Tree Path Labeling of Path Hypergraphs – A Generalization of the Consecutive Ones Property

N. S. Narayanaswamy[1] and Anju Srinivasan[2]

Indian Institute of Technology Madras, Chennai - 600036.
[1]swamy@cse.iitm.ernet.in, [2]asz@cse.iitm.ac.in

Abstract. We consider a natural generalization of the *consecutive ones property* (COP) on binary matrices. Given a set system $\mathcal{F} \subseteq (2^U \setminus \emptyset)$ of a finite set U of cardinality n and a tree T of size n, does there exist does there exist at least one bijection $\phi : U \to V(T)$ such that for each $S \in \mathcal{F}$, the set $\{\phi(x) \mid x \in S\}$ is the vertex set of a path in T? Our main result is that the existence of such a bijection from U to $V(T)$ is equivalent to the existence of a *tree path labeling* ℓ of $\mathcal{F}$ such that for any *three*, not necessarily distinct, $S_1, S_2, S_3 \in \mathcal{F}$, $|S_1 \cap S_2 \cap S_3| = |\ell(S_1) \cap \ell(S_2) \cap \ell(S_3)|$. Informally, a tree path labeling is a function from $\mathcal{F}$ to the set of all paths in T. Finally, if T is P_n, the path on n-vertices, then we get a reformulation of the problem of testing for COP. We conclude with a polynomial time algorithm to test for a *feasible* tree path labeling (formally defined later) of a restricted kind of set system, and where T is a special type of a tree called a *k-subdivided star*.

Keywords: consecutive ones property, algorithmic graph theory, hypergraph isomorphism, interval labeling

1 Introduction

Consecutive ones property (COP) of binary matrices is a widely studied combinatorial problem. The first mention of COP, according to D.G. Kendall [Ken69], was made by Petrie, an archaeologist, in 1899. The problem is to rearrange rows (columns) of a binary matrix in such a way that every column (row) has its 1s occur consecutively. If this is possible the matrix is said to have the COP. This problem has several practical applications in diverse fields including scheduling [HL06], information retrieval [Kou77] and computational biology [ABH98]. Further, it is a tool in graph theory [Gol04] for interval graph recognition, characterization of Hamiltonian graphs, and in integer linear programming [HT02,HL06].

The recognition of COP is polynomial time solvable by a rich class of algorithms. Some heuristics were proposed for testing the COP in [Rob51] before the work of Fulkerson and Gross [FG65] who presented the first polynomial time algorithm. Subsequently Tucker [Tuc72] presented a characterization of matrices with the COP based on certain forbidden matrix configurations. Booth and Leuker's PQ trees [BL76], PC trees invented by Hsu [Hsu01,Hsu02], and PQR trees by Meidanis and Munuera [MM96], are data structures to encode the class of permutations that realize COP for a given binary matrix. Specifically PQ trees also find application in other algorithms like planarity testing algorithms. PC-trees were used to test if a binary matrix has the CiRcular Ones Property (CROP). PQR tree were designed to provide a witness, via the R nodes, when the input binary matrix did not have the COP. McConnell in [McC04] transformed the problem for testing for COP to the problem of testing if a transformed graph was bipartite, thus providing a witness for the absence of COP. In a more recent work, the second author and Subashini [NS09] gave a novel characterization of matrices with the COP. They showed that a matrix has the COP if and only if an associated set system $\mathcal{F}$ on a universe U of cardinality n has a path labeling ℓ of sets in $\mathcal{F}$

from paths in P_n such that for any two $S_1, S_2 \in \mathcal{F}$, $|S_1 \cap S_2| = |\ell(S_1) \cap \ell(S_2)|$. Such path labelings were christened Intersection Cardinality Preserving Interval Assignments (ICPIA). The universe U in this characterization was the set of column indices in a given binary matrix. Each element of the set system $\mathcal{F}$ corresponded to a row in the matrix, and the associated set was the set of column indices with a 1. Clearly, if the binary matrix has the COP witnessed by a permutation ϕ of the column indices, then each set S in $\mathcal{F}$ has a natural associated interval, denoted by $\ell(S)$, and clearly $\ell(S) = \{\phi(x)|x \in S\}$. It is also clear that for any two $S_1, S_2 \in \mathcal{F}$, $|S_1 \cap S_2| = |\ell(S_1) \cap \ell(S_2)|$. Also, ϕ was viewed as a permutation from U to $V(P_n) = \{1, 2, \ldots, n\}$, where $n = |U|$, and P_n is the path on n vertices. Further, $\ell(S)$ turns out to be the vertex set of a path in P_n. This natural necessary condition was shown to be sufficient in [NS09]. In other words, they constructively obtained a permutation witnessing the COP from a ICPIA ℓ for $\mathcal{F}$. Finally, an ICPIA was constructed by adapting the prime matrix decomposition ideas of Hsu [Hsu02]. While the consecutive ones testing algorithm in [NS09] is same as the one presented in [Hsu02], the ICPIA provides a natural way of generalizing the COP of binary matrices.

Our Result. The generalization of the matrix COP testing problem is defined as follows – given a set system $\mathcal{F}$ from a universe U and a tree T, does there exist a bijection ϕ from U to the vertices of T such that for each $S \in \mathcal{F}$, the set $\{\phi(x)|x \in S\}$ is the vertex set of a path in T? Our result is that such a bijection ϕ exists if and only if there is a tree path labeling ℓ of the elements of $\mathcal{F}$ such that for any *three*, not necessarily distinct, $S_1, S_2, S_3 \in \mathcal{F}$, $|S_1 \cap S_2 \cap S_3| = |\ell(S_1) \cap \ell(S_2) \cap \ell(S_3)|$. We refer to such a path labeling as an Intersection Cardinality Preserving Path Labeling (ICPPL). Clearly, given a bijection ϕ such that $\{\phi(x)|x \in S\}$ is the vertex set of a path in T, we get an ICPPL ℓ in which for each $S \in \mathcal{F}$, $\ell(S) = \{\phi(x)|x \in S\}$. Therefore, the challenging question is the following which is what is mainly addressed in this paper: Given a path labeling ℓ of $\mathcal{F}$ from a tree T, does there exist a bijection $\phi : U \to V(T)$ such that $\ell(S) = \{\phi(x)|x \in S\}$? A given path labeling is referred to as a *feasible tree path labeling* if such a ϕ exists. We refer to such a *phi* as a hypergraph isomorphism from U to $V(T)$. We show that a given tree path labeling is feasible if and only if it is an ICPPL. This characterization is proved constructively by obtaining a relevant bijection ϕ from a given ICPPL in Section 2. This generalizes the characterization of matrices that have the COP in [NS09]. We then consider the complexity of finding an ICPPL for a given family $\mathcal{F}$ in a given tree T in Section 3. In the case when T is a path, this is exactly the problem of testing for a COP that is well-known to be polynomial time solvable, also known to be in Deterministic Logspace [KKLV10]. We consider the case of a k-subdivided star which is a special kind of a tree. By extending the algorithmic technique of testing for an ICPPL in a path, we present a polynomial time algorithm to test for the existence of an ICPPL in a k-subdivided star for given family $\mathcal{F}$ which is restricted in two ways (see Section 3). The complexity of finding an ICPPL for arbitrary trees is open to the best of our knowledge, and we feel that it could have the same flavor of Graph Isomorphism. Among many other things we do not know of a co-NP algorithm for deciding the existence of an ICPPL. For the sake of clarity, we refer to the ordered pair $(\mathcal{F}, \ell)$ as a path labeling . Also, we use the word labeling, as each set in $\mathcal{F}$ is assigned a path label by ℓ. Finally, we assume that the support of $\mathcal{F}$ has the same cardinality as the support of ℓ. Mainly, this leaves out uninteresting situations were some element of U do not occur in the support of $\mathcal{F}$, and some elements of $V(T)$ do not occur in the support of ℓ.

2 Characterization of Feasible Tree Path Labelings

Consider a tree path labeling $(\mathcal{F}, \ell)$ on the given tree T. We call $(\mathcal{F}, \ell)$ an *Intersection Cardinality Preserving Path Labeling (ICPPL)* if it has the following properties.

(Property i)	$\lvert S\rvert = \lvert\ell(S)\rvert$	for all $S \in \mathcal{F}$
(Property ii)	$\lvert S_1 \cap S_2\rvert = \lvert\ell(S_1) \cap \ell(S_2)\rvert$	for all distinct $S_1, S_2 \in \mathcal{F}$
(Property iii)	$\lvert S_1 \cap S_2 \cap S_3\rvert = \lvert\ell(S_1) \cap \ell(S_2) \cap \ell(S_3)\rvert$	for all distinct $S_1, S_2, S_3 \in \mathcal{F}$

If T is a path, and if only conditions i and ii are satisfied, then ℓ is referred to an intersection cardinality preserving interval assignment (ICPIA). The following two lemmas are proved in [NS09]. The first one shows that the preservation of pairwise intersection cardinality in an interval assignment is sufficient to preserve three way intersection cardinality.

Lemma 2.1. *Let P be a path, S_1, S_2, S_3 be 3 sets, and T_1, T_2, T_3 be paths from P, such that $\lvert S_i \cap S_j\rvert = \lvert T_i \cap T_j\rvert, 1 \leq i, j \leq 3$. Then, $\lvert S_1 \cap S_2 \cap S_3\rvert = \lvert T_1 \cap T_2 \cap T_3\rvert$.*

The following lemma shows that if a path labeling that preserves pairwise intersection cardinalities is feasible.

Lemma 2.2. *A path labeling $(\mathcal{F}, \ell)$ on a path is feasible iff it is an ICPIA.*

The following lemma is useful in subsequent arguments.

Lemma 2.3. *If ℓ is an ICPPL, and $S_1, S_2, S_3 \in \mathcal{F}$, then $\lvert S_1 \cap (S_2 \setminus S_3)\rvert = \lvert\ell(S_1) \cap (\ell(S_2) \setminus \ell(S_3))\rvert$.*

Proof. Let $P_i = \ell(S_i)$, for all $1 \leq i \leq 3$. $\lvert S_1 \cap (S_2 \setminus S_3)\rvert = \lvert(S_1 \cap S_2) \setminus S_3\rvert = \lvert S_1 \cap S_2\rvert - \lvert S_1 \cap S_2 \cap S_3\rvert$. Due to properties (ii) and (iii) of ICPPL, $\lvert S_1 \cap S_2\rvert - \lvert S_1 \cap S_2 \cap S_3\rvert = \lvert P_1 \cap P_2\rvert - \lvert P_1 \cap P_2 \cap P_3\rvert = \lvert(P_1 \cap P_2) \setminus P_3\rvert = \lvert P_1 \cap (P_2 \setminus P_3)\rvert$. Thus lemma is proved. □

Lemma 2.4. *Let ℓ be an ICPPL, let $\ell(S_i) = P_i, 1 \leq i \leq 4$. Then, $\lvert \cap_{i=1}^{4} S_i\rvert = \lvert \cap_{i=1}^{4} P_i\rvert$.*

Proof. Consider the sets $S_2 \cap S_1$, $S_3 \cap S_1$, and $S_4 \cap S_1$, and their images $P_2 \cap P_1$, $P_3 \cap P_1$, and $P_4 \cap P_1$ respectively. It is clear that this path labeling preserves pairwise intersection cardinalities. In this case, it is clear that the 3 sets are assigned paths in the path P_1. Now by applying lemma 2.1, it follows that $\lvert \cap_{i=1}^{4} S_i\rvert = \lvert \cap_{i=1}^{4} P_i\rvert$.

Corollary 2.5. *Let P_1, P_2, P_3, P_4 be paths such that $P_1 \cap P_2, P_1 \setminus P_2, P_2 \setminus P_1$ are paths. Then the intersection cardinalities of $(P_1 \setminus P_2) \cap P_3 \cap P_4$, and $(P_2 \setminus P_1) \cap P_3 \cap P_4$ are preserved.*

Proof. It is clear that $\lvert(P_1 \setminus P_2) \cap P_3 \cap P_4\rvert = \lvert \cap_{i=1}^{4} P_i\rvert - \lvert P_2 \cap P_3 \cap P_4\rvert$. Since we have an ICPPL and from lemma 2.4, it follows that the intersection cardinalities are preserved.

Lemma 2.6. *Let ℓ be an ICPPL, and let $\ell(S) = P$. Let S_{priv} and P_{priv} be the elements of S and P that occur in no set other than S and P respectively. The $\lvert S_{priv}\rvert = \lvert P_{priv}\rvert$.*

Proof. Let us consider P and the set of paths $\{P_i \cap P\}$ and the sets $\{S_i \cap S\}$. Since ℓ is an ICPPL, it follows that the mapping of $S_i \cap S$ to $P_i \cap P$ is an ICPIA in P. Let S_{two} and P_{two} be the elements of S and P which are in one *other* set and path, respectively. From lemma 2.2 it follows that the ICPIA is feasible, that is there is a bijection ϕ from S_{two} to P_{two} such that for each i, the image of $S_i \cap S$ under ϕ is $P_i \cap P$. Most importantly, for this lemma, it follows that $\lvert S_{two}\rvert = \lvert P_{two}\rvert$. This shows that $\lvert S_{priv}\rvert = \lvert S \setminus S_{two}\rvert = \lvert P \setminus P_{two}\rvert = \lvert P_{priv}\rvert$. Hence the lemma.

We now constructively show that $(\mathcal{F}, \ell)$ is feasible if and only if it is an ICPPL. Algorithm 3 recursively does two levels of filtering of $(\mathcal{F}, \ell)$ to make it simpler while retaining the set of isomorphisms. $(\mathcal{F}, \ell)$ is made a simpler ICPPL by iteratively applying Algorithm 1 and Algorithms 2 one after the other. They are named `filter_common_leaf` and `filter_fix_leaf`. As a result of Algorithm 1

Algorithm 1 Refine ICPPL `filter_common_leaf`$(\mathcal{F}, \ell, T)$

$\mathcal{F}_0 \leftarrow \mathcal{F}$, $\ell_0(S) \leftarrow \ell(S)$ for all $S \in \mathcal{F}_0$
$j \leftarrow 1$
while there is $S_1, S_2 \in \mathcal{F}_{j-1}$ such that $\ell_{j-1}(S_1)$ and $\ell_{j-1}(S_2)$ have a common leaf in T **do**
  $\mathcal{F}_j \leftarrow (\mathcal{F}_{j-1} \setminus \{S_1, S_2\}) \cup \{S_1 \cap S_2, S_1 \setminus S_2, S_2 \setminus S_1\}$ | `Remove` S_1, S_2 `and add the ''filtered'' sets`
  for every $S \in \mathcal{F}_{j-1}$ s.t. $S \neq S_1$ and $S \neq S_2$ **do** $\ell_j(S) \leftarrow \ell_{j-1}(S)$ **end for**
  $\ell_j(S_1 \cap S_2) \leftarrow \ell_{j-1}(S_1) \cap \ell_{j-1}(S_2)$ | `Carry forward the path labeling for all existing sets other than` S_1, S_2
  $\ell_j(S_1 \setminus S_2) \leftarrow \ell_{j-1}(S_1) \setminus \ell_{j-1}(S_2)$ | `Define path labeling for new sets`
  $\ell_j(S_2 \setminus S_1) \leftarrow \ell_{j-1}(S_2) \setminus \ell_{j-1}(S_1)$
  $j \leftarrow j + 1$
end while
$\mathcal{F}' \leftarrow \mathcal{F}_j$, $\ell' \leftarrow \ell_j$
return $(\mathcal{F}', \ell')$

each leaf v in T is such that there is exactly one set in $\mathcal{F}$ with v as a vertex in the path assigned to it. In Algorithm 2 we identify elements in $supp(\mathcal{F})$ whose images are leaves in a hypergraph isomorphism. Let $S \in \mathcal{F}$ be such that $\ell(S)$ is a path with leaf and $v \in V(T)$ is the unique leaf incident on it. We define a new path labeling ℓ_{new} such that $\ell_{new}(\{x\}) = \{v\}$ where x an arbitrary element from $S \setminus \bigcup_{\hat{S} \neq S} \hat{S}$. In other words, x is an element present in no other set in $\mathcal{F}$ except S. This is intuitive since v is present in no other path image under ℓ other than $\ell(S)$. The element x and leaf v are then removed from the set S and path $\ell(S)$ respectively. After doing this for all leaves in T, all path images in the new path labeling ℓ_{new} except leaf labels (a path that has only a leaf is called the *leaf label* for the corresponding single element hyperedge or set) are paths from a new pruned tree $T_0 = T \setminus \{v \mid v \text{ is a leaf in } T\}$. Algorithm 2 is now presented with details.

Algorithm 2 Leaf labeling from an ICPPL `filter_fix_leaf`$(\mathcal{F}, \ell, T)$

$\mathcal{F}_0 \leftarrow \mathcal{F}$, $\ell_0(S) \leftarrow \ell(S)$ for all $S \in \mathcal{F}_0$ | `Path images are such that no two path images share a leaf.`
$j \leftarrow 1$
while there is a leaf v in T and a unique $S_1 \in \mathcal{F}_{j-1}$ such that $v \in \ell_{j-1}(S_1)$ **do**
  $\mathcal{F}_j \leftarrow \mathcal{F}_{j-1} \setminus \{S_1\}$
  for all $S \in \mathcal{F}_{j-1}$ such that $S \neq S_1$ set $\ell_j(S) \leftarrow \ell_{j-1}(S)$
  $X \leftarrow S_1 \setminus \bigcup_{S \in \mathcal{F}_{j-1}, S \neq S_1} S$
  $x \leftarrow$ arbitrary element from X
  $\mathcal{F}_j \leftarrow \mathcal{F}_j \cup \{\{x\}, S_1 \setminus \{x\}\}$
  $\ell_j(\{x\}) \leftarrow \{v\}$
  $\ell_j(S_1 \setminus \{x\}) \leftarrow \ell_{j-1}(S_1) \setminus \{v\}$
  $j \leftarrow j + 1$
end while
$\mathcal{F}' \leftarrow \mathcal{F}_j$, $\ell' \leftarrow \ell_j$
return $(\mathcal{F}', \ell')$

Algorithm 3 computes a hypergraph isomorphism ϕ recursively using Algorithm 1 and Algorithm 2 and pruning the leaves of the input tree. In brief, it is done as follows. Algorithm 2 gives us the leaf labels in $\mathcal{F}_2$, i.e., the elements in $supp(\mathcal{F})$ that map to leaves in T, where $(\mathcal{F}_2, \ell_2)$ is the output of Algorithm 2. All leaves in T are then pruned away. The leaf labels are removed from the path labeling ℓ_2 and the corresponding elements are removed from the corresponding sets in $\mathcal{F}_2$. This tree pruning algorithm is recursively called on the altered hypergraph $\mathcal{F}'$, path label ℓ' and tree T'. The recursive call returns the bijection ϕ'' for the rest of the elements in $supp(\mathcal{F})$ which along with the leaf labels ϕ' gives us the hypergraph isomorphism ϕ.

Algorithm 3 `get-hypergraph-isomorphism`$(\mathcal{F}, \ell, T)$

if T is empty **then**
  return $\emptyset$
end if
$L \leftarrow \{v \mid v \text{ is a leaf in } T\}$
$(\mathcal{F}_1, \ell_1) \leftarrow$ `filter_common_leaf`$(\mathcal{F}, \ell, T)$
$(\mathcal{F}_2, \ell_2) \leftarrow$ `filter_fix_leaf`$(\mathcal{F}_1, \ell_1, T)$
$(\mathcal{F}', \ell') \leftarrow (\mathcal{F}_2, \ell_2)$
$\phi' \leftarrow \emptyset$
for every $v \in L$ **do**
  $\phi'(x) \leftarrow v$ where $x \in \ell_2^{-1}(\{v\})$
  Remove $\{x\}$ and $\{v\}$ from $\mathcal{F}'$, ℓ' appropriately
end for
$T' \leftarrow T \setminus L$
$\phi'' \leftarrow$ `get-hypergraph-isomorphism`$(\mathcal{F}', \ell', T')$
$\phi \leftarrow \phi'' \cup \phi'$
return ϕ

`Copy the leaf labels to a one to one function` $\phi' : supp(\mathcal{F}) \to L$

Lemma 2.7. *In Algorithm 1, at the end of jth iteration, $j \geq 0$, of the* **while** *loop, the following invariants are maintained.*

I	$\ell_j(R)$ *is a path in* T,	*for all* $R \in \mathcal{F}_j$
II	$\lvert R\rvert = \lvert\ell_j(R)\rvert$,	*for all* $R \in \mathcal{F}_j$
III	$\lvert R \cap R'\rvert = \lvert\ell_j(R) \cap \ell_j(R')\rvert$,	*for all* $R, R' \in \mathcal{F}_j$
IV	$\lvert R \cap R' \cap R''\rvert = \lvert\ell_j(R) \cap \ell_j(R') \cap \ell_j(R'')\rvert$,	*for all* $R, R', R'' \in \mathcal{F}_j$

Proof. Proof is by induction on the number of iterations, j. In this proof, the term "new sets" will refer to the sets added to $\mathcal{F}_j$ in jth iteration in line 4 of Algorithm 1, $S_1 \cap S_2, S_1 \setminus S_2, S_2 \setminus S_1$ and its images in ℓ_j where $\ell_{j-1}(S_1)$ and $\ell_{j-1}(S_2)$ intersect and share a leaf.
The invariants are true in the base case $(\mathcal{F}_0, \ell_0)$, since it is the input ICPPL. Assume the lemma is true till the $j-1$th iteration. Let us consider the possible cases for each of the above invariants for the jth iteration.

※ *Invariant* I/II

I/IIa | *R is not a new set.* It is in $\mathcal{F}_{j-1}$. Thus trivially true by induction hypothesis.

I/IIb | *R is a new set.* If R is in $\mathcal{F}_j$ and not in $\mathcal{F}_{j-1}$, then it must be one of the new sets added in $\mathcal{F}_j$. In this case, it is clear that for each new set, the image under ℓ_j is a path since by definition the chosen sets S_1, S_2 are from $\mathcal{F}_{j-1}$ and due to the while loop condition,

$\ell_{j-1}(S_1)$, $\ell_{j-1}(S_2)$ have a common leaf. Thus invariant I is proved.
Moreover, due to induction hypothesis of invariant III and the definition of l_j in terms of l_{j-1}, invariant II is indeed true in the jth iteration for any of the new sets. If $R = S_1 \cap S_2$, $|R| = |S_1 \cap S_2| = |\ell_{j-1}(S_1) \cap \ell_{j-1}(S_2)| = |\ell_j(S_1 \cap S_2)| = |\ell_j(R)|$. If $R = S_1 \setminus S_2$, $|R| = |S_1 \setminus S_2| = |S_1| - |S_1 \cap S_2| = |\ell_{j-1}(S_1)| - |\ell_{j-1}(S_1) \cap \ell_{j-1}(S_2)| = |\ell_{j-1}(S_1) \setminus \ell_{j-1}(S_2)| = |\ell_j(S_1 \setminus S_2)| = |\ell_j(R)|$. Similarly if $R = S_2 \setminus S_1$.

※ *Invariant* III

IIIa | *R and R' are not new sets.* It is in $\mathcal{F}_{j-1}$. Thus trivially true by induction hypothesis.

IIIb | *Only one, say R, is a new set.* Due to invariant IV induction hypothesis, Lemma 2.3 and definition of ℓ_j, it follows that invariant III is true no matter which of the new sets R is equal to. If $R = S_1 \cap S_2$, $|R \cap R'| = |S_1 \cap S_2 \cap R'| = |\ell_{j-1}(S_1) \cap \ell_{j-1}(S_2) \cap \ell_{j-1}(R')| = |\ell_j(S_1 \cap S_2) \cap \ell_j(R')| = |\ell_j(R) \cap \ell_j(R')|$. If $R = S_1 \setminus S_2$, $|R \cap R'| = |(S_1 \setminus S_2) \cap R'| = |(\ell_{j-1}(S_1) \setminus \ell_{j-1}(S_2)) \cap \ell_{j-1}(R')| = |\ell_j(S_1 \cap S_2) \cap \ell_j(R')| = |\ell_j(R) \cap \ell_j(R')|$. Similarly, if $R = S_2 \setminus S_1$. Note R' is not a new set.

IIIc | *R and R' are new sets.* By definition, the new sets and their path images in path label ℓ_j are disjoint so $|R \cap R'| = |\ell_j(R) \cap \ell_j(R)| = 0$. Thus case proved.

※ *Invariant* IV This invariant is ensured at the end of every iteration due to lemma 2.4 and corollary 2.5.

□

Lemma 2.8. *In Algorithm 2, for all $j > 0$, at the end of the jth iteration of the* **while** *loop the four invariants given in Lemma 2.7 hold.*

Proof. Also note that removing a leaf from any path keeps the new path connected. Thus invariant I is obviously true. In each iteration, the set S is non-empty, and this follows from lemma 2.6. In every iteration $j > 0$, we remove exactly one element x from one set S in $\mathcal{F}$ and exactly one vertex v which is a leaf from one path $\ell_{j-1}(S)$ in T. This is because x is exclusive to S and v is exclusive to $\ell_{j-1}(S)$. Due to this fact, it is clear that the intersection cardinality equations do not change, i.e., invariants II, III, IV remain true. □

The following lemma formalizes the characterization of feasible path labeling.

Lemma 2.9. *If $(\mathcal{F}, \ell)$ is an ICPPL from a tree T , then there exists a bijection $\phi : supp(\mathcal{F}) \to V(T)$ such that for each $S \in \mathcal{F}$, $\ell(S) = \{\phi(x) | x \in S\}$.*

Proof. It is clear that in the end of every recursive call to Algorithm 3, the function ϕ' is one to one involving all the leaves in the tree passed as input to that recursive call. The tree pruning is done by only removing leaves in each call to the function and is done till the tree becomes empty. Thus the returned function $\phi : supp(\mathcal{F}) \to V(T)$ is a union of mutually exclusive one to one functions exhausting all vertices of the tree. In other words, it is a bijection from $supp(\mathcal{F})$ to $V(T)$ and the invariants of the two flitering algorithms guarantees that for each $S \in \mathcal{F}$, $\ell(S) = \{\phi(x) | x \in S\}$. □

Theorem 2.10. *A path labeling $(\mathcal{F}, \ell)$ on tree T is a feasible tree path labeling iff it is an ICPPL.*

Proof. From Lemma 2.9, we know that if $(\mathcal{F}, \ell)$ is an ICPPL then $(\mathcal{F}, \ell)$ is feasible. Now consider the case where $(\mathcal{F}, \ell)$ is feasible tree path labeling. Consequently, there exists a bijection $\phi : \mathcal{F} \to V(T)$ such that for each $S \in \mathcal{F}$, $\ell(S) = \{\phi(x) | x \in S\}$. Since ϕ is a bijection, it follows that for each $k \geq 1$,

and for all sets $S_1, S_2, \ldots, S_k \subseteq U$, $|\bigcap_{i=1}^{k} S_i| = |\bigcap_{i=1}^{k} \{\phi(x) | x \in S_i\}|$. In particular, it follows that ℓ is an ICPPL. □

3 Complexity of testing for an ICPPL

In this section, we focus on the problem of testing if a given set system $\mathcal{F}$ has an ICPPL in a given tree T. We also assume that support of $\mathcal{F}$ and $V(T)$ are of the same cardinality. The decision question of testing for the existence of an ICPPL for arbitrary T is clearly in NP. To understand the complexity of this question, we consider special trees. For the case when $T = P_n$, then as we have mentioned earlier finding an ICPPL is essentially equivalent to testing for COP of the binary matrix associated with $\mathcal{F}$. Recall that in this matrix, the columns correspond to U, and the rows correspond to the sets in $\mathcal{F}$. Indeed it is shown in [NS09] that one needs to find a path labeling ℓ of sets in $\mathcal{F}$ such that for any two $S_1, S_2 \in \mathcal{F}$, $|S_1 \cap S_2| = |\ell(S_1) \cap \ell(S_2)|$. Further, it is known from the work of [KKLV10] that the complexity of the case when $T = P_n$ is $DLOG$, that is the existence of the COP can be checked deterministically using logarithmic space as a function of the input size. It is this gap between $DLOG$ and NP that motivates us to explore the complexity of the existence of an ICPPL for different classes of trees. We consider the special case of a tree that we call the k-subdivided star.

k-subdivided star. A *star* graph is a complete bipartite graph $K_{1,p}$ which is clearly a tree and p is the number of leaves. The vertex with maximum degree is called the *center* of the star and the edges are called *rays* of the star. A *k-subdivided star* is a star with all its rays subdivided exactly k times. It is clear that all rays are of length $k+2$. First, we observe that by removing the center r from T, we get a collection of p vertex disjoint paths of length $k+1$, p being the number of leaves in T. We denote the rays by $R_1, \ldots, R_p$, and the number of vertices in R_i, $i \in [p]$ is $k+2$. Let $\langle v_{i1}, \ldots, v_{i(k+2)} = r \rangle$ denote the sequence of vertices in R_i, where v_{i1} is the leaf. Note that r is a common vertex to all R_i.

Restricted Nature of $\mathcal{F}$. In this section we consider the problem of finding an ICPPL from a k-subdivided star T to a given set system $\mathcal{F}$. Firstly we restrict our attention to $\mathcal{F}$ such that each set $X \in \mathcal{F}$ is of cardinality at most $k+2$. We restrict our attention only to those $\mathcal{F}$ for which the *overlap graph* (see next sentence) $\mathbb{O}(\mathcal{F})$ can be decomposed into overlap components that form a single rooted containment tree.

The overlap graph is well-known from the work of [KKLV10,NS09,Hsu02] and plays a very a crucial role in the algorithms for testing for COP. We use the notation in [KKLV10]. Sets S and S' are said to overlap, denoted by $S \between S'$, if S and S' have a non-empty intersection but neither of them is contained in the other. The overlap graph $\mathbb{O}(\mathcal{F})$ is a graph in which the vertices correspond to the sets in $\mathcal{F}$, and the vertices corresponding to the sets S and S' are adjacent if and only if they overlap. A connected component of $\mathbb{O}(\mathcal{F})$ is called an overlap component of $\mathcal{F}$. Clearly, two distinct overlap components, say $\mathcal{O}_1$ and $\mathcal{O}_2$, are such that any two sets $S_1 \in \mathcal{O}_1$ and $S_2 \in \mathcal{O}_2$ are disjoint, or, w.l.o.g, all the sets in $\mathcal{O}_1$ are contained within one set in $\mathcal{O}_2$. This containment relation naturally determines a decomposition of the overlap components into rooted containment trees. Finally, a set $S \in \mathcal{F}$ is called *marginal* if for all $S' \between S$, the sets $S \cap S'$ form a single inclusion chain [KKLV10]. Additionally, if S is such that it is contained in no other marginal set in $\mathcal{F}$, then it is called *super-marginal*.

3.1 Description of the Algorithm

The first step is to test if the root overlap component in the containment tree has an ICPPL, then the descendant overlap components are testing for an path labeling that satisfies only the first two conditions of an ICPPL (recall, this is an ICPIA from [NS09]). The reason is that each descendant overlap component is contained in *one* set in its parent in the containment tree. If the parent has an ICPPL, then this set corresponds to a *path*, and this basically means that for the descendant overlap components we need to test only for an ICPIA. Therefore, in this section, we address the question of testing for an ICPPL in the root overlap component. The root overlap component, the k-subdivided star, and the center of the star are denoted by $\mathcal{O}$, T and vertex r, respectively.

Values and Sets maintained by the Algorithm: For each set $X \in \mathcal{O}$, we will maintain a integer value that we denote by $s(X)$. The algorithm proceeds in iterations and maintains three sets $\mathcal{O}$, $\mathcal{L}$, and $\mathcal{T}$. $\mathcal{O}$ contains the those sets that have not been assigned a path, $\mathcal{L}$ contains those sets that have been assigned a path, and $\mathcal{T}$ contains those sets that have been given a path containing r partially from a ray, and the second ray from which the rest of the path is to assigned is yet undecided.

Specification of the Algorithm: The basic idea of the algorithm is to assign paths to the sets in $\mathcal{O}$ guided by the overlap relation. In the case when T is a path [NS09], the idea is to identify the marginal sets, that is the set that gets the leftmost vertex in the path. Subsequently, iteratively an unlabeled set that overlaps with a labeled set is selected, and a unique path is assigned to it such that the labeling remains an ICPPL. If such a unique path cannot be found, the algorithm reports failure and stops. In the case when T is a k-subdivided star, we use the same idea of identifying a unique path to an unlabeled set such that the resulting labeling remains an ICPPL.

The Candidate path for an unlabeled X: Let $X \in \mathcal{O}$ and let $Y \in \mathcal{L}$ such that $X \between Y$ and R is a ray such that $\ell(Y) \subseteq R$. Let P be a path such that $P \subseteq R$ and, and $|P \cap \ell(Z)| = |X \cap Z|$ for all $Z \in \mathcal{L}$. We refer to P as the *candidate path for* X, and there are two possibilities when P exists:$|P| = |X|$ and $|P| < |X|$. In the latter possibility, it means that the path for X has to be in the union of two rays, and specifically in the case of the k-subdivided star must contain the root r. The algorithm exits reporting failure when a candidate path P does not exist. It is important to note that there is no backtracking search for the path labeling.

I. **Step 1- Identifying marginal set for each ray:** Let $S = \{X_1, \ldots, X_s\}$ denote the super-marginal sets from $\mathcal{O}_1$. If $|S| = s \neq p$, then exit reporting failure. Else, to each $X_j \in S$, assign the path from R_j such that the path contains the leaf in R_j, and the number of vertices in the path is $|X_j|$. This path is referred to as $\ell(X_j)$. Set $s(X_j) = 0$. X_j is removed from $\mathcal{O}$, and X_j is added to $\mathcal{L}$.

II. **Step 2 - Saturating the Rays:** Let $X \in \mathcal{O}$ be set such that $X \between Y$ for some $Y \in \mathcal{L}$. If the candidate path for X does not exist, exit reporting failure. Else, let P denote the candidate path for X such that P is contained in the ray containing Y and $|P| = |X|$. Then let $\ell(X) = P$. Set $s(X) = 0$, and add X into $\mathcal{L}$ and remove it from $\mathcal{O}$. Iterate till the step is no longer applicable.

III. **Step 3 - Partial labeling:** At the beginning of this step, every $X \in \mathcal{O}$ such that $X \between Y$ for some $Y \in \mathcal{L}$ has the property that the candidate path for X, say P, is such that $|P| < |X|$. Now, let $\ell(X) = P$, $s(X) = |X| - |P|$. Add X to $\mathcal{T}$ and remove it from $\mathcal{O}$. Iterate till this step is not applicable.

IV. **Step 4 - Completing the labeling of partially labeled sets:** If some vertex of tree T is not in $\ell(X)$ for some $X \in \mathcal{L} \cup \mathcal{T}$, exit reporting failure. Let $X \in \mathcal{T}$, that is X is partially

labeled. Let R be the ray that contains $\ell(X)$. If there is a $Y \in \mathcal{L}$ such that $|X \cap Y| > 1$ and $\ell(Y)$ is contained in a ray R' different from R, then *extend* $\ell(X)$ to a path additionally of length $s(X)$ in R' containing the root r. Set $s(X) = 0$, add X into $\mathcal{L}$, and remove from $\mathcal{T}$. Iterate till this step is not applicable.

V. **Step 5:** Let $X, Y \in \mathcal{T}$ such that $\ell(X)$ and $\ell(Y)$ are in different rays R and R', respectively. Further, if $|\ell(X)| + |\ell(Y)| - 1 = |X \cup Y|$, then extend $\ell(X)$ to a path additionally of length $s(X)$ in R' containing r. Set $s(X) = 0$, add X into $\mathcal{L}$ and remove it from $\mathcal{T}$. The same step is done for Y in R. Iterate till this step is not applicable.

VI. **Step 6:** Let $X \in \mathcal{T}$. Find a ray R such that after extending $\ell(X)$ to a path P additionally of length $s(X)$ in R containing r, for any $Z \in \mathcal{L} \cup \mathcal{T}$ such that $\ell(Z) \subseteq R$, $|P \cap \ell(Z)| = |X \cap Z|$. Set $\ell(X) = P, s(X) = 0$, add X into $\mathcal{L}$ and remove it from $\mathcal{T}$. If such a ray R cannot be found, exit reporting failure. Iterate till this step is not applicable.

VII. **Step 7: Completing the labeling:** If $\mathcal{O}$ is empty, skip to Step 8. Let R_{ij} be the path formed by the two rays R_i and R_j. Let $\mathcal{L}_{ij} = \{Y \in \mathcal{L} | \ell(Y) \subseteq R_{ij}\}$. For every $X \in \mathcal{O}$, such that $X \between Z$ for some $Z \in \mathcal{L}_{ij}$, check if there is a path P such that $|P| = |X|$ and $|P \cap \ell(Y)| = |X \cap Y|$ for all $Y \in \mathcal{L}$. If P exists, the set $\ell(X) = P, s(X) = 0$, remove X from $\mathcal{O}$ and add X to $\mathcal{L}$. Iterate over all pairs R_i and R_j, $1 \leq i \neq j \leq p$.

VIII. **Step 8:** If $\mathcal{O}$ is non-empty then exit reporting failure. If ℓ is a path labeling for $\mathcal{O}$ that satisfies the properties of an ICPPL, return ℓ, else exit reporting failure.

Lemma 3.1. *If $X \in \mathcal{F}$ is super-marginal and ℓ is a feasible tree path labeling to tree T, then $\ell(X)$ will contain a leaf in T.*

Proof. Suppose $X \in \mathcal{F}$ is super-marginal and $(\mathcal{F}, \ell)$ is a feasible path labeling from T. Assume $\ell(X)$ does not have a leaf. Let R_i be one of the rays (or the only ray) $\ell(X)$ is part of. Since X is in a connected overlap component, there exists $Y_1 \in \mathcal{F}$ and $X \nsubseteq Y_1$ such that $Y_1 \between X$ and Y_1 has at least one vertex closer to the leaf in R_i than any vertex in X. Similarly with the same argument there exists $Y_2 \in \mathcal{F}$ with same subset and overlap relation with X except it has has at least one vertex farther away from the leaf in R_i than any vertex in X. Clearly $Y_1 \cap X$ and $Y_2 \cap X$ cannot be part of same inclusion chain which contradicts that assumption X is super-marginal. Thus the claim is proved. □

Lemma 3.2. *If $\mathcal{O}$ does not have any super-marginal edges, then in any feasible path labeling ℓ of $\mathcal{O}$ with paths from T is such that, for any set X for which $\ell(X)$ contains a leaf, $|X| \geq k + 3$.*

Proof. The proof of this lemma is by contradiction. Let X be a sets such that $|X| \leq k+2$ and that $\ell(X)$ has a leaf. This implies that the overlap regions with X, which are captured by the overlap regions with $\ell(X)$, will form a single inclusion chain. This shows that X is a marginal set which contradicts the assumption that $\mathcal{O}$ does not have super-marginal sets. □

This lemma is used to prove the next lemma for the case when for all $X \in \mathcal{O}$, $|X| \leq k + 2$. The proof is left out as it just uses the previous lemma and the fact that the sets in X have at most $k + 2$ elements.

Lemma 3.3. *If there is a feasible path labeling for $\mathcal{O}$ in T, then there are exactly p super-marginal sets.*

These lemmas now are used to prove the following theorem.

Theorem 3.4. *Given $\mathcal{O}$ and a k-subdivided star T, the above algorithm decides correctly if there is a feasible path labeling ℓ.*

References

[ABH98] J. E. Atkins, E. G. Boman, and B. Hendrickson. A spectral algorithm for seriation and the consecutive ones problem. *SICOMP: SIAM Journal on Computing*, 28, 1998.

[BL76] Kellogg S. Booth and George S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using *PQ*-tree algorithms. *Journal of Computer and System Sciences*, 13(3):335–379, December 1976.

[FG65] D. R. Fulkerson and O. A. Gross. Incidence matrices and interval graphs. *Pac. J. Math.*, 15:835–855, 1965.

[Gol04] Martin Charles Golumbic. *Algorithmic graph theory and perfect graphs*, volume 57 of *Annals of Discrete Mathematics*. Elsevier Science B.V., 2004. Second Edition.

[HL06] Dorit S. Hochbaum and Asaf Levin. Cyclical scheduling and multi-shift scheduling: Complexity and approximation algorithms. *Discrete Optimization*, 3(4):327–340, 2006.

[Hsu01] Wen-Lian Hsu. PC-trees vs. PQ-trees. *Lecture Notes in Computer Science*, 2108:207–217, 2001.

[Hsu02] Wen-Lian Hsu. A simple test for the consecutive ones property. *J. Algorithms*, 43(1):1–16, 2002.

[HT02] Hochbaum and Tucker. Minimax problems with bitonic matrices. *NETWORKS: Networks: An International Journal*, 40, 2002.

[Ken69] D. Kendall. Incidence matrices, interval graphs and seriation in archaeology. *Pacific Journal of Mathematics*, 3(28):565–570, 1969.

[KKLV10] Johannes Köbler, Sebastian Kuhnert, Bastian Laubner, and Oleg Verbitsky. Interval graphs: Canonical representation in logspace. *Electronic Colloquium on Computational Complexity (ECCC)*, 17:43, 2010.

[Kou77] Lawrence T. Kou. Polynomial complete consecutive information retrieval problems. *SIAM Journal on Computing*, 6(1):67–75, March 1977.

[McC04] Ross M. McConnell. A certifying algorithm for the consecutive-ones property. In *SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms)*, 2004.

[MM96] J. Meidanis and Erasmo G. Munuera. A theory for the consecutive ones property. In *Proceedings of WSP'96 - Third South American Workshop on String Processing*, pages 194–202, 1996.

[NS09] N. S. Narayanaswamy and R. Subashini. A new characterization of matrices with the consecutive ones property. *Discrete Applied Mathematics*, 157(18):3721–3727, 2009.

[Rob51] W. S. Robinson. A method for chronologically ordering archaeological deposits. *American Antiquity*, 16(4):293–301, 1951.

[Tuc72] Alan Tucker. A structure theorem for the consecutive 1's property. *J. Comb. Theory Series B*, 12:153–162, 1972.