. Recall that $H$ is in $\mathcal{C}^{\perp}$. Hence, there are only three types of sets $A$ in $\mathcal{C}$ with respect to $H$: those contained in $H$, which are left out of $\mathcal{C}/H$, those disjoint from $H$, which are passed to $\mathcal{C}/H$ undisturbed, and those that contain $H$, which suffer the same transformation as $U$ before entering $\mathcal{C}/H$.

Now we need a few definitions regarding the permutations in each of these sets. If $\beta$ is a permutation on $U/H$ and $\gamma$ is a permutation on $H$, denote by $\beta[H/\gamma]$ the permutation $\alpha$ obtained from $\beta$ by substituting $\gamma$ for $H$. For instance, if $\beta = abHfd$ and $\gamma = ecg$, then $\beta[H/\gamma] = abecgfd$. If $H$ is clear from the context, we will write simply $\beta * \gamma$ instead of $\beta[H/\gamma]$. Accordingly, if $X$ and $Y$ are sets of permutations of $U/H$ and $H$, respectively, we will write

$$X * Y = \{\beta * \gamma | \beta \in X, \gamma \in Y\}.$$

We are now ready for an important result.

**Theorem 7.** *Given a collection $\mathcal{C}$ and a nonempty set $H \in \overline{\mathcal{C}} \cap \mathcal{C}^{\perp}$ we have*

$$\mathit{valid}(U, \mathcal{C}) = \mathit{valid}(U/H, \mathcal{C}/H) * \mathit{valid}(H, \mathcal{C} \cap \mathcal{P}(H)).$$

*Proof.* Given $\alpha \in \mathit{valid}(U, \mathcal{C})$, the elements of $H$ appear consecutive in $\alpha$, since $H \in \overline{\mathcal{C}}$. Hence, $\alpha$ can be written uniquely as

$$\alpha = \beta[H/\gamma],$$

where $\beta$ is a permutation on $U/H$ and $\gamma$ is a permutation on $H$.

Now for any $A \in \mathcal{C}$ we have that $A \cap H$ is equal to either $\emptyset$ or $A$ or $H$, since $H \in \mathcal{C}^{\perp}$. If $A \cap H = \emptyset$ or $H$, $\gamma$ is clearly valid for $A \cap H$. If $A \cap H = A$, $\gamma$ is valid for $A \cap H$ because $\alpha$ is valid for $A \in \mathcal{C}$. Hence, $\gamma \in \mathit{valid}(H, \mathcal{C} \cap \mathcal{P}(H))$.

Let $A/H$ be an element of $\mathcal{C}/H$. Recall that $\alpha$ is valid for $A \in \mathcal{C}$. Now if $H \subseteq A$ we have of course $\beta$ valid for $A/H$, since $H$ was contracted in both $\alpha$ and $A$ to a single element. If $H \cap A = \emptyset$ then the section involving $A$ in $\alpha$ is not changed, that is, remains consecutive in $\beta$. In both cases $\beta$ is valid for $A/H$. This shows that $\beta \in \mathit{valid}(U/H, \mathcal{C}/H)$.

Conversely, it is easy to see that if $\beta \in \mathit{valid}(U/H, \mathcal{C}/H)$ and $\gamma \in \mathit{valid}(H, \mathcal{C} \cap \mathcal{P}(H))$, then $\beta[H/\gamma] \in \mathit{valid}(U, \mathcal{C})$, because $H$ is orthogonal.

**Corollary 8.** *Let $\mathcal{C}$ be a collection of subsets of $U$ and $\emptyset \neq H \in \overline{\mathcal{C}} \cap \mathcal{C}^{\perp}$. If there is a PQR-tree $T$ that encodes all valid permutation for $(U/H, \mathcal{C}/H)$, and also a PQR-tree $T'$ that encodes all valid permutations for $(H, \mathcal{C} \cap \mathcal{P}(H))$, then a PQR-tree for $\mathcal{C}$ can be obtained by replacing the leaf $H$ in $T$ by $T'$.*

### 3.1 Finding sets in $\overline{\mathcal{C}} \cap \mathcal{C}^{\perp}$

Nonempty sets in $\overline{\mathcal{C}} \cap \mathcal{C}^{\perp}$ are called **node sets** because they are exactly the sets of leaves descending from a given node in the PQR-tree of $\mathcal{C}$, as we will see.

Here we describe two ways of finding node sets. These methods will be sufficient for our purposes.

One way of obtaining node sets is as follows. Call two sets $A$ and $B$ **strictly overlapping** if $A \not\perp B$. This is a symmetric relation. Taking the reflexive-transitive closure of this relation we get an equivalence relation on $\mathcal{C}$. For each equivalence class, take the union of all its members. These unions always form node sets, which correspond to either Q- or R-nodes, provided that they have at least two elements. An adaptation of a recent algorithm by Meidanis and Munuera [9] gives a linear time method for producing such components. Alternatively, Hsu [6] gives another linear method to compute them.

After all these components have been factored out, it is easy to determine the remaining node sets. Two elements $a$ and $b$ of $U$ are called **twins** when $\{a,b\} \perp \mathcal{C}$. This is an equivalence relation. It turns out that in instances where all unions of components of strictly overlapping sets are trivial, each class of inseparable elements is a set in $\overline{\mathcal{C}} \cap \mathcal{C}^{\perp}$. Moreover, all nontrivial node sets are of this form. Finding the equivalence classes of inseparable elements can be acomplished in linear time in a standard way [11, 6].

# 4 Characterization of C1P and Algorithm

We saw in the last section how to decompose problem instances into smaller ones. In this section we concentrate on instances that cannot be further decomposed, that is, such that $\overline{\mathcal{C}} \cap \mathcal{C}^{\perp} = \mathcal{T}(U)$. We call such instances **prime** instances. (This term has been used by Hsu [6] to indicate a related but different concept.)

If $|U| = 1$ or 2 every set is trivial and every permutation is valid with respect to every set, so we will only consider prime instances with $|U| \geq 3$. Our strategy is based on a closer examination of $\overline{\mathcal{C}}$. Our main result is the following.

**Theorem 9.** *Let $\mathcal{C}$ be a prime collection of subsets of $U$ with $|U| \geq 3$. Then either*

1. $\overline{\mathcal{C}} = \mathcal{T}(U)$, *or*
2. $\overline{\mathcal{C}} = \mathit{consec}(\alpha)$, *for some permutation $\alpha$ on $U$, or else*
3. $\overline{\mathcal{C}} = \mathcal{P}(U)$.

This result has the following immediate consequences. In case (1), all permutations are valid. Thus, we can represent the valid permutations by a P-node with all elements of $U$ as children. By convention, the case $|U| = 2$ will be handled by a P-node as well.

In case (2) only two permutations are valid, namely, $\alpha$ and its reverse. This is because $|U| \geq 3$. Thus, we can represent all valid permutations by a Q-node with all elements of $U$ as children, in the order given by $\alpha$.

Finally, in case (3) no permutation is valid. The instance $(U, \mathcal{C})$ does not have the C1P. We represent this fact using an R-node with all elements of $U$ as children.

Using this result plus the decomposition theorem of last section, we see that every instance $(U, \mathcal{C})$ has all its valid permutations described by a suitable PQR-tree, and that the C1P is equivalent to absence of R-nodes. In addition, we have the recursive algorithm of Figure 3 for constructing a PQR-tree for $(U, \mathcal{C})$.

## 4.1 Complexity Analysis

We saw in Section 3.1 that finding a nontrivial set in $\overline{\mathcal{C}} \cap \mathcal{C}^{\perp}$ can be done in linear time. To check whether a collection $\mathcal{C}$ is trivial can also be easily done

```
Function PQR-Tree(U, C)
    if there is a nontrivial set H in C̄ ∩ C^⊥ then
        T_1 ← PQR-Tree(U/H, C/H)
        T_2 ← PQR-Tree(U ∩ H, C ∩ P(H))
        return T_1 with leaf H replaced by T_2
    else
        { C is prime }
        case
            C trivial:           return P-node(U)
            ∃α ∈ valid(C):       return Q-node(α)
            otherwise:           return R-node(U)
        endcase
    endif
```

**Fig. 3.** A recursive algorithm to construct PQR-trees.

in linear time. A straightforward adaptation of a partitioning algorithm used by Figueiredo et al. [4] to recognize proper interval graphs can be used to check whether a prime collection $\mathcal{C}$ admits at least one valid permutation $\alpha$, and also recover $\alpha$, in linear time. While this may seem like solving the whole problem (i.e., knowing whether there is a valid permutation), the fact that we have a prime collection greatly simplifies the job, and a simple partitioning algorithm suffices here.

Thus, apart from the recursive calls, the algorithm requires $O(n + m + r)$ time. To establish the total running time, we need to assess the time spent on recursive calls. Since the division into two subproblems can be very unbalanced, in the worst case one of the subproblems will have constant size, while the other will have total minus this constant. Hence, we end up with a worst case bound of $O(s^2)$, where $s = n + m + r$.

## 5 Conclusion

We proposed a new theory to deal with the consecutive ones property of binary matrices. As a result of the theory, we managed to extend the notion of PQ-tree to create a new structure, the PQR-tree, that exists for all matrices, regardless of the C1P. The matrix has the C1P if and only if this structure, which is uniquely defined up to equivalence, does not contains R-nodes. If a PQR-tree does contain R-nodes, these nodes give information about which sets and/or elements are responsible for C1P failure.

We also presented a simpler, but still efficient, algorithm for testing for the C1P and constructing the appropriate tree. In practice, it is sometimes preferable to use an easily implementable algorithm over one that is cumbersome to implement. In addition, there is hope of reducing the complexity of our algorithm to linear by clever reusage of information obtained in previous calls.

## References

1. F. S. Annexstein and R. P. Swaminathan. On testing consecutive-ones property in parallel. In *Proc. 7th Annual ACM Symposium on Parallel Algorithms and Architectures SPAA'95*, pages 234–243, Santa Barbara, California, July 1995. U. of Cincinnati.

2. K. S. Booth and G. S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *J. Comput. Sys. Sci.*, 13(3):335–379, December 1976.

3. D. G. Corneil, H. Kim, S. Natarajan, S. Olariu, and A. P. Sprague. Simple linear time recognition of unit interval graphs. *Information Perocessing Letters*, 55:99–104, 1995.

4. C. M. H. de Figueiredo, J. Meidanis, and C. P. de Mello. A linear-time algorithm for proper interval graph recongnition. *Information Processing Letters*, 56:179–184, 1995.

5. S. P. Ghosh. On the theory of consecutive storage of relevant records. *Jounal Inform. Sci.*, 6:1–9, 1973.

6. W. L. Hsu. A simple test for the consecutive ones property. In *Proc. of the ISAAC'92*, volume 650 of *LNCS*, pages 459–469. Springer-Verlag, 1992.

7. D. G. Kendall. Incidence matrices, interval graphs and seriation in archaeology. *Pacific J. Math*, 28(3):565–570, 1969.

8. N. Korte and R. H. Möring. An incremental linear-time algorithm for recognizing interval graphs. *SIAM J. Comput.*, 18:68–81, 1989.

9. J. Meidanis and E. G. Munuera. A simple linear time algorithm for binary phylogeny. In N. Ziviani, J. Piquer, B. Ribeiro, and R. Baeza-Yates, editors, *Proc. of the XV International Conference of the Chilean Computing Society*, pages 275–283, Nov 1995.

10. Mark B. Novick. Generalized PQ-trees. Technical Report 1074, Dept. of Computer Science, Cornell University, Ithaca, NY 14853-7501, Dec 1989. Available from `http://www.cs.cornell.edu`.

11. Tarjan R. E. Rose, D. J. and Lueker G. S. Algorithmic aspects of vertex elimination on graphs. *SIAM J. Computing*, 5(2):266–283, June 1976.

This article was processed using the LaTeX $2_\varepsilon$ macro package with CUP_CS class