ELSEVIER Discrete Applied Mathematics 88 (1998) 325–354

DISCRETE APPLIED MATHEMATICS

On the consecutive ones property

João Meidanis[a,*,1], Oscar Porto[b,2], Guilherme P. Telles[c,3]

[a] *Institute of Computing, University of Campinas, Brazil*
[b] *Department of Electrical Engineering, Catholic University of Rio de Janeiro, Brazil*
[c] *Institute of Computing, University of Campinas, Brazil*

Received 14 September 1996; received in revised form 21 December 1997; accepted 22 December 1997

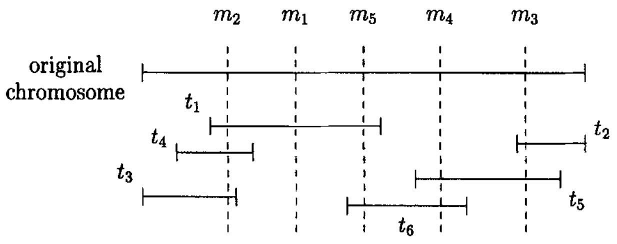
Abstract

A binary matrix has the Consecutive Ones Property (C1P) when there is a permutation of its rows that leaves the 1's consecutive in every column. We study the recognition problem for these matrices, giving a structure, PQR trees, generalizing the PQ trees of Booth and Lueker (1976). This new structure is capable of, not only recording all valid permutations when the matrix has the C1P, but also pointing out possible obstructions when the property does not hold. We recast the problem using collections of sets, developing a new theory for it. This problem appears naturally in several applications in molecular biology, for instance, in the construction of physical maps from hybridization data. © 1998 Elsevier Science B.V. All right reserved.

1. Introduction

A binary matrix has the Consecutive Ones Property (C1P) for columns when there is a permutation of its rows that leaves the 1's consecutive in every column. One can analogously define the C1P for rows. It appears naturally in a wide range of applications [2, 3, 5, 6]. Essentially, the C1P finds applications in any problem in which we are required to linearly arrange a set of objects subjected to restrictions of the form: objects in a given subset must appear consecutively in this order. In molecular biology, one such problem arose when Benzer [1], in 1959, performed a series of experiments aimed at verifying whether a chromosome was a linear arrangement of genes. In combinatorial terms, he had the adjacency relations defining a graph and wanted to know whether it was an interval graph. He solved the problem using a certain characterization of interval graphs that did not require the C1P, but the C1P can be used to recognize interval graphs as well [2, 4]. In any case, this application is

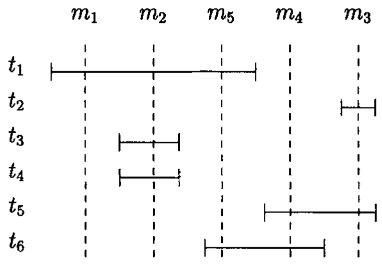
* Corresponding author. E-mail: meidanis@dcc.unicamp.br.
[1] Partially supported by grants from Brazilian agencies FAPESP and CNPq.
[2] Partially supported by grants from Brazilian agencies CNPq and FAPERJ.
[3] Supported by grants from Brazilian agencies FAPESP and CNPq.



(a)

	m_1	m_2	m_3	m_4	m_5
t_1	1	1	0	0	1
t_2	0	0	1	0	0
t_3	0	1	0	0	0
t_4	0	1	0	0	0
t_5	0	0	1	1	0
t_6	0	0	0	1	1

(b)



(c)

Fig. 1. Example of a hybridization experiment.

of historical interest only since we know now that genes are in fact arranged linearly on a chromosome.

A more recent application is the construction of physical maps by hybridization. In this application, several contiguous stretches, possibly overlapping, are extracted from a long DNA molecule. These stretches are called clones, and the problem is to place them correctly along the DNA molecule. The only available information is the outcome of hybridization experiments, where a given probe is tested against each of the clones. A probe corresponds to a uniquely defined position in the DNA, and, in the absence of experimental errors, the only clones that will hybridize with it are the ones that contain this position. The problem here is to infer the correct positioning of the clones based on this data. An example is given in Fig. 1(a)–(c). There, six clones, represented by

$t_1, \ldots, t_6$, are tested against five probes, represented by $m_1, \ldots, m_5$, and the resulting matrix is shown in Fig. 1(b). This matrix has the C1P for rows, and one possible permutation of its columns that leaves the ones consecutive on its rows leads to the solution presented in Fig. 1(c). Notice that the solution is different from the original arrangement, illustrating the fact that, in some cases, the data given is not enough to recover the original order. More experiments involving new probes must be done. As Greenberg and Istrail [6] point out, it is important to know all possible solutions in this application. This knowledge can help the scientist in (1) verifying if the data is enough to produce a unique solution, and (2) concluding that experimental errors occurred, in case there is no solution, and locating the errors. The discussion in Section 7 is related to these issues.

1.1. Previous work

The C1P has been widely studied. The first mention to this property, according to Kendall [8], was made by Petrie, an archaeologist, in 1899. Some heuristic methods were proposed for the problem [11] before Fulkerson and Gross [4] presented the first polynomial complexity solution. In 1972, Tucker [13] presented a characterization of the problem based on forbidden configurations for matrices.

Booth and Lueker [2] proposed the first linear complexity solution for the problem in 1976, in addition to a compact data structure able to elegantly represent all valid permutations. This structure, called PQ tree, has a complicated implementation, despite its nice theory. In 1992, Hsu [7] also presented a linear complexity solution for the problem without using PQ trees and avoiding its complexity of implementation, but the details of his method are still difficult to grasp.

In 1996, Meidanis and Munuera [9] proposed a new theory and a simple extension of Booth and Lueker's structures. The new trees, called PQR trees, exist for any instance and carry extra information that can be useful in several applications. For instance, they can help finding erros in the data (Section 7), or finding all orthogonal sets (Definition 17), a problem already solved by Novick [10]. In the present work we complete the formalization of the theory proposed by Meidanis and Munuera, including and proving some important results left behind in the earliest version of the new approach. This paper can be seen as a complete version of that conference paper.

1.2. Contribution of this work

As mentioned earlier, the C1P can be seen as the problem of producing a linear order compatible to restrictions given by subsets of the elements. In this sense, the input instance consists of a collection $\mathscr{C}$ of subsets, and this is the point of view adopted in this paper.

We believe the main contribution of this work is a fresh way of thinking about the problem, that, in our opinion, will bring a much deeper understanding of its combinatorial nature. The introduction of the completion $\overline{\mathscr{C}}$ of a collection $\mathscr{C}$ (Definition 8), composed of all sets that must be consecutive if the sets in $\mathscr{C}$ are consecutive, is an important step for two reasons. First, it allows a "normalization" of collections, in the sense that two collections with the same completion impose essentially equivalent restrictions on the linear order. Second, it helps in the construction of the tree that will represent all solutions, because the domain of every tree node is a set in the completion. Three operations (intersection, nondisjoint union, and noncontained difference) applied repeatedly, are sufficient to generate $\overline{\mathscr{C}}$ from $\mathscr{C}$.

The concept of orthogonality (Definition 16) has been important to the C1P ever since the work of Fulkerson and Gross [4], although with different names. Fulkerson and Gross say that two sets *overlap* when they have a nonempty intersection that is properly contained in both. Booth and Lueker [2] do not use this concept explicitly. Novick [10], uses the notation $A \sim B$ for this concept, and Hsu [7] calls the sets *strictly overlapping*, which is the terminology we adopt here as well. This is all related to orthogonality, which is just the opposite concept: two sets are orthogonal when they do *not* strictly overlap. Another contribution of this work is the characterization of domains of the tree nodes as sets in the completion that are orthogonal to all sets in $\mathscr{C}$. If we call $\mathscr{C}^{\perp}$ the collection of sets orthogonal to all sets in $\mathscr{C}$, the nodes correspond to sets in $\overline{\mathscr{C}} \cap \mathscr{C}^{\perp}$. Novick uses the notation $M(T)$ to indicate $\mathscr{C}^{\perp}$.

Other important results include a decomposition of valid permutations based on sets in $\overline{\mathscr{C}} \cap \mathscr{C}^{\perp}$, and an extension of Hsu's definition of prime instances for the case of nonnormalized collections.

The inspection of possible types of prime collections lead to the discovery of another internal node type, besides the already known P and Q. We called the new type R node, and the new tree, PQR tree. These trees generalize both the PQ tree of Booth and Lueker [2] and the gPQ tree of Novick [10]. Essentially, a PQ tree is a PQR tree without R nodes, and a gPQ tree is a PQR tree where R nodes are transformed into Q nodes, and the order of children is ignored. The presence of R nodes indicate that the instance does not have the C1P, and, in addition, R nodes can help in some cases to identify why the C1P does not hold. To illustrate the power of the new techniques, we prove again, in a much more uniform way, several known facts about PQ trees, for instance, the fact that they represent all valid permutations (Theorem 29).

The rest of this paper is organized as follows. Section 2 gives the basic definitions needed throughout. (Additional, specific definitions appear in the rest of the paper as needed.) The PQR trees are introduced in Section 3. The new theory that leads to the construction of PQR trees is introduced in Section 4. The main results of the paper are developed in Section 5. An algorithm for constructing PQR trees is given in Section 6, along with proofs of correctness and complexity analysis. Section 7 contains examples that illustrate the behavior of PQR trees in the presence of errors. Finally, Section 8 contains our concluding remarks.

2. Basic definitions

The *consecutive ones property* is a property of two-dimensional matrices whose entries are only 0 or 1. These matrices are called *binary matrices* and a binary matrix has the consecutive ones property (C1P) for columns when its rows can be permuted so that in each of its columns the 1's appear consecutively. The permutations that leave the ones in such order are called *valid permutations.* The C1P for rows is analogous and the problem of testing one of them can be replaced by the other, simply by transposing the matrix.

It can also be viewed as a property of collections of sets. The term *collection* will be used here as a synonym for set of sets. In this work, collections will always be denoted by calligraph capitals, such as $\mathscr{C}$. In order to explain the connection better, we need a few definitions. A *permutation* of a finite set U, of size n, is a bijection $\alpha:\{1,2,\dots,n\}\mapsto U$. We denote the set of all permutations of U by $Perm(U)$. The set of all subsets of U will be indicated by $Subsets(U)$. To simplify notation, we sometimes write a set as a list of its elements in any order. For example, $A=\{k,l,m,n\}$ can be written as $A=lnkm$.

Given this finite set U, a permutation α of the elements of U, and a subset A of U, we say that A is *consecutive* in α when the elements of A appear consecutively in α. For example, if $U=abcdef$ and $\alpha=efcbda$, the subset $A=cdb$ is consecutive in α, while $B=efa$ is not. Given a pair $(U,\mathscr{C})$, with $\mathscr{C}\subseteq Subsets(U)$, we say that a permutation α of U is *valid* (with respect to $\mathscr{C}$) if all sets $A\in\mathscr{C}$ are consecutive in α. The pair $(U,\mathscr{C})$ has the C1P if there is at least one valid permutation.

It is possible to view one application of the property as the other, as follows: 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 as a pair $(U,\mathscr{C})$ where $\mathscr{C}$ is the collection of the subsets of U that correspond to its columns.

Example 1. For example, take the matrix below with its rows labeled:

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

A permutation of its rows shows that this matrix has the C1P:

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

In terms of sets, we have

$$U=\{a,b,c,d,e\},$$
$$A=\{a,b,d\},$$
$$B=\{a,c,d\},$$
$$C=\{b,d,e\},$$

where the sets A,B, and C form the collection $\mathscr{C}$. The permutation $\alpha = cadbe$ is valid and shows that the pair $(U,\mathscr{C})$ has the C1P.

Example 2. As a counter-example take the matrix below:

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

$$U=\{a,b,c\},$$
$$A=\{b,c\},$$
$$B=\{a,b\},$$
$$C=\{a,c\}.$$

There is no permutation that leaves b next to c and a next to b and c at the same time.

In this work we will be interested in the complexity of recognizing whether a given instance has the C1P and of finding valid permutations. For this we need a notion of size of an input instance. Following previous work in this area we will define $n+m+r$ to be the size of an instance, where n is the number of rows, m is the number of columns of the binary matrix and r is the number of ones in the matrix. If we use sets, $n=|U|$, $m=|\mathscr{C}|$ and $r=\sum_{A\in\mathscr{C}}|A|$.

3. PQR trees

PQR trees, defined by Meidanis and Munuera [9], are an extension of Booth and Lueker's PQ trees [2].

A PQR tree over a set U is a rooted tree with three different types of internal nodes: P, Q and R. Its leaves are elements of U, without repetition. The internal nodes of a PQR tree must satisfy the following rules:

(1) Each P node has at least two children.
(2) Each Q node has at least three children.
(3) Each R node has at least three children.

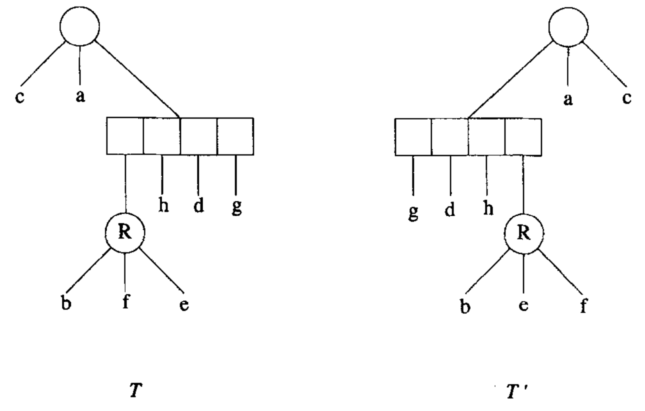


Fig. 2. Two equivalent PQR trees.

We define the *domain* of a PQR tree T, $Dom(T)$, as the union of its leaves. As mentioned earlier, $Dom(T)$ is a subset of U.

The way a PQR tree represents all valid permutations for a pair $(U,\mathscr{C})$ with the C1P is by admitting some equivalence transformations over its internal nodes. Each transformation specifies a valid reordering of the children of a node. There are three kinds of valid reorderings for PQR trees:

(1) arbitrary permutations of the children of a P node,
(2) reversal of the children of a Q node, and
(3) arbitrary permutations of the children of an R node.

Two trees are *equivalent*, $T \equiv T'$, if and only if one of them can be transformed into the other by applying zero or more equivalence transformations.

Fig. 2 shows two equivalent PQR trees. We represent Q nodes by rectangular boxes, and P and R nodes by circles, distinguishing R nodes with an "R" inside the circle.

Reading the leaves of a PQR tree, from left to right, yields a permutation of its domain. This permutation is called the *frontier* of the PQR tree. For example, the PQR tree T in Fig. 2 has frontier *cabfehdg* while the tree T' has frontier *gdhbefac*. The frontier of a tree is denoted by $Frontier(T)$. We expand this definition.

Definition 3. Given a PQR tree T, define

$$Compat(T) = \{Frontier(T')\colon\ T' \equiv T\}.$$

This is the set of all permutations that are frontiers of trees equivalent to T. We say that these permutations are *compatible* with T.

PQR trees can always be built for a pair $(U,\mathscr{C})$, which is not the case for PQ trees. For the former, if the instance $(U,\mathscr{C})$ does not have the C1P, no tree is going to be built. In such a case, the PQR tree will be built having at least one R node, which may indicate the elements that obstruct the property. This is not always the case, as we shall see in Section 7. When the instance has the C1P, the PQR tree is equivalent to the PQ tree for the same instance.

In the construction of PQR trees two new collections are characterized: $\overline{\mathscr{C}}$ and $\mathscr{C}^{\perp}$. These two collections, formally described in the following sections, are used to divide the problem of the C1P into two smaller instances, using sets in $\overline{\mathscr{C}} \cap \mathscr{C}^{\perp}$. We shall see later that every nonempty set in $\overline{\mathscr{C}} \cap \mathscr{C}^{\perp}$ corresponds to a node in the PQR tree for $\mathscr{C}$.

4. The new theory

We present the results of this section grouped in some subsections. Before introducing the new theory, we make some additional definitions.

Definition 4. For a finite set U, define the *trivial subsets* of U as being the empty set, the singletons and U itself. So, $\mathscr{T}(U) = \{\emptyset, U\} \cup \{\{a\}: a \in U\}$ is the collection of the trivial subsets of U.

Definition 5. We will denote by *Valid*$(U, \mathscr{C})$ or sometimes just by *Valid*$(\mathscr{C})$ the set of all valid permutations with respect to $(U, \mathscr{C})$.

Definition 6. Similarly, *Consec*(α) is the collection of all sets $A \subseteq U$ that are consecutive in α. If S is a set of permutations, we define *Consec*(S) as the collection of sets consecutive in every $\alpha \in S$, that is,

$$\mathit{Consec}(S) = \bigcap_{\alpha \in S} \mathit{Consec}(\alpha).$$

4.1. New collections: $\overline{\mathscr{C}}$ and $\mathscr{C}^{\perp}$

The very first observation is that if the sets in $\mathscr{C}$ are to be consecutive, other sets must also be consecutive. They are defined by the operations:

(1) the *intersection* $A \cap B$ of two sets in $\mathscr{C}$,
(2) the union $A \cup B$ of two sets in $\mathscr{C}$, provided that $A \cap B \neq \emptyset$ (*nondisjoint union*), and
(3) the relative complement $A \backslash B$ of two sets in $\mathscr{C}$, provided that $B \nsubseteq A$ (*noncontained complement*).

In addition, the singletons $\{a\}$ for $a \in U$ and U itself are always consecutive in any permutation. By convention, the empty set is always consecutive as well. Hence, the trivial subsets are always consecutive.

Definition 7. A collection $\mathscr{C}$ of subsets of U is *complete* if it contains all the trivial subsets of U and is closed under operations (1), (2) and (3) above.

Definition 8. Given a collection $\mathscr{C}$, the *completion* of $\mathscr{C}$, denoted by $\overline{\mathscr{C}}$, is the smallest complete collection that contains $\mathscr{C}$. The collection $\overline{\mathscr{C}}$ is well defined since it is the intersection of all complete supercollections of $\mathscr{C}$.

Example 9. If $U = abcde$ and $\mathscr{C} = \{ab, be, cde\}$, then

$$\overline{\mathscr{C}} = \mathscr{T}(U) \cup \{ab, be, cde, abe, bcde, cd\}.$$

Example 10. For the same U, collection $\mathscr{C} = \{ac\}$ has completion $\overline{\mathscr{C}} = \mathscr{C} \cup \mathscr{T}(U)$.

Immediate consequences of these definitions are given below.

Theorem 11. *Given two collections $\mathscr{C}$, $\mathscr{D}$ over the same set U, we have*
(1) $\mathscr{T}(U) \subseteq \overline{\mathscr{C}}$,
(2) $\mathscr{C} \subseteq \overline{\mathscr{C}}$,
(3) $\mathscr{C} \subseteq \mathscr{T}(U) \Leftrightarrow \overline{\mathscr{C}} = \mathscr{T}(U)$,
(4) $\mathscr{C} \subseteq \mathscr{D} \Leftrightarrow \overline{\mathscr{C}} \subseteq \overline{\mathscr{D}}$.

We now investigate the relationship between completions and valid permutations. To this effect we prove some lemmas.

Since a permutation α is a sequence of elements, we will denote by $\alpha(i)$ the element of α that is at position i in α and by $\alpha^{-1}(x)$ the position occupied by element $x \in U$. By $\alpha^{-1}(A)$ we mean the set of indices i such that $\alpha(i) \in A$. If A is consecutive, then $\alpha^{-1}(A)$ is an interval $[i..j]$.

Lemma 12. *If the sets $A, B \in \mathscr{C}$ are consecutive in α, $\alpha \in Valid(\mathscr{C})$, then $A \cap B$ is consecutive in α.*

Proof. Let $\alpha^{-1}(A) = [i..j]$ and $\alpha^{-1}(B) = [k..l]$. If $A \cap B = \emptyset$ it is consecutive in α. Otherwise, the intersection is given by interval $[max(i,k)..min(j,l)]$, and is consecutive in α. □

Lemma 13. *If the sets $A, B \in \mathscr{C}$ are consecutive in α, $\alpha \in Valid(\mathscr{C})$ and $A \cap B \neq \emptyset$, then $A \cup B$ is consecutive in α.*

Proof. Let $\alpha^{-1}(A) = [i..j]$ and $\alpha^{-1}(B) = [k..l]$. Suppose, without loss of generality, $i \leqslant k$. If $l \leqslant j$, then $B \subseteq A$ and $A \cup B = A$, which is consecutive in α. Otherwise, given that $A \cap B \neq \emptyset$, we have $\alpha^{-1}(A \cup B) = [i..l]$ and $A \cup B$ is consecutive in α. □

Lemma 14. *If the sets $A, B \in \mathscr{C}$ are consecutive in α, $\alpha \in Valid(\mathscr{C})$ and $B \nsubseteq A$, then $A \backslash B$ is consecutive in α.*

Proof. Let $\alpha^{-1}(A) = [i..j]$ and $\alpha^{-1}(B) = [k..l]$.

If $l < i$ or $j < k$, the sets are disjoint and then $A \backslash B = A$, which is consecutive in α. If $i \leqslant k$ and $l \leqslant j$ then $B \subseteq A$. Otherwise, we have,

- if $k \leqslant i$ and $l < j$ then $A \backslash B = [l+1..j]$,
- if $k \leqslant i$ and $j \leqslant l$ then $A \backslash B = \emptyset$,
- if $i \leqslant k$ and $j < l$ then $A \backslash B = [i..k-1]$.

In all cases, the resulting interval is consecutive in α, and this concludes our proof. □

An important relation follows:

Theorem 15. *For any collection* $\mathscr{C}$,

$$Valid(\mathscr{C}) = Valid(\overline{\mathscr{C}}).$$

Proof. It is enough to prove that $Valid(\mathscr{C}) \subseteq Valid(\overline{\mathscr{C}})$ because $\mathscr{C} \subseteq \overline{\mathscr{C}}$ guarantees that $Valid(\overline{\mathscr{C}}) \subseteq Valid(\mathscr{C})$. By Lemmas 12–14, we can see that the sets obtained by operations (1)–(3) are all consecutive in a permutation $\alpha \in Valid(\mathscr{C})$. Since this is also true for every trivial set we can conclude that $Valid(\mathscr{C}) \subseteq Valid(\overline{\mathscr{C}})$. □

Now we define the collection $\mathscr{C}^{\perp}$.

Definition 16. 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 17. Given a subset A of U and a collection $\mathscr{C} \subseteq Subsets(U)$, we write $A \perp \mathscr{C}$ when $A \perp B$ for every $B \in \mathscr{C}$. Such a set A is said to be orthogonal to $\mathscr{C}$. The collection of all sets orthogonal to $\mathscr{C}$ is denoted by $\mathscr{C}^{\perp}$.

We write $\mathscr{C} \perp \mathscr{D}$ to indicate that $A \perp B$ for all $A \in \mathscr{C}$ and $B \in \mathscr{D}$.

Easy consequences of these definitions follow.

Theorem 18. *Given two collections* $\mathscr{C}$, $\mathscr{D}$ *over the same set* U, *we have*

(1) $\mathscr{T}(U) \subseteq \mathscr{C}^{\perp}$,

(2) $\mathscr{C} \subseteq \mathscr{D} \Rightarrow \mathscr{C}^{\perp} \supseteq \mathscr{D}^{\perp}$.

An interesting relation involving composition of the operators previously defined follows.

Theorem 19. *If* $\mathscr{A} \perp \mathscr{B}$ *then* $\bar{\mathscr{A}} \perp \mathscr{B}$.

Proof. Let $\mathscr{A}$ and $\mathscr{B}$ be as in the statement of the theorem. Suppose we have $A, B \in \mathscr{A}$ and $H \in \mathscr{B}$. Then $A \perp H$ and $B \perp H$. Table 1 proves that the sets $A \cap B$, $A \cup B$ when $A \cap B \neq \emptyset$ and $A \backslash B$ when $B \nsubseteq A$ are all orthogonal to H.

Table 1
Possible outcomes of intersection, union, and complement for sets

$\cap$	$B\cap H=\emptyset$	$B\subseteq H$	$H\subseteq B$
$A\cap H=\emptyset$	$(A\cap B)\cap H=\emptyset$	$(A\cap B)\cap H=\emptyset$	$(A\cap B)\cap H=\emptyset$
$A\subseteq H$	$(A\cap B)\cap H=\emptyset$	$(A\cap B)\subseteq H$	$A\cap B=A$
$H\subseteq A$	$(A\cap B)\cap H=\emptyset$	$A\cap B=B$	$(A\cap B)\supseteq H$
$\cup$	$B\cap H=\emptyset$	$B\subseteq H$	$H\subseteq B$
$A\cap H=\emptyset$	$(A\cup B)\cap H=\emptyset$	$A\cap B=\emptyset$	$(A\cup B)\supseteq H$
$A\subseteq H$	$A\cap B=\emptyset$	$(A\cup B)\subseteq H$	$(A\cup B)\supseteq H$
$H\subseteq A$	$(A\cup B)\supseteq H$	$(A\cup B)\supseteq H$	$(A\cup B)\supseteq H$
$\backslash$	$B\cap H=\emptyset$	$B\subseteq H$	$H\subseteq B$
$A\cap H=\emptyset$	$(A\backslash B)\cap H=\emptyset$	$A\backslash B=A$	$(A\backslash B)\cap H=\emptyset$
$A\subseteq H$	$A\backslash B=A$	$A\backslash B\subseteq H$	$A\backslash B=\emptyset$
$H\subseteq A$	$A\backslash B\supseteq H$	$B\subseteq A$	$(A\backslash B)\cap H=\emptyset$

In addition, trivial sets are orthogonal to any other set. Since we construct $\bar{\mathscr{A}}$ by repeated use of operations (1)–(3) given at the beginning of Section 4.1, and adding trivial sets, this implies that $\bar{\mathscr{A}}\perp\mathscr{B}$. □

We can now use the previous result to prove:

Theorem 20. *For any collection* $\mathscr{C}$,

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

Proof. Applying Theorem 19 on the relation $\mathscr{C}\perp\mathscr{C}^{\perp}$, and observing that $\mathscr{A}\perp\mathscr{B}$ implies that $\mathscr{A}\subseteq\mathscr{B}^{\perp}$ and that $\mathscr{B}\subseteq\mathscr{A}^{\perp}$ we derive:

$$\mathscr{C}\perp\overline{\mathscr{C}^{\perp}}\Rightarrow\overline{\mathscr{C}^{\perp}}\subseteq\mathscr{C}^{\perp},$$

$$\overline{\mathscr{C}}\perp\mathscr{C}^{\perp}\Rightarrow\mathscr{C}^{\perp}\subseteq(\overline{\mathscr{C}})^{\perp}.$$

On the other hand, a collection is always contained in its completion. Hence, $\mathscr{C}^{\perp}\subseteq\overline{\mathscr{C}^{\perp}}$ and this implies the first equality in the theorem.

Using Theorem 18, from $\mathscr{C}\subseteq\overline{\mathscr{C}}$ we see that $\mathscr{C}^{\perp}\supseteq(\overline{\mathscr{C}})^{\perp}$. Then $\mathscr{C}^{\perp}=(\overline{\mathscr{C}})^{\perp}$. □

4.2. *Decomposition*

Now we are going to see how the problem can be nicely decomposed using a set in $\overline{\mathscr{C}}\cap\mathscr{C}^{\perp}$. We need the following definition.

Definition 21. Define

$$A/H=\begin{cases}(A\backslash H)\cup\{H\} & \text{if } H\subseteq A,\\ A & \text{if } H\cap A=\emptyset\end{cases}$$

and the inverse operation

$$A \bowtie H = \begin{cases} (A\backslash\{H\}) \cup H & \text{if } H \in A, \\ A & \text{if } H \notin A. \end{cases}$$

Given a pair $(U, \mathscr{C})$ with $\mathscr{C} \subseteq \mathit{Subsets}(U)$ and a subset $H \in \overline{\mathscr{C}} \cap \mathscr{C}^{\perp}$, the first subinstance of the problem is defined taking the pair $(H, \mathscr{C} \cap \mathit{Subsets}(H))$. We denote this subinstance by $\mathscr{C} \wedge H$.

The other subinstance of the problem is $(U/H, \mathscr{C}/H)$. The collection $\mathscr{C}/H$ is defined as: $\mathscr{C}/H = \{A/H\colon\ A \in \mathscr{C},\ A \not\subset H\}$. We denote this subinstance by $\mathscr{C}/H$.

When the problem cannot be decomposed any further, we have the instances such that $\overline{\mathscr{C}} \cap \mathscr{C}^{\perp} = \mathscr{T}(U)$. These instances are called *prime* and correspond to PQR trees with one internal node.

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

$$X \star Y = \{\beta \star \gamma\colon\ \beta \in X,\ \gamma \in Y\}.$$

An important result is what follows. We observe that no tree is necessary for this conclusion.

Theorem 22. *Given a collection $\mathscr{C}$ and a nonempty set $H \in \overline{\mathscr{C}} \cap \mathscr{C}^{\perp}$ we have*

$$\mathit{Valid}(U, \mathscr{C}) = \mathit{Valid}(U/H, \mathscr{C}/H) \star \mathit{Valid}(H, \mathscr{C} \wedge H).$$

Proof. Given $\alpha \in \mathit{Valid}(U, \mathscr{C})$, the elements of H appear consecutively in α, since $H \in \overline{\mathscr{C}}$. Hence, α can be written uniquely as $\alpha = \beta[H/\gamma]$, where β is a permutation on U/H and γ is a permutation on H.

Now for any $A \in \mathscr{C}$, $A \cap H$ must be either $\emptyset$ or A or H, since $H \in \mathscr{C}^{\perp}$. If $A \cap H = \emptyset$ or H, γ is clearly valid for $A \cap H$. If $A \cap H = A$, γ is valid for $A \cap H$ because α is valid for $A \in \mathscr{C}$. Hence, $\gamma \in \mathit{Valid}(H, \mathscr{C} \wedge H)$.

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

Conversely, it is easy to see that if $\beta \in \mathit{Valid}(U/H, \mathscr{C}/H)$ and $\gamma \in \mathit{Valid}(H, \mathscr{C} \wedge H)$, then $\beta[H/\gamma] \in \mathit{Valid}(U, \mathscr{C})$, because H is orthogonal. $\square$

4.3. *Prime instances*

In this section we inspect the prime instances, using an auxiliary graph defined below.

Definition 23. Define the graph $\mathscr{B}(U,\mathscr{C})=(V,E)$, where $V=U$ and $E=\{ab:\{a,b\}\in\overline{\mathscr{C}}\}$. This is the graph of *binaries* of $\overline{\mathscr{C}}$. Sometimes we will write just $\mathscr{B}(\mathscr{C})$ for $\mathscr{B}(U,\mathscr{C})$.

Theorem 24. *For any collection $\mathscr{C}$, each connected component of $\mathscr{B}(\mathscr{C})$ is either a path or a complete subgraph.*

Proof. This is obvious for components of up to three vertices. Let A be a component that is not a path and assume that $|A|\geqslant 4$. The component A either has a cycle or a vertex of degree at least three. If A has a cycle involving, say, a, b and c as consecutive elements, then $abc\in\overline{\mathscr{C}}$. There is also another set B in $\overline{\mathscr{C}}$ containing a and c but not b obtained going around the cycle. The intersection $B\cap abc=ac$ is also in $\overline{\mathscr{C}}$ and therefore a is an element of degree at least three in $\mathscr{B}(\mathscr{C})$.

Let x be any vertex of degree at least three. We claim that the neighborhood of x is a complete subgraph. Indeed, if edges ax, bx and cx exist, then $ab=(ax\cup bx)\backslash cx$ is in $\overline{\mathscr{C}}$. Analogously we prove that ac and bc are also in $\overline{\mathscr{C}}$. Continuing this way with the remaining vertices of A, i.e., starting with a vertex of the neighborhood of the complete graph already obtained, then with a vertex of the neighborhood of the new complete graph and so on, we can conclude that A is a complete subgraph. $\square$

In order to prove Lemma 25 we need the following definition. We say that a subset $A\subseteq U$ is *connected* (with respect to $\mathscr{C}$) when A induces a connected subgraph of $\mathscr{B}(\mathscr{C})$.

Lemma 25. *If $\mathscr{C}$ is prime and $A\in\overline{\mathscr{C}}$ is nontrivial, then A is connected.*

Proof. Induction on $|A|$. The result is immediate for $|A|=2$. If $|A|\geqslant 3$, A cannot be orthogonal to $\mathscr{C}$. Hence there is $B\in\mathscr{C}$ with $A\not\perp B$. Then $A\cap B$ and $A\backslash B$ are both in $\overline{\mathscr{C}}$ and strictly smaller than A, so the result is true for them, i.e., they are both connected. Also, since $|A|\geqslant 3$, at least one of them is nontrivial.

By induction hypothesis, both $A\cap B$ and $A\backslash B$ are connected. However, A cannot be written as a nondisjoint union of $A\cap B$ and $A\backslash B$. Our strategy, then, is to use a "bridge", that is, a connected set X that strictly overlaps both $A\cap B$ and $A\backslash B$ and is contained in A. Then, A can be written as a nondisjoint union of connected sets $A\cap B$, X and $A\backslash B$, and is therefore connected.

If $A\cap B$ is nontrivial, let $D=A\cap B$; otherwise, let $D=A\backslash B$. Since in any case D is nontrivial and $D\in\overline{\mathscr{C}}$, we have $D\notin\mathscr{C}^{\perp}$ and then, there is $E\in\mathscr{C}$ such that $D\not\perp E$.

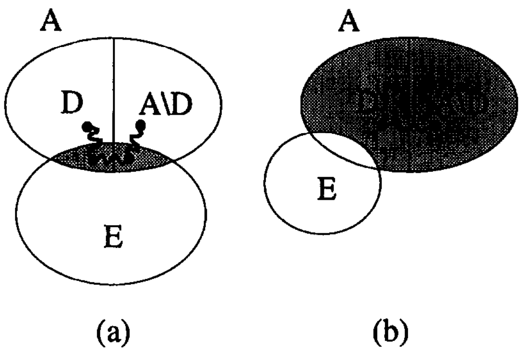


Fig. 3. The set X, shown in gray, acts as a "bridge" between D and $A\backslash D$: (a) $X=A\cap E$; (b) $X=A\backslash E$.

If $E\cap(A\backslash D)\neq\emptyset$, then $(A\cap E)\cap D\neq\emptyset$ and $(A\cap E)\cap(A\backslash D)\neq\emptyset$. Take $X=A\cap E$ (Fig. 3(a)). Since $X\in\overline{\mathscr{C}}$ and is connected by induction, we see that there is a path in X between elements of D and elements of $A\backslash D$. Since both A and $A\backslash D$ are in $\overline{\mathscr{C}}$ and are also connected by induction, we conclude that A is connected. If $E\cap(A\backslash D)=\emptyset$ then E must have some element not in A. But then $(A\backslash E)\cap D\neq\emptyset$ and $(A\backslash E)\cap(A\backslash D)\neq\emptyset$; using the same argument for $X=A\backslash E$ that we used above for $A\cap E$ we can also conclude that A is connected (Fig. 3(b)). □

Theorem 26. *If $\mathscr{C}$ is prime and $\overline{\mathscr{C}}$ is not trivial, then $\mathscr{B}(\mathscr{C})$ is a connected graph.*

Proof. Let A be the largest connected subset of U. Since $\overline{\mathscr{C}}$ is not trivial, by the above lemma we know that $|A|\geqslant 2$. We claim that $A=U$. If not then A is nontrivial and hence $A\notin\mathscr{C}^{\perp}$, since $A\in\overline{\mathscr{C}}$. Then there is $B\in\mathscr{C}$ with $A\not\perp B$ and therefore $A\cup B$ would be a larger connected subset, again by the lemma. This contradiction proves that indeed $A=U$. □

In several of our induction proofs the base case refers to prime collections. It is important to characterize prime instances in order to solve the questions posed in these base cases. For $|U|<3$ every set is trivial and every permutation is valid with respect to every set. The following result is of fundamental importance for larger collections.

Theorem 27. *Let $\mathscr{C}$ be a prime collection of subsets of U with $|U|\geqslant 3$. Then either*
(1) $\overline{\mathscr{C}}=\mathscr{T}(U)$, *or*
(2) $\overline{\mathscr{C}}=\mathit{Consec}(\alpha)$, *for some permutation α on U, or else*
(3) $\overline{\mathscr{C}}=\mathit{Subsets}(U)$.

Proof. If $\mathscr{C}\subseteq\mathscr{T}(U)$ then obviously $\overline{\mathscr{C}}=\mathscr{T}(U)$. If $\mathscr{C}$ is not trivial, then by Theorem 26 we have $\mathscr{B}(\mathscr{C})$ connected. Furthermore, by Theorem 24, $\mathscr{B}(\mathscr{C})$ is either a path or complete.

If $\mathscr{B}(\mathscr{C})$ is a path, this path defines two permutations of the elements of U, one being the reverse of the other: just traverse the path from one extreme to the other. Let α be one of these permutations. Any set consecutive in α can be constructed from the binaries by nondisjoint union. Thus $Consec(\alpha) \subseteq \overline{\mathscr{C}}$. If a set not consecutive in α were present in $\overline{\mathscr{C}}$, then we would be able to construct a binary not consecutive in α using intersection, and $\mathscr{B}(\mathscr{C})$ would not be a path.

Finally, if $\mathscr{B}(\mathscr{C})$ is complete, all binaries belong to $\overline{\mathscr{C}}$, and therefore all subsets of U belong to $\overline{\mathscr{C}}$. □

5. Relation between the C1P and PQR trees

We proceed by giving some definitions and proofs which show that for every binary matrix a PQR tree can be built and that each PQR tree is an instance of a C1P problem.

Let $Desc_T(v)$ be the set formed by all leaves that descend from node v in T:

$$Desc_T(v) = \{a \in U\colon\ v \text{ is an ancestor of } a \text{ in } T\}.$$

Notice that $Desc_T(v)$ is nothing else than the domain of the subtree rooted at v.

We extend this definition to sets S of nodes as follows:

$$Desc_T(S) = \bigcup_{v \in S} Desc_T(v).$$

The collection $Compl(T)$ is defined as follows.

(1) $\mathscr{T}(U)$ is contained in $Compl(T)$.

(2) $Desc_T(S)$ is in $Compl(T)$ if S is the set of all children of a P node of T.

(3) $Desc_T(S)$ is in $Compl(T)$ if S is a set of consecutive children of a Q node of T.

(4) $Desc_T(S)$ is in $Compl(T)$ if S is an arbitrary set of children of an R node of T.

(5) No other sets are in $Compl(T)$.

In (2)–(4) above S is a set of *siblings*, that is, nodes that share a common parent.

The collection $Compl(T)$ constructed from tree T has several important properties. First of all, it is complete, as its name already suggests. Then, $Compl(T)$ has the C1P if and only if T has no R nodes. Thus, a good way of knowing if a collection $\mathscr{C}$ has the C1P is to find a tree T with the property that $Compl(T) = \overline{\mathscr{C}}$, and then check whether this tree has R nodes. Such a tree will be called a *PQR tree for* $\mathscr{C}$.

All these properties will be shown in the rest of this section. Furthermore, we will indicate how to construct a PQR tree T for any collection $\mathscr{C}$. It turns out that for a fixed $\mathscr{C}$ all these trees are equivalent to each other. The section closes with a few extra results.

We now present the proofs of the results mentioned above.

Theorem 28. *The collection $Compl(T)$ is complete.*

Proof. The trivial sets are in *Compl*(T) by definition. We must therefore show that *Compl*(T) is closed under intersection, nondisjoint union, and noncontained complement.

It suffices to consider two sets $Desc_T(S)$ and $Desc_T(S')$ with S and S' being sets of children of the same internal node v. Otherwise the sets would be mutually orthogonal and the operations yield either one of the original sets or a trivial set. Now there are three choices for the type of v.

If v is a P node, $S=S'$ and we are done. If v is a Q node, the operation, be it intersection, nondisjoint union, or noncontained complement, yields $Desc_T(S'')$ for some set S'' of consecutive children of v. Finally, if v is an R node, anything goes, so again we obtain a set in *Compl*(T). □

The following result shows that we can view *Compl*(T) as the collection of all sets consecutive in every permutation compatible with T. However, it is only valid if T is a PQ tree, that is, has no R nodes.

Theorem 29. *If T has no R nodes, then*

$$Compl(T) = Consec(Compat(T)).$$

Proof. It is easy to see from the definitions that every set in *Compl*(T) is consecutive in T's frontier, so

$$Compl(T) \subseteq Consec(Frontier(T)).$$

It is also straightforward to see that equivalent trees yield the same complete collection, that is,

$$T \equiv T' \Rightarrow Compl(T) = Compl(T').$$

These two observations imply that

$$Compl(T) \subseteq Consec(Compat(T)).$$

Now for the other direction, let A be a set in *Consec*(*Compat*(T)). Mark all leaves in T that belong to A and then perform the following procedure:

```
while there is a node $v$ with all children marked then
unmark $v$'s children and mark $v$.
```

Let X be the set of marked nodes after this procedure. We can write

$$A = \bigcup_{v \in X} Desc_T(v).$$

If X is a singleton then A belongs to *Compl*(T) by definition. If X has at least two elements, then all elements of X must be siblings. In fact, let x and y be two nodes in X. Observe that if x is an ancestor of y one of them would not be marked, so there is no ancestry relationship between them. Let v be the lowest common ancestor of x and y. If there is a node z in the path from x to v, then z is not marked, and therefore there is a leaf a descending from z which does not belong to A. Regardless of the type of node z, there will be a permutation of its children that leaves a between the descendants of x and y in the frontier. This contradicts the fact that A is consecutive in the frontier of every tree equivalent to T. Therefore, there are no intermediate nodes between x and v, and an analogous argument holds for y and v as well.

It follows that X is a set of siblings. Let v be the common parent. If v is a P node, then X must include all children of v, otherwise we can find a permutation of these children yielding an equivalent tree in whose frontier A is not consecutive. If v is a Q node, then X must be formed by consecutive children by the same reason. In both cases A fits the description of a set in *Compl*(T). Because T has no R nodes, this completes our proof. □

As remarked, the consecutive ones property is related to the absence of R nodes. The following result makes this relationship more precise.

Theorem 30. *There is a valid permutation for collection Compl*(T) *if and only if* T *has no R nodes.*

Proof. Notice that if T has no R nodes, its frontier is a valid permutation for *Compl*(T).

On the other hand, if T has at least one R node no permutation is valid for *Compl*(T). Indeed, let α be any permutation and let v be an R node of T. Let x, y, and z be three distinct children of v (an R node has at least three children). Choose any leaf a descending from x, any leaf b descending from y, and any leaf c descending from z. Regardless of the order in which a, b, and c appear in α, one of the following three sets in *Compl*(T) will fail to be consecutive:

$$Desc_T(x) \cup Desc_T(y),$$

$$Desc_T(y) \cup Desc_T(z),$$

$$Desc_T(z) \cup Desc_T(x). \qquad \square$$

It is true that for any collection $\mathscr{C}$ there is a PQR tree T such that *Compl*$(T) = \overline{\mathscr{C}}$. However, we will postpone the proof of this result until the section containing an algorithm for this task. We proceed here showing how a tree compactly represents the valid permutations of a collection. We need a few preliminary definitions and results.

Theorem 31. *If* T *does not have R nodes, then*

$$Valid(Compl(T)) = Compat(T).$$

Proof. By induction on the number of internal nodes of T.

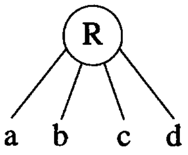


Fig. 4. A tree for a collection $\mathscr{C}$ such that $Valid(Compl(T)) \neq Compat(T)$.

If T has only one internal node, this node is necessarily the root. We have two possibilities: the root can be either a P node or a Q node. Recall that R nodes do not exist in T.

If T has a P node, then $Compl(T) = \mathscr{T}(U)$ and $Valid(Compl(T)) = Perm(U)$. On the other hand, $Compat(T) = Perm(U)$ as well, so we have equality in this case.

If T has a Q node, then $Compat(T) = \{\alpha, \overline{\alpha}\}$ for some permutation α, where $\overline{\alpha}$ is the reverse of α. But we also have $Compl(T) = Consec(\alpha)$ and it is not hard to see that $Valid(Compl(T)) = Compat(T)$.

Let us now proceed with the induction considering trees with more than one internal node. Let T be such a tree and take v an internal node of T whose children are all leaves. The set $H = Desc_T(v)$ satisfies the hypotheses of Theorem 22 and therefore we have

$$Valid(Compl(T)) = Valid(Compl(T)/H) \star Valid(Compl(T) \wedge H).$$

However, $Compl(T)/H = Compl(T_1)$, where T_1 is the tree obtained from T by replacing v by a single element called H. On the other hand $Compl(T) \wedge H = Compl(T_2)$, where T_2 is the subtree rooted at v. By induction hypothesis, we have

$$Valid(Compl(T_1)) = Compat(T_1)$$

and

$$Valid(Compl(T_2)) = Compat(T_2).$$

On the other hand, it is also true that

$$Compat(T) = Compat(T_1) \star Compat(T_2),$$

since T is the result of replacing the leaf H in T_1 by T_2. The result follows from these last three equalities. □

Example 32. Let $U = abcd$ and $\mathscr{C} = \{ab, bc, cd, ad\}$. The PQR tree T for $\mathscr{C}$ is shown in Fig. 4. We observe that the permutation $abcd$ belongs to $Compat(T)$ but is clearly not consecutive for $\mathscr{C}$, since $Valid(Compl(T)) = \emptyset$. This justifies the need for the restriction in Theorem 31.

We are now ready for one of our main results, which states that if a collection has the C1P, its valid permutations are described by a tree that satisfies some criteria. Later on, we will see that any collection admits a tree satisfying these criteria (Theorem 38).

Theorem 33. *If $\mathscr{C}$ has the C1P and the tree T is such that $Compl(T)=\overline{\mathscr{C}}$, then*

$$Valid(\mathscr{C}) = Compat(T).$$

Proof. By Theorems 15 and 31 we have

$$Valid(\mathscr{C}) = Valid(\overline{\mathscr{C}}) = Valid(Compl(T)) = Compat(T).$$

Observe that we can use Theorem 31 because Theorem 30 guarantees that T does not have R nodes. □

Example 34. To see that the restriction is necessary in the previous theorem, consider the collection in Example 32. It is clear that the permutation *abcd* belongs to $Compat(T)$ but does not belong to $Valid(\mathscr{C})$, which is empty in this case.

The remaining of this section is devoted to some interesting results that shed more light on the relationship among trees, collections, and permutations.

We start by characterizing the sets in $Compl(T) \cap Compl(T)^{\perp}$.

Lemma 35. *For any tree T and any node v of T we have*

$$Desc_T(v) \in Compl(T) \cap Compl(T)^{\perp}.$$

Conversely, every nonempty set in $Compl(T) \cap Compl(T)^{\perp}$ is of the form $Desc_T(v)$ for some node v of T.

Proof. Let us first show that $Desc_T(v) \in Compl(T)$. If v is a leaf, then $Desc_T(v)$ is trivial and we are done. If v is an internal node, we can take S as the set of all children of v, and, regardless of the type of v, $Desc_T(S) \in Compl(T)$ by definition. Note that $Desc_T(S) = Desc_T(v)$ in this case.

To prove that $Desc_T(v) \in Compl(T)^{\perp}$ we have to worry only about nontrivial sets in $Compl(T)$. Such sets are of the form $Desc_T(S)$, where S is a set of children of a certain node x. We have three cases to consider:

- v is an ancestor of x (includes the case $v=x$): then $Desc_T(v) \supseteq Desc_T(S)$.
- There is an ancestor of v in S: then $Desc_T(v) \subseteq Desc_T(S)$.
- remaining cases: $Desc_T(v) \cap Desc_T(S) = \emptyset$.

Conversely, suppose that a nonempty set $H \in Compl(T)$ also belongs to $Compl(T)^{\perp}$. The result is obvious if H is trivial (take the root or a leaf as v), so let us assume that H is not trivial. Then $H = Desc_T(S)$ for some set S of children of a node v, with $|S| \geqslant 2$. If v is a P node, S must include all its children by definition. If v is a Q node and S does not include all children of v, then there is another set S' of consecutive children of v such that $Desc_T(S) \not\perp Desc_T(S')$, a contradiction. Similarly, if v is an R node S must also contain all its children.

In all cases we have then $H = Desc_T(S) = Desc_T(v)$. □

Now we present some results involving equivalent trees.

Theorem 36. *Given two PQR trees T and T' over the same set U, we have*

$$Compl(T) = Compl(T') \Leftrightarrow T \equiv T'.$$

Proof. It is straightforward to see that if $T \equiv T'$ they yield the same complete collection. The rules for forming sets in $Compl(T)$ are invariant under equivalence transformations.

To prove that $Compl(T) = Compl(T')$ implies the equivalence between T and T' is more involved, and we will do it by induction on the number of elements of U, starting with the base case $|U| = 2$. In this case the trees must have a P node with two leaves as children, and they are clearly equivalent.

If $|U| \geqslant 3$, let us first treat the case where the collection $Compl(T)$ is prime. By Lemma 35 both trees T and T' have only one internal node – the root. Furthermore, Theorem 27 describes all possibilities for $Compl(T)$. It is easy to see that the roots of T and T' must be of the same type. If this type is P or R, the trees are obviously equivalent. If the roots are Q nodes, then to generate the same collection $Compl(T)$ we must necessarily have the children in T' in the same order they appear in T, or in the reverse order. Therefore, once again the trees are equivalent.

Now let us tackle the case where $Compl(T)$ is not prime. Let H be a nontrivial set in $Compl(T) \cap Compl(T)^{\perp}$. By Lemma 35 there are nodes v in T and v' in T' such that

$$H = Desc_T(v) = Desc_{T'}(v').$$

Let T/v be the PQR tree obtained from T by substituting the subtree rooted at v by a new leaf $H = Desc_T(v)$. Let also $T \wedge v$ be the subtree rooted at v. It is straightforward to verify that

$$Compl(T/v) = Compl(T)/Desc_T(v)$$

and

$$Compl(T \wedge v) = Compl(T) \wedge Desc_T(v).$$

Because $Compl(T) = Compl(T')$ and $Desc_T(v) = Desc_{T'}(v')$, we have

$$Compl(T/v) = Compl(T'/v')$$

and

$$Compl(T \wedge v) = Compl(T' \wedge v').$$

Now by the induction hypothesis we have the equivalences

$$T/v \equiv T'/v'$$

and

$$T \wedge v \equiv T' \wedge v'.$$

But this implies at once the equivalence between T and T'. $\square$

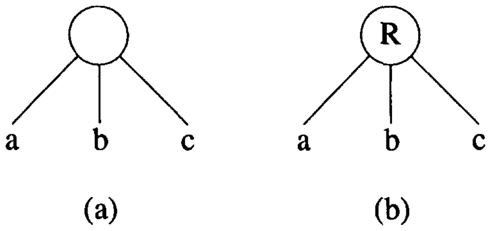


Fig. 5. (a) PQR tree rooted at a P node. (b) PQR tree rooted at an R node.

Theorem 37. *If T and T' are two trees without R nodes, then*

$$Compat(T) = Compat(T') \Leftrightarrow T \equiv T'.$$

Proof. The $\Leftarrow$ part follows from the definition. For the $\Rightarrow$ part, notice that by Theorem 29 we have

$$Compat(T) = Compat(T') \Rightarrow Compl(T) = Compl(T'),$$

because none of the trees has R nodes. The result then follows immediately from Theorem 36. □

The example in Fig. 5 shows that the hypothesis is necessary in the previous theorem.

6. Algorithm

The algorithm we are about to present is based on the following result, which tells us how to construct a PQR tree for a given collection $\mathscr{C}$.

Theorem 38. *For any collection $\mathscr{C}$ over a set U, there is a PQR tree T such that*

$$Compl(T) = \overline{\mathscr{C}}.$$

Proof. By induction on the number of elements of U. If $|U| = 2$, a tree consisting of just a root of type P and two children satisfies the requirements for any collection $\mathscr{C}$.

Now let us turn to the case $|U| \geqslant 3$. We investigate first the case where $\mathscr{C}$ is prime. In this case, Theorem 27 describes $\overline{\mathscr{C}}$, and in each case a tree T can be constructed such that $Compl(T) = \overline{\mathscr{C}}$ as follows.

In case (1) we have a trivial completion. Thus, T can be a P node with all elements of U as children. It is immediate that $Compl(T) = \mathscr{T}(U)$.

In case (2) only two permutations are valid, namely, α and its reverse. Take T as a Q node with all elements of U as children, in the order given by α. It is clear that $Compl(T) = Consec(\alpha)$.

Finally, in case (3) all sets are in the completion. Take T as an R node with all elements of U as children. By definition, we have $Compl(T) = Subsets(U)$.

Let us now treat the nonprime instances. Let H be a nontrivial set in $\overline{\mathscr{C}} \cap \mathscr{C}^{\perp}$. We construct subinstances $\mathscr{C}/H$ and $\mathscr{C} \wedge H$ as in Section 5. By induction hypothesis, there are trees T_1 and T_2 such that

$$Compl(T_1) = \overline{\mathscr{C}/H}$$

and

$$Compl(T_2) = \overline{\mathscr{C} \wedge H}.$$

Let $T_1[H/T_2]$ be the tree obtained from T_1 by replacing leaf H by T_2. We claim that this tree is a PQR tree for $\mathscr{C}$. To prove this claim, we need the following intermediate results:

$$Compl(T_1[H/T_2]) = (Compl(T_1) \bowtie H) \cup Compl(T_2)$$

and

$$\overline{\mathscr{C}} = (\overline{\mathscr{C}/H} \bowtie H) \cup \overline{\mathscr{C} \wedge H}. \tag{1}$$

The first result can be easily proven without difficulty from the definitions of $Compl(T)$ and the operations on sets defined in Section 5. The second result is also not difficult. We briefly sketch the proof here. We begin by observing that the right-hand collection is complete and contains $\mathscr{C}$; therefore it contains $\overline{\mathscr{C}}$. On the other hand, both $\mathscr{C}/H \bowtie H$ and $\mathscr{C} \wedge H$ are contained in $\mathscr{C}$, and this implies that their completions are contained in the completion of $\mathscr{C}$. Eq. (1) now follows easily because of the relation

$$\overline{\mathscr{C}/H \bowtie H} = \overline{\mathscr{C}/H} \bowtie H,$$

valid in this context. □

As a direct consequence of this result, we prove the following lemma.

Lemma 39. *If $Valid(\mathscr{C}) \neq \emptyset$, then $\overline{\mathscr{C}} = Consec(Valid(\mathscr{C}))$.*

Proof. By Theorem 38 there is a tree T with

$$\overline{\mathscr{C}} = Compl(T).$$

This tree has no R nodes (Theorem 30), hence we can write,

$$Compl(T) = Consec(Compat(T)) = Consec(Valid(\overline{\mathscr{C}})),$$

using Theorems 29 and 31, respectively. The conclusion follows form Theorem 15. □

Using the results presented so far and using the convention that any PQR tree containing an R node has the empty set as its set of valid permutations, we see that

```
**Function** $PQR\text{-}Tree(U,\mathscr{C})$
  **if** there is a nontrivial set $H$ in $\overline{\mathscr{C}} \cap \mathscr{C}^{\perp}$ **then**
    $T_1 \leftarrow PQR\text{-}Tree(U/H, \mathscr{C}/H)$
    $T_2 \leftarrow PQR\text{-}Tree(U \cap H, \mathscr{C} \wedge H)$
    **return** $T_1$ with leaf $H$ replaced by $T_2$
  **else**
    {$\mathscr{C}$ is prime }
    **case**
      $\mathscr{C}$ trivial:          **return** $P\text{-}node(U)$
      $\exists \alpha \in valid(\mathscr{C})$: **return** $Q\text{-}node(\alpha)$
      otherwise:          **return** $R\text{-}node(U)$
    **endcase**
  **endif**
```

Fig. 6. A recursive algorithm to construct PQR trees.

every instance $(U,\mathscr{C})$ 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 Fig. 6 for constructing a PQR tree for $(U,\mathscr{C})$.

The correctness of the algorithm follows from these results. We will now investigate its complexity, dividing it in some smaller steps. In what follows we clarify how to execute every step.

The steps that must be accomplished are:

(1) Find a nontrivial set $H \in \overline{\mathscr{C}} \cap \mathscr{C}^{\perp}$, or report that none exists.
(2) Decompose $\mathscr{C}$ into $\mathscr{C}/H$ and $\mathscr{C} \wedge H$.
(3) Join trees T_1 and T_2.
(4) Test whether a prime collection is trivial.
(5) Find a valid permutation for a nontrivial prime collection, or report that none exists.

Let us define a new graph related to these tasks. This is the *overlapping graph* defined by Fulkerson and Gross [4]. Associate with $\mathscr{C}$ a graph $G(\mathscr{C})$, in which each vertex denotes a set of $\mathscr{C}$ and an edge connects two vertices of $G(\mathscr{C})$ if the corresponding sets A and B of $\mathscr{C}$ have nonempty intersection and neither of them is included in the other, i.e., $A \not\perp B$. To make the reading easier, whenever we mention *sets in a component* we mean *vertices representing sets in a connected component.*

We have the following results concerning sets H in $\overline{\mathscr{C}} \cap \mathscr{C}^{\perp}$.

Theorem 40. *Let X be a connected component of $G(\mathscr{C})$. Then*

$$H = \bigcup_{A \in X} A$$

is a set in $\overline{\mathscr{C}} \cap \mathscr{C}^{\perp}$.

Proof. The set H can be obtained by nondisjoint union of (some) sets in X, hence it belongs to $\overline{\mathscr{C}}$. Notice also that H contains every set in X, so it is orthogonal to those. If a set $B \in \mathscr{C}$ outside X is not orthogonal to H, then it is not orthogonal to some $A \in X$, a contradiction. Therefore, $H \in \mathscr{C}^{\perp}$. □

The set H obtained as suggested by Theorem 40 is called a *union of component.*

Before we can proceed on our results, we must prove a property of $G(\mathscr{C})$. An auxiliary lemma precedes the result.

Lemma 41. *If A, B, and H are subsets of U with $A \subseteq H$, $B \perp H$, and $A \not\perp B$, then $B \subseteq H$.*

Proof. The fact that $B \perp H$ leaves three possibilities. If $H \subseteq B$ we would have $A \subseteq B$, which is not true. If $H \cap B = \emptyset$ then $A \cap B = \emptyset$ also, which is impossible. The only possibility left is $B \subseteq H$, which must then be true. □

Lemma 42. *If all the unions of components in $G(\mathscr{C})$ are trivial, then all nontrivial sets in $\mathscr{C}$ belong to the same connected component of $G(\mathscr{C})$.*

Proof. Suppose the conclusion is not true and let X and Y be two connected components of $G(\mathscr{C})$, with nontrivial sets $A_1,\dots,A_k$ and $B_1,\dots,B_l$, respectively. Since all the unions of connected components are trivial,

$$\bigcup_{1 \leqslant i \leqslant k} A_i = \bigcup_{1 \leqslant j \leqslant l} B_j = U.$$

As the components are disjoint and their unions are equal to U, at least one set of X is properly contained in one set of Y or one set of Y is properly contained in some set of X. Suppose, without loss of generality, that $A_1 \subset B_1$. Then, by Lemma 41, every set adjacent to A_1 must be also contained in B_1, otherwise there would be an edge between B_1 and some A_i. The same applies to the rest of the sets in the connected component. So, the entire component X would be included in B_1, but B_1 cannot be U. □

Building unions of components is one way of obtaining suitable sets for the decomposition. Another way is afforded by the following concept. Two elements a and b of U are called *twins* with respect to $\mathscr{C}$ if the binary set ab is in $\mathscr{C}^{\perp}$; notice that $ab \in \mathscr{C}^{\perp}$ if and only if no set $A \in \mathscr{C}$, $|A| \geqslant 2$, has $|\{a,b\} \cap A| = 1$. This is an equivalence relation.

Lemma 43. *If all unions of components in $G(\mathscr{C})$ are trivial, then the equivalence classes of the twins relation are sets in $\overline{\mathscr{C}} \cap \mathscr{C}^{\perp}$.*

Proof. As the twins relation is an equivalence relation, and every pair in an equivalence class H is contained in $\mathscr{C}^{\perp}$, we have $H \in \mathscr{C}^{\perp}$.

To see that $H \in \overline{\mathscr{C}}$, it suffices to consider H nontrivial. Let us divide the nontrivial sets in $\mathscr{C}$ into $A_1, A_2, \dots, A_k, B_1, B_2, \dots, B_l$ so that $H \subseteq A_i$, $1 \leqslant i \leqslant k$, and $H \cap B_i = \emptyset$, $1 \leqslant i \leqslant l$.

Notice that no set in $\mathscr{C}$ is properly contained in or strictly overlaps H. Because H is nontrivial we know there are nontrivial sets in $\mathscr{C}$; by Lemma 42 they are all in one component of $G(\mathscr{C})$, whose union must be U. Therefore, $k \geqslant 1$, that is, there is at least one set $A_1 \in \mathscr{C}$ that contains H.

Let us suppose, also, that the sets $B_1, B_2, \dots, B_l$ were ordered by a BFS applied in $G(\mathscr{C})$ starting in A_1. We proceed to show by induction on j that, for $0 \leqslant j \leqslant l$,

$$H_j = (\cdots(A_1 \cap A_2 \cap \cdots \cap A_k)\backslash B_1)\backslash B_2), \backslash \cdots)\backslash B_j) \in \overline{\mathscr{C}}.$$

For a base case, let $j=0$. Then $H_0 = (A_1 \cap A_2 \cap \cdots \cap A_k) \in \overline{\mathscr{C}}$. For $j>0$, suppose that

$$(\cdots(A_1 \cap A_2 \cap \cdots \cap A_k)\backslash B_1)\backslash B_2), \backslash \cdots)\backslash B_{j-1}) \in \overline{\mathscr{C}}.$$

Since B_j is nontrivial, it belongs to the same connected component as the other sets, and there is a previous A_i or B_i, $i<j$, adjacent to it. Let us consider the two possible cases:

(1) If $B_j \not\perp A_i$ then $B_j \not\subseteq A_i$, and hence

$$B_j \not\subseteq (\cdots(A_1 \cap A_2 \cap \cdots \cap A_k)\backslash B_1)\backslash B_2), \backslash \cdots)\backslash B_{j-1}),$$

which implies that

$$(\cdots(A_1 \cap A_2 \cap \cdots \cap A_k)\backslash B_1)\backslash B_2), \backslash \cdots)\backslash B_j) \in \overline{\mathscr{C}}$$

by noncontained difference.

(2) If $B_j \not\perp B_i$ then $B_j \cap B_i \neq \emptyset$, and hence

$$B_j \not\subseteq (\cdots(A_1 \cap A_2 \cap \cdots \cap A_k)\backslash B_1)\backslash B_2), \backslash \cdots)\backslash B_{j-1}),$$

which implies that

$$(\cdots(A_1 \cap A_2 \cap \cdots \cap A_k)\backslash B_1)\backslash B_2), \backslash \cdots)\backslash B_j) \in \overline{\mathscr{C}}$$

also by noncontained difference.

The result follows observing that $H = H_l$. □

The following example shows why the restrictions in Lemma 43 are necessary.

Example 44. Let $U = abcde$ and $\mathscr{C} = \{cde\}$. The set ab is a twin class but it does not belong to $\overline{\mathscr{C}}$. But the component of $G(\mathscr{C})$ consisting of cde alone has a nontrivial union.

Definition 45. A *partition* of a set U with respect to $\mathscr{C}$ is the division of U into nonempty pairwise disjoint subsets (blocks) such that the union of all of them is U and every set in $\mathscr{C}$ can be constructed by doing the union of some blocks.

We say that a partition A of U with respect to $\mathscr{C}$ refines another partition B with respect to the same collection if every block of A is contained in some block of B.

The coarsest partition of U with respect to $\mathscr{C}$ is a partition that does not refine any other.

The next lemma proves the equivalence between the equivalence classes of the twins relation for $\mathscr{C}$ and the nontrivial blocks of the coarsest partition of U with respect to $\mathscr{C}$. *This claim is true only if no singleton is present in* $\mathscr{C}$, as we can easily realize, so we must remove the singletons before applying a partition algorithm. This operation does not affect *Valid*($\mathscr{C}$).

Lemma 46. *Let $\mathscr{C}$ be a collection with no singletons and such that all unions of components in $G(\mathscr{C})$ are trivial. Then the nontrivial blocks of the coarsest partition of U with respect to $\mathscr{C}$ are exactly the equivalence classes for the twins relation.*

Proof. Since the equivalence classes of the twins relation are in $\overline{\mathscr{C}} \cap \mathscr{C}^{\perp}$, its easy to see that each class is completely contained in a single block of the coarsest partition of U with respect to $\mathscr{C}$.

Conversely, let H be a block of the coarsest partition of U with respect to $\mathscr{C}$ and suppose that H is not contained in any equivalence class of the twins relation. Then its elements are split in at least two disjoint blocks A and B. Then there must exist $S \in \mathscr{C}$ such that either $A \subseteq S$ and $B \nsubseteq S$ or $A \nsubseteq S$ and $B \subseteq S$. So S refines H which would not be a block of the coarsest partition anymore. □

The following result shows that no other ways of obtaining sets in $\overline{\mathscr{C}} \cap \mathscr{C}^{\perp}$ are necessary.

Theorem 47. *If $\mathscr{C}$ is a collection such that*
- *all unions of components in $G(\mathscr{C})$ are trivial,*
- *$\mathscr{C}$ has no twins,*

then $\mathscr{C}$ is prime.

Proof. The result is obvious if $\mathscr{C} \subseteq \mathscr{T}(U)$, so we assume $\mathscr{C}$ has at least one nontrivial set. By Lemma 42, all nontrivial sets in $\mathscr{C}$ are part of the same connected component of $G(\mathscr{C})$, and the union of this component equals U. In other words, for every $a \in U$ there is a nontrivial set $A \in \mathscr{C}$ such that $a \in A$.

We will prove the result by contradiction. Suppose that H is a nontrivial set in $\overline{\mathscr{C}} \cap \mathscr{C}^{\perp}$. Because H is nontrivial, there are at least two distinct elements a and b in H, and at least one element c not in H. By hypothesis a and b are not twins, hence there is a set $A \in \mathscr{C}$ that separates a and b, that is, $ab \not\perp A$. But $A \perp H$, therefore $A \subseteq H$.

On the other hand, there is a nontrivial set $B \in \mathscr{C}$ such that $c \in B$. Because both A and B correspond to vertices in $G(\mathscr{C})$, and $G(\mathscr{C})$ is connected, there is a path

$$A \not\perp A_1 \not\perp \cdots A_k \not\perp B$$

in $G(\mathscr{C})$ leading from A to B. By Lemma 41, $A_1 \subseteq H$. Repeated application of this lemma eventually results in $B \subseteq H$, but this contradicts the fact that $c \in B \backslash H$. □

Now we consider the cost of building the PQR tree. Based on the previous results, we can subdivide step (1) into:

(1) find a nontrivial set $H \in \overline{\mathscr{C}} \cap \mathscr{C}^{\perp}$, or report that none exists
  (a) Find unions of components in $G(\mathscr{C})$.
  (b) If all are trivial, find twin classes.
  (c) If all twin classes are trivial as well, report that $\mathscr{C}$ is prime.

Steps (2)–(4) are easily accomplished in linear time. The more intricate steps are (1) and (5). In step (1), steps (1b) and (1c) are simple also, and can be accomplished in linear time (singletons can be removed easily, twin classes can be found by coarsest partitioning, which is a process similar to lexBFS [12, 7]). Step (1c) can be acomplished in linear time provided that a spanning tree for the component is given [7, Section 2].

The crucial step is then (1a) finding unions of components. We currently do not know how to implement this step in linear time, except by resorting to previous linear algorithms for the C1P in the literature. With this, each invocation of the PQR tree routine will cost linear time, and the overall running time will be quadratic because each time the instance is broken into two others whose combined size is bounded by the size of the original instance plus a constant:

Lemma 48. *For any collection $\mathscr{C}$ and any set $H \in \overline{\mathscr{C}} \cap \mathscr{C}^{\perp}$ we have*

$$size(U/H, \mathscr{C}/H) + size(H, \mathscr{C} \wedge H) \leqslant size(U, \mathscr{C}) + 1.$$

What is the point of having a new algorithm that is slower than previous ones and still needs these as subroutines? One reason is that the algorithm is much clearer than the previous ones, and if we could find simpler algorithms to implement step (1a), it may very well be that in some cases people would rather implement a simpler algorithm than deal with intricate code.

7. Robustness

As a final remark, we provide examples to illustrate how a PQR tree can behave in the presence of errors.

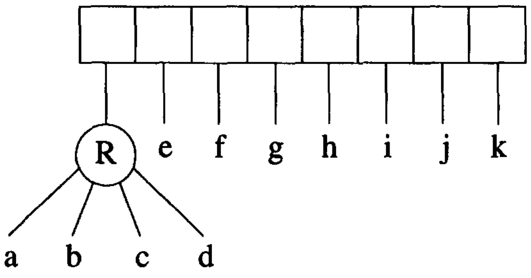


Fig. 7. PQR tree for collection $\mathscr{C}$.

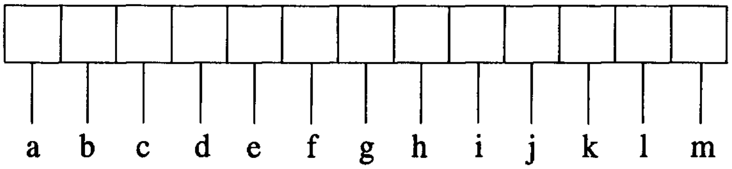


Fig. 8. What the PQR tree for collection $\mathscr{C}$ would be without errors.

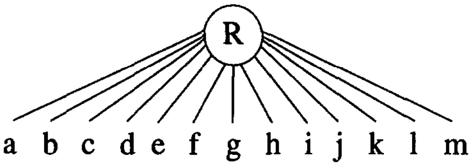


Fig. 9. PQR tree for collection $\mathscr{C}$ with an error.

Example 49. Let us suppose we made an hybridization experiment, and we obtained the following collection:

$$U = abcdefghij,$$
$$\mathscr{C} = \{ab, bc, cd, bd, ac, abcde, efgh, ef, ghijk, fg, ghi, ij, jk\}.$$

The tree for this collection appears in Fig. 7. As we can see, it has an R node, so the collection does not have the C1P. But this R node tells us that an error may had occurred in the experiments involving only a, b, c and d, a small subset of the probes.

Example 50. Suppose now we were to obtain the tree in Fig. 8, which looks like a good answer, after performing the experiments that resulted in the family below.

$$U = abcdefghijklm,$$
$$\mathscr{C} = \{abc, ab, bcde, defg, de, ef, fg, ghij, hijkl, ij, jk, kl, lm\}.$$

But let us consider that an error included the set al in $\mathscr{C}$. The resulting tree would be the one in Fig. 9, a tree that provides no help in finding out the experimental error. Other analysis methods are needed.

In both examples, a PQ tree would not be constructed and no information could be retrieved from the experiments.

8. Conclusion

This paper contributes to a better understanding of the Consecutive Ones Property in several ways, following the steps of Meidanis and Munuera [9].

First, we develop a new theory, that recasts the property in terms of collections of sets. Then, the class of PQ trees of Booth and Lueker [2] is extended to include PQR trees. The new trees exist for any collection, not only collections that have the C1P.

New proofs of the fact that a PQ tree is capable of recording all valid permutations of a collection are given. In addition, we prove that for every PQ tree there is a collection of sets for which this tree records the valid permutations.

A new, simpler, algorithm for building a PQR tree for a given collection is given. The algorithm runs in polynomial time, but it depends on previous algorithms, and it would be interesting to find a linear time algorithm based on the ideas developed here. The linear time algorithms in the literature are hard to implement, and do not construct R nodes.

Acknowledgements

We thank Erasmo G. Munuera, who helped develop some of the ideas presented here. We also acknowledge the excellent work of the referees, who provided us with very helpful and detailed comments.

References

[1] S. Benzer, On the topology of genetic fine structure, Proc. Natl. Acad. Sci. USA 45 (1959) 1607–1620.

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

[3] A.G. Ferreira, S.W. Song, Achieving optimality for gate matrix layout and PLA folding: a graph theoretic approach, in: I. Simon (Ed), Latin'92, Lecture Notes in Computer Science, vol. 583, São Paulo, Brasil, Springer, Berlin, 1992, pp. 139–153.

[4] D.R. Fulkerson, O.A. Gross, Incidence matrices and interval graphs, Pacific J. Math. 15 (3) (1965) 835–855.

[5] S.P. Ghosh, File organization: the consecutive retrieval property, Commun. ACM 15 (9) (1972) 802–808.

[6] D.S. Greenberg, S. Istrail, Physical mapping by STS hybridization: algorithmic strategies and the challenge of software evaluation, J. Comput. Biol. 2 (2) (1995) 219–274.

[7] W.L. Hsu, A simple test for the consecutive ones property, in: Ibaraki, T., Inagaki, Y., Iwama, K., et al. (Eds.), ISAAC'92, Lecture Notes in Computer Science, vol. 650, Nagoya, Japan, Springer, Berlin, 1992, pp. 459–468.

[8] D.G. Kendall, Incidence matrices, interval graphs and seriation in archaeology, Pacific J. Math. 2 (28) (1969) 265–570.

[9] J. Meidanis, E.G. Munuera, A theory for the consecutive ones property, in: Ziviani, N., Ricardo Baeza-Yates, Guimarães, K. (Eds.), Proc. III South American Workshop on String Processing, Recife, Brazil, 1996, pp. 194–202.

[10] M.B. Novick, Generalized PQ-trees, Technical Report 89-1074, Cornell University, 1989.

[11] W.S. Robinson, A method for chronologically ordering archaeological deposits, Amer. Antiquity 16 (1951) 293–301.

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

[13] A.C. Tucker, A structure theorem for the consecutive 1's property, J. Combin. Theory 12 (B) (1972) 153–162.