Tree Path Labeling of Path Hypergraphs - A Generalization of Consecutive Ones Property

N.S. Narayanaswamy and Anju Srinivasan[1]

1 **Computer Science and Engineering Department,**
Indian Institute of Technology Madras, Chennai - 600036, India
swamy@cse.iitm.ernet.in, asz@cse.iitm.ac.in

Abstract

We consider the following constraint satisfaction problem. Given (i) a set system $\mathcal{F} \subseteq (2^U \setminus \emptyset)$ of a finite set U of cardinality n, (ii) a tree T of size n and (iii) a bijection ℓ, defined as *tree path labeling*, mapping the sets in $\mathcal{F}$ to paths in T, does there exist at least one bijection $\phi : U \to V(T)$ such that for each $S \in \mathcal{F}$, $\{\phi(x) \mid x \in S\} = \ell(S)$? A tree path labeling of a set system is called *feasible* if there exists such a bijection ϕ. In this paper, we characterize feasible tree path labeling of a given set system to a tree. This result is a natural generalization of results on matrices with the Consecutive Ones Property. COP is a special instance of tree path labeling problem when T is a path. We also present an algorithm to find the tree path labeling of a given set system when T is a *k-subdivided star* as well as a characterization of set systems which have a feasible tree path labeling.

1998 ACM Subject Classification XXXXXXXX TBD XXXXXXXX Dummy classification

Keywords and phrases XXXXXXXX TBD XXXXXXXX

1 Introduction

Consecutive ones property (COP) of binary matrices is a widely studied combinatorial problem. 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. It has several practical applications in diverse fields including scheduling [9], information retrieval [13] and computational biology [1]. Further, it is a tool in graph theory [7] for interval graph recognition, characterization of hamiltonian graphs, and in integer linear programming [8, 9]. Recognition of COP is polynomial time solvable by several algorithms. PQ trees [3], variations of PQ trees [16, 10, 11, 15], ICPIA [17] are the main ones.
The problem of COP testing can be easily seen as a simple constraint satisfaction problem involving a system of sets from a universe. Every column of the binary matrix can be converted into a set of integers which are the indices of the rows with 1s in that column. When observed in this context, if the matrix has the COP, a reordering of its rows will result in sets that have only consecutive integers. In other words, the sets after reordering are intervals. Indeed the problem now becomes finding interval assignments to the given set system [17] with a single permutation of the universe (set of row indices) which permutes each set to its interval. The result in [17] characterize interval assignments to the sets which can be obtained from a single permutation of the rows. They show that for each set, the cardinality of the interval assigned to it must be same as the cardinality of the set, and the intersection cardinality of any two sets must be same as the interesction cardinality of the corresponding intervals. While this is naturally a necessary condition, [17] shows this is indeed sufficient. Such an interval assignment is called an Intersection Cardinality Preserving Interval Assignment (ICPIA). Finally, the idea of decomposing a given 0-1 matrix into prime

Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

matrices to check for COP is adopted from [11] to test if an ICPIA exists for a given set system.

Our Work. A natural generalization of the interval assignment problem is feasible tree path labeling problem of a set system. The problem is defined as follows - given a set system $\mathcal{F}$ from a universe U and a tree T, does there exist a bijection from the U to the vertices of T such that each set in the system maps to a path in T. We refer to this as the tree path labeling problem for an input $(\mathcal{F}, T)$ pair. As a special case if T is a path the problem becomes the interval assignment problem. We focus on the question of generalizing the notion of an ICPIA [17] to characterize feasible path assignments. We show that for a given set system $\mathcal{F}$, a tree T, and an assignment of paths from T to the sets, there is a bijection between U and $V(T)$ if and only if all intersection cardinalities among any three sets (not necessarily distinct) is same as the intersection cardinality of the paths assigned to them and the input runs a filtering algorithm (described in this paper) without prematurely exiting. This characterization is proved constructively and it gives a natural data structure that stores all the relevant bijections between U and $V(T)$. Further, the filtering algorithm is also an efficient algorithm to test if a tree path labeling to the set system is feasible. This generalizes the result in [17].

It is an interesting fact that for a matrix with the COP, the intersection graph of the corresponding set system is an interval graph. A similar connection to a subclass of chordal graphs and a superclass of interval graphs exists for the generalization of COP. In this case, the intersection graph of the corresponding set system must be a *path graph*. Chordal graphs are of great significance, extensively studied, and have several applications. One of the well known and interesting properties of a chordal graphs is its connection with intersection graphs [7]. For every chordal graph, there exists a tree and a family of subtrees of this tree such that the intersection graph of this family is isomorphic to the chordal graph [19, 6, 2]. These trees when in a certain format, are called clique trees [18] of the chordal graph. This is a compact representation of the chordal graph. There has also been work done on the generalization of clique trees to clique hypergraphs [14]. If the chordal graph can be represented as the intersection graph of paths in a tree, then the graph is called path graph [7]. Therefore, it is clear that if there is a bijection from U to $V(T)$ such that for every set, the elements in it map to vertices of a unique path in T, then the intersection graph of the set system is indeed a path graph. However, this is only a necessary condition and can be checked efficiently because path graph recognition is polynomial time solvable[6, 20]. Indeed, it is possible to construct a set system and tree, such that the intersection graph is a path graph, but there is no bijection between U and $V(T)$ such that the sets map to paths. Path graph isomorphism is known be isomorphism-complete, see for example [12]. An interesting area of research would be to see what this result tells us about the complexity of the tree path labeling problem (not covered in this paper). In the later part of this paper, we decompose our search for a bijection between U and $V(T)$ into subproblems. Each subproblem is on a set subsystem in which for each set, there is another set in the set subsystem with which the intersection is *strict* - i.e., there is a non-empty intersection, but neither is contained in the other. This is in the spirit of results in [11, 17] where to test for the COP in a given matrix, the COP problem is solved on an equivalent set of prime matrices.

Roadmap. In Section 2 we present the necessary preliminaries, in Section 3 we present our characterization of feasible tree path assignments, and in Section 5 we present the characterizing subproblems for finding a bijection between U and $V(T)$ such that sets map to tree paths. In Section 4 we present a polynomial time algorithm to find the tree path labeling of a given set system from a given k-subdivided tree.

2 Preliminaries

In this paper, the set $\mathcal{F} \subseteq (2^U \setminus \emptyset)$ is a *set system* of a universe U with $|U| = n$. The *support* of a set system $\mathcal{F}$ denoted by $supp(\mathcal{F})$ is the union of all the sets in $\mathcal{F}$, i.e., $supp(\mathcal{F}) = \bigcup_{S \in \mathcal{F}} S$. For the purposes of this paper, a set system is required to "cover" the universe, i.e. $supp(\mathcal{F}) = U$. In brief, the *intersection graph* $\mathbb{I}(\mathcal{F})$ of a set system $\mathcal{F}$ is a graph such that its vertex set has a bijection to $\mathcal{F}$ and there exists an edge between two vertices iff their corresponding sets have a non-empty intersection [7].
The graph T represents a given tree with $|V(T)| = n$. A *path system* $\mathcal{P}$ is a set system of paths from T i.e., $\mathcal{P} = \{P \mid P \subseteq V,\ T[P]$ is a path.$\}$ A *star* graph is a complete bipartite graph $K_{1,l}$ which is clearly a tree and l 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. The path from the center to a leaf is called a ray of a k-subdivided star and they are all of length $k+2$.
A graph G that is isomorphic to the intersection graph $\mathbb{I}(\mathcal{P})$ of a path system $\mathcal{P}$ of T, is a *path graph*. This isomorphism gives a bijection $\ell' : V(G) \to \mathcal{P}$ and is called a path labeling of G. Moreover, for the purposes of this paper, we require that in a path labeling, $supp(\mathcal{P}) = V(T)$. If $G = \mathbb{I}(\mathcal{F})$ where $\mathcal{F}$ is any set system, then clearly there is a bijection $\ell : \mathcal{F} \to \mathcal{P}$ such that $\ell(S) = \ell'(v_S)$ where v_S is the vertex corresponding to set S in $\mathbb{I}(\mathcal{F})$ for any $S \in \mathcal{F}$. This bijection ℓ is called the *path labeling* of set system $\mathcal{F}$ and the path system $\mathcal{P}$ may be alternatively denoted as $\mathcal{F}^\ell$.
A set system $\mathcal{F}$ can be alternatively represented by a *hypergraph* $\mathcal{H}_\mathcal{F}$ whose vertex set is $supp(\mathcal{F})$ and hyperedges are the sets in $\mathcal{F}$. This is a known representation for interval systems in literature [4]. We extend this definition here to path systems. Two hypergraphs $\mathcal{H}$, $\mathcal{K}$ are said to be isomorphic to each other, denoted by $\mathcal{H} \cong \mathcal{K}$, iff there exists a bijection $\phi : supp(\mathcal{H}) \to supp(\mathcal{K})$ such that for all sets $H \subseteq supp(\mathcal{H})$, H is a hyperedge in $\mathcal{H}$ iff K is a hyperedge in $\mathcal{K}$ where $K = \{\phi(x) \mid x \in H\}$. If $\mathcal{H}_\mathcal{F} \cong \mathcal{H}_\mathcal{P}$ where $\mathcal{P}$ is a path system, then $\mathcal{H}_\mathcal{F}$ is called a *path hypergraph* and $\mathcal{P}$ is called *path representation* of $\mathcal{H}_\mathcal{F}$. If isomorphism is $\phi : supp(\mathcal{H}_\mathcal{F}) \to supp(\mathcal{H}_\mathcal{P})$, then it is clear that there is an induced path labeling $l_\phi : \mathcal{F} \to \mathcal{P}$ to the set system. In the rest of the document, we may use $\mathcal{F}$ and $\mathcal{H}_\mathcal{F}$ interchangeably to refer the set system and/or its hypergraph.
An *overlap graph* $\mathbb{O}(\mathcal{F})$ of a set system $\mathcal{F}$ is a graph such that its vertex set has a bijection to $\mathcal{F}$ and there exists an edge between two vertices iff their corresponding sets overlap. Two sets A and B are said to overlap, denoted by $A \between B$, if they have a non-empty intersection and neither is contained in the other i.e. $A \between B$ iff $A \cap B \neq \emptyset, A \nsubseteq B, B \nsubseteq A$. Thus $\mathbb{O}(\mathcal{F})$ is a subgraph of $\mathbb{I}(\mathcal{F})$ and not necessarily connected. Each connected component of $\mathbb{O}(\mathcal{F})$ is called an *overlap component*. If there are d overlap components in $\mathbb{O}(\mathcal{F})$, the set subsystems are denoted by $\mathcal{O}_1, \mathcal{O}_2, \ldots \mathcal{O}_d$. Clearly $\mathcal{O}_i \subseteq \mathcal{F}, i \in [d]$. For any $i, j \in [d]$, it can be verified that one of the following is true.

i. $supp(\mathcal{O}_i)$ and $supp(\mathcal{O}_j)$ are disjoint
ii. $supp(\mathcal{O}_i)$ is a subset of a set in $\mathcal{O}_j$
iii. $supp(\mathcal{O}_j)$ is a subset of a set in $\mathcal{O}_i$

A path labeling $\ell : \mathcal{F} \to \mathcal{P}$ of setsystem $\mathcal{F}$ is defined to be *feasible* if their hypergraphs are isomorphic to each other, $\mathcal{H}_\mathcal{F} \cong \mathcal{H}_\mathcal{P}$ and if this isomorphism $\phi : supp(\mathcal{F}) \to supp(\mathcal{P})$ induces a path labeling $\ell_\phi : \mathcal{F} \to \mathcal{P}$ such that $\ell_\phi = \ell$.
For any partial order $(X, \preccurlyeq)$, the notation $mub(X)$ represents an element $X_m \in X$ which is called a *maximal upper bound* on X. The element X_m is a maximal upper bound of X if $\nexists X_q \in X$ such that $X_m \preccurlyeq X_q$.

An *in-tree* is a directed rooted tree in which all edges are directed toward to the root.

3 Characterization of Feasible Tree Path Labeling

Consider a path labeling $\ell : \mathcal{F} \rightarrow \mathcal{P}$ for set system $\mathcal{F}$ and path system $\mathcal{P}$ on the given tree T. We call ℓ an *Intersection Cardinality Preserving Path Labeling (ICPPL)* if it has the following properties.

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

The following two lemma are useful in characterizing feasible tree path labelings. Their proofs are in the appendix.

▶ **Lemma 3.1.** *If ℓ is an ICPPL, and $S_1, S_2, S_3 \in \mathcal{F}$, then $|S_1 \cap (S_2 \setminus S_3)| = |\ell(S_1) \cap (\ell(S_2) \setminus \ell(S_3))|$.*

▶ **Lemma 3.2.** *Consider four paths in a tree P_1, P_2, P_3, P_4 such that they have non-empty pairwise intersections and paths P_1, P_2 share a leaf. Then there exist distinct integers $i, j, k \in \{1, 2, 3, 4\}$ such that, $P_1 \cap P_2 \cap P_3 \cap P_4 = P_i \cap P_j \cap P_k$.*

In the remaining part of this section we describe an algorithmic characterization for a feasible tree path labeling. We show that a path labeling is feasible if and only if it is an ICPPL and it successfully passes the filtering algorithms 1 and 2. One direction of this claim is clear: that if a path labeling is feasible, then all intersection cardinalities are preserved, i.e. the path labeling is an ICPPL. Algorithm 1 has no premature exit condition hence any input will go through it. Algorithm 2 has an exit condition at line 7. It can be easily verified that X cannot be empty if ℓ is a feasible path labeling. The reason is that a feasible path labeling has an associated bijection between $supp(\mathcal{F})$ and $V(T)$ i.e. $supp(\mathcal{F}^\ell)$ such that the sets map to paths, "preserving" the path labeling. The rest of the section is devoted to constructively proving that it is sufficient for a path labeling to be an ICPPL and pass the two filtering algorithms. To describe in brief, the constructive approaches refine an ICPPL iteratively, such that at the end of each iteration we have a "filtered" path labeling, and finally we have a path labeling that defines a family of bijections from $supp(\mathcal{F})$ to $V(T)$ i.e. $supp(\mathcal{F}^\ell)$.
First we present Algorithm 1 or Filter 1, and prove its correctness. This algorithm refines the path labeling by considering pairs of paths that share a leaf.

▶ **Lemma 3.3.** *In Algorithm 1, if input $(\mathcal{F}, \ell)$ is a feasible path assignment then at the end of jth iteration of the* **while** *loop, $j \geq 0$, $(\mathcal{F}_j, \ell_j)$ is a feasible path assignment.*

Proof. We will prove this by mathematical induction on the number of iterations. The base case $(\mathcal{F}_0, \ell_0)$ is feasible since it is the input itself. Assume the lemma is true till $j-1$th iteration. i.e. there is a bijection $\phi : supp(\mathcal{F}_{j-1}) \rightarrow V(T)$ such that the induced path labeling on $\mathcal{F}_{j-1}$, $\ell_{\phi[\mathcal{F}_{j-1}]}$ is equal to ℓ_{j-1}. We will prove that ϕ is also the bijection that makes $(\mathcal{F}_j, \ell_j)$ feasible. Note that $supp(\mathcal{F}_{j-1}) = supp(\mathcal{F}_j)$ since the new sets in $\mathcal{F}_j$ are created from basic set operations to the sets in $\mathcal{F}_{j-1}$. For the same reason and ϕ being a bijection, it is clear that $\ell_{\phi[\mathcal{F}_j]}(S_1 \setminus S_2) = \ell_{\phi[\mathcal{F}_{j-1}]}(S_1) \setminus \ell_{\phi[\mathcal{F}_{j-1}]}(S_2)$. Now observe that $\ell_j(S_1 \setminus S_2) = \ell_{j-1}(S_1) \setminus \ell_{j-1}(S_2) = \ell_{\phi[\mathcal{F}_{j-1}]}(S_1) \setminus \ell_{\phi[\mathcal{F}_{j-1}]}(S_2)$. Thus the induced path labeling $\ell_{\phi[\mathcal{F}_j]} = \ell_j$. Therefore lemma is proven. ◀ ◀

▶ **Lemma 3.4.** *In Algorithm 1, at the end of jth iteration, $j \geq 0$, of the* **while** *loop of Algorithm 1, the following invariants are maintained.*

Algorithm 1 FILTER 1: Refine ICPPL ($\mathcal{F}$, ℓ)

Let $\mathcal{F}_0 = \mathcal{F}$
Let $\ell_0(S) = \ell(S)$ for all $S \in \mathcal{F}_0$
$j = 1$
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**
  $\mathcal{F}_j = (\mathcal{F}_{j-1} \setminus \{S_1, S_2\}) \cup \{S_1 \cap S_2, S_1 \setminus S_2, S_2 \setminus S_1\}$ `/* Remove` S_1`,` S_2 `and add the ''filtered'' sets */`
  for all $S \in \mathcal{F}_{j-1}$ such that $S \neq S_1$ and $S \neq S_2$, set $\ell_j(S) = \ell_{j-1}(S)$ `/* Do not change path labeling for any set other than` S_1`,` S_2 `*/`
  $\ell_j(S_1 \cap S_2) = \ell_{j-1}(S_1) \cap \ell_{j-1}(S_2)$
  $\ell_j(S_1 \setminus S_2) = \ell_{j-1}(S_1) \setminus \ell_{j-1}(S_2)$
  $\ell_j(S_2 \setminus S_1) = \ell_{j-1}(S_2) \setminus \ell_{j-1}(S_1)$
  if $(\mathcal{F}_j, \ell_j)$ does not satisfy condition (iii) of ICPPL **then**
    exit
  end if
  $j = j + 1$
end while
$\mathcal{F}' = \mathcal{F}_j$, $\ell' = \ell_j$
return $\mathcal{F}', \ell'$

Invariant I: $\ell_j(R)$ is a path in T for each $R \in \mathcal{F}_j$
Invariant II: $|R| = |\ell_j(R)|$ for each $R \in \mathcal{F}_j$
Invariant III: For any two $R, R' \in \mathcal{F}_j$, $|R \cap R'| = |\ell_j(R) \cap \ell_j(R')|$
Invariant IV: For any three, $R, R', R'' \in \mathcal{F}_j$, $|R \cap R' \cap R''| = |\ell_j(R) \cap \ell_j(R') \cap \ell_j(R'')|$

Proof. The detailed proofs of the some of the cases below are in the appendix. Proof is by induction on the number of iterations, j. In the rest of the 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 base case, ℓ_0 is an ICPPL on $\mathcal{F}_0$, since it is the input. Assume the lemma is true till the $j-1$ iteration. Let us consider the possible cases for each invariant for the jth iteration.

Case 1: *Invariant I and II*

Case 1.1: *R is not a new set.* If R is in $\mathcal{F}_{j-1}$, then by induction hypothesis this case is trivially proven.

Case 1.2: *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 proven. Moreover, due to induction hypothesis of invariant III ($j-1$th iteration) 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.

Case 2: *Invariant III*

Case 2.1: *R and R' are not new sets.* Trivially true by induction hypothesis.

Case 2.2: *Only one, say R, is a new set.* Due to invariant IV induction hypothesis, lemma 3.1 and definition of l_j, it follows that invariant III is true no matter which of the new sets R is equal to. It is important to note that R' is not a new set here.

Case 2.3: *R and R' are new sets.* By definition, the new sets and their path images in path label l_j are disjoint so $|R \cap R'| = |l_j(R) \cap l_j(R)| = 0$. Thus case proven.

Case 3: *Invariant IV* Due to the condition in line 9, invariant IV is ensured at the end of every iteration.

◀ ◀

▶ **Lemma 3.5.** *In Algorithm 1, consider an ICPPL input $(\mathcal{F}, \ell)$ which is also a feasible path labeling. Then in the execution of the algorithm its exit condition in line 9, i.e. failure of three way intersection preservation, will not be true in any iteration of the* **while** *loop and the algorithm executes without a premature exit.*

Proof. This proof uses mathematical induction on the number of iterations j, $j \geq 0$, of the **while** loop that executed without exiting. The base case, $j = 0$ is obviously true since the input is an ICPPL and the exit condition cannot hold true due to ICPPL condition (iii). Assume the algorithm executes till the end of $j-1$th iteration without exiting at line 9. Consider the jth iteration. From lemma 3.3 we know that $(\mathcal{F}_j, \ell_j)$ and $(\mathcal{F}_{j-1}, \ell_{j-1})$ are feasible. Thus there exists a bijection $\phi : supp(\mathcal{F}) \rightarrow V(T)$ such that the induced path labeling on $\mathcal{F}_j$, $\ell_{\phi[\mathcal{F}_j]}$ and on $\mathcal{F}_{j-1}$, $\ell_{\phi[\mathcal{F}_{j-1}]}$ are equal to ℓ_j and ℓ_{j-1} respectively. We need to prove that for any $R, R', R'' \in \mathcal{F}_j$, $|R \cap R' \cap R''| = |\ell_j(R) \cap \ell_j(R') \cap \ell_j(R'')|$.
The following are the possible cases that could arise. From argument above, $|\ell_j(R) \cap \ell_j(R') \cap \ell_j(R'')| = |\ell_{\phi[\mathcal{F}_j]}(R) \cap \ell_{\phi[\mathcal{F}_j]}(R') \cap \ell_{\phi[\mathcal{F}_j]}(R'')|$

Case 1: *None of the sets are new.* $R, R', R'' \in \mathcal{F}_{j-1}$. We know $(\mathcal{F}_{j-1}, \ell_{j-1})$ is feasible. Thus $|R \cap R' \cap R''| = |\ell_{j-1}(R) \cap \ell_{j-1}(R') \cap \ell_{j-1}(R'')| = |\ell_j(R) \cap \ell_j(R') \cap \ell_j(R'')|$.

Case 2: *Only one, say R, is a new set.* Let $R = S_1 \cap S_2$ (S_1, S_2 are defined in the proof of lemma 3.4). Now we have $|R \cap R' \cap R''| = |S_1 \cap S_2 \cap R' \cap R''| = |\ell_{j-1}(S_1) \cap \ell_{j-1}(S_2) \cap \ell_{j-1}(R') \cap \ell_{j-1}(R'')| = |\ell_j(R) \cap \ell_j(R') \cap \ell_j(R'')|$. Thus proven. If R is any of the other new sets, the same claim can be verified using lemma 3.1.

Case 3: *At least two of R, R', R'' are new sets.* The new sets are disjoint hence this case is vacuously true.

◀ ◀

As a result of Algorithm 1 each leaf v in T is such that there is exactly one set in $\mathcal{F}$ such that v is a node in the path assigned to it. In Algorithm 2 we identify elements in $supp(\mathcal{F})$ whose images are leaves in a feasible path labeling if one exists. Let vertex $v \in T$ be the unique leaf incident on a path image P in ℓ. We define a new path labeling ℓ_{new} such that $\ell_{new}(\{x\}) = \{v\}$ where x an arbitrary element from $\ell^{-1}(P) \setminus \bigcup_{\hat{P} \neq P} \ell^{-1}(\hat{P})$. In other words, x is an element present in no other set in $\mathcal{F}$ except $\ell^{-1}(P)$. This is intuitive since v is present in no other path image other than P. The element x and leaf v are then removed from the set $\ell^{-1}(P)$ and path P respectively. The tree is pruned off v and the refined set system will have $\ell^{-1}(P) \setminus \{x\}$ instead of $\ell^{-1}(P)$. After doing this for all leaves in T, all path images in the new path labeling ℓ_{new} except single leaf labels (the pruned out vertex is called the *leaf label* for the corresponding set item) are paths from the pruned tree $T_0 = T \setminus \{v \mid v \text{ is a leaf in } T\}$. Algorithm 2 is now presented with details.

▶ **Lemma 3.6.** *In Algorithm 2, for all $j \geq 0$, at the end of the jth iteration the four invariants given in lemma 3.4 are valid.*

Proof. Suppose the input ICPPL is also feasible but X is empty. This will prematurely exit the algorithm and thus prevent us from finding the permutation. We will now show that this cannot happen. We know that v is an element of $\ell_{j-1}(S_1)$. Since it is uniquely present in $\ell_{j-1}(S_1)$, it is clear that $v \in \ell_{j-1}(S_1) \setminus \bigcup_{S \in \mathcal{F}_{j-1}, S \neq S_1} \ell_{j-1}(S)$. Note that for any $x \in S_1$ it is contained in at least two sets due to our assumption about cardinality of X. Let $S_2 \in \mathcal{F}_{j-1}$ be another set that contains x. From the above argument, we know $v \notin \ell_{j-1}(S_2)$. Therefore

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

Let $\mathcal{F}_0 = \mathcal{F}$
Let $\ell_0(S) = \ell(S)$ for all $S \in \mathcal{F}_0$. Note: Path images are such that no two path images share a leaf.
$j = 1$
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**
  $\mathcal{F}_j = \mathcal{F}_{j-1} \setminus \{S_1\}$
  for all $S \in \mathcal{F}_{j-1}$ such that $S \neq S_1$ set $\ell_j(S) = \ell_{j-1}(S)$
  $X = S_1 \setminus \bigcup_{S \in \mathcal{F}_{j-1}, S \neq S_1} S$
  if X is empty **then**
    exit
  end if
  Let $x =$ arbitrary element from X
  $\mathcal{F}_j = \mathcal{F}_j \cup \{\{x\}, S_1 \setminus \{x\}\}$
  $\ell_j(\{x\}) = \{v\}$
  $\ell_j(S_1 \setminus \{x\}) = \ell_{j-1}(S_1) \setminus \{v\}$
  $j = j + 1$
end while
$\mathcal{F}' = \mathcal{F}_j$
$\ell' = \ell_j$
return $\mathcal{F}', \ell'$

there cannot exist a permutation that maps elements of S_2 to $\ell_{j-1}(S_2)$. This contradicts our assumption that the input is feasible. Thus X cannot be empty if input is ICPPL and feasible. For the rest of the proof we use mathematical induction on the number of iterations j. As before, the term "new sets" will refer to the sets added in $\mathcal{F}_j$ in the jth iteration, i.e. $S_1 \setminus \{x\}$ and $\{x\}$ as defined in line 3.

For $\mathcal{F}_0, \ell_0$ all invariants hold because it is output from algorithm 1 which is an ICPPL. Hence base case is proved. Assume the lemma holds for the $j-1$th iteration. Consider jth iteration.

Case 1: *Invariant I and II*

Case 1.1: *R is not a new set.* If R is in $\mathcal{F}_{j-1}$, then by induction hypothesis this case is trivially proven.

Case 1.2: *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 $\mathcal{F}_j$. Removing a leaf v from path $\ell_{j-1}(S_1)$ results in another path. Moreover, $\{v\}$ is trivially a path. Hence regardless of which new set R is, by definition of ℓ_j, $\ell_j(R)$ is a path. Thus invariant I is proven.
We know $|S_1| = |\ell_{j-1}(S_1)|$, due to induction hypothesis. Therefore $|S_1 \setminus \{x\}| = |\ell_{j-1}(S_1) \setminus \{v\}|$. This is because $x \in S_1$ iff $v \in \ell_{j-1}(S_1)$. If $R = \{x\}$, invariant II is trivially true. Thus invariant II is proven.

Case 2: *Invariant III*

Case 2.1: *R and R' are not new sets.* Trivially true by induction hypothesis.

Case 2.2: *Only one, say R, is a new set.* By definition, $\ell_{j-1}(S_1)$ is the only path with v and S_1 the only set with x in the previous iteration, hence $|R' \cap (S_1 \setminus \{x\})| = |R' \cap S_1|$ and $|\ell_{j-1}(R') \cap (\ell_{j-1}(S_1) \setminus \{v\})| = |\ell_{j-1}(R') \cap \ell_{j-1}(S_1)|$ and $|R' \cap \{x\}| = 0$, $|\ell_{j-1}(R') \cap \{v\}| = 0$. Thus case proven.

Case 2.3: *R and R' are new sets.* By definition, the new sets and their path images in path label l_j are disjoint so $|R \cap R'| = |l_j(R) \cap l_j(R)| = 0$. Thus case proven.

Case 3: *Invariant IV*

Case 3.1: *R, R' and R'' are not new sets.* Trivially true by induction hypothesis.

Case 3.2: *Only one, say R, is a new set.* By the same argument used to prove invariant III, $|R' \cap R'' \cap (S_1 \setminus \{x\})| = |R' \cap R'' \cap S_1|$ and $|\ell_{j-1}(R') \cap \ell_{j-1}(R'') \cap (\ell_{j-1}(S_1) \setminus \{v\})| = |\ell_{j-1}(R') \cap \ell_{j-1}(R'') \cap \ell_{j-1}(S_1)|$. Since R', R'', S_1 are all in $\mathcal{F}_{j-1}$, by induction hypothesis of invariant IV, $|R' \cap R'' \cap S_1| = |\ell_{j-1}(R') \cap \ell_{j-1}(R'') \cap \ell_{j-1}(S_1)|$. Also, $|R' \cap R'' \cap \{x\}| = |\ell_{j-1}(R') \cap \ell_{j-1}(R'') \cap \{v\}| = 0$.

Case 3.3: *At least two of R, R', R'' are new sets.* If two or more of them are not in $\mathcal{F}_{j-1}$, then it can be verified that $|R \cap R' \cap R''| = |\ell_j(R) \cap \ell_j(R') \cap \ell_j(R'')|$ since the new sets in $\mathcal{F}_j$ are disjoint . Thus invariant IV is also proven.

◀ ◀

We have seen two filtering algorithms above - algorithms 1 and 2. We also proved that if the input is indeed feasible, these algorithms do not exit prematurely and succesfully filters the input keeping the ICPPL conditions intact. Using these algorithms we now prove the following theorem.

▶ **Theorem 3.7.** *If $\mathcal{F}$ has an ICPPL ℓ to a tree T, then there exists a hypergraph isomorphism ϕ : supp($\mathcal{F}$) $\rightarrow$ supp($\mathcal{F}^\ell$) such that the ϕ-induced tree path labeling is equal to ℓ, $\ell_\phi = \ell$.*

Proof. This is a contructive proof. We find ϕ part by part by running algorithms 1 and 2 one after the other in a loop. After each iteration we calculate an exclusive subset of the bijection ϕ, namely that which involves all the leaves of the tree in that iteration. Then all the leaves are pruned off the tree before the next iteration. The loop terminates when the pruned tree becomes a single path after which ICPIA algorithm is used to find the final subset (interval assignment) that exhausts ϕ. This is the brief outline of the algorithm and now we describe it in detail below.

First, the given ICPPL $(\mathcal{F}, \ell)$ and tree T are given as input to Algorithm 1. This yields a "filtered" ICPPL as the output which is input to Algorithm 2. Let the output of Algorithm 2 be $(\mathcal{F}', \ell')$. We define a bijection $\phi_1 : Y_1 \rightarrow L_1$ where $Y_1 \subseteq$ *supp*$(\mathcal{F})$ and $V_1 = \{v \mid v \text{ is a leaf in T}\}$. It can be observed that the output of Algorithm 2 is a set of path assignments to sets and one-to-one assignment of elements of U to each leaf of T. These are defined below as ℓ_1 and ϕ_1 respectively.

$$\ell_1(S) = \ell'(S) \qquad \text{when } \ell'(S) \text{ has non leaf vertices}$$
$$\phi_1(x) = v \qquad \text{when } \ell'(S) \text{ has only a leaf, and } v \in \ell'(S), x \in S$$

Consider the tree T_1 which is isomorphic to $T[V(T) \setminus V_0]$, i.e. it is T with all its leaves removed. Let U_1 be the universe of the subsystem that is not mapped to a leaf, i.e. $U_1 = \mathit{supp}(\mathcal{F}) \setminus \{x \mid x = \ell_1^{-1}(v), v \in V_1\}$.

Let $\mathcal{F}_1$ be the set system induced by $\mathcal{F}'$ on universe U_1. Clearly, now we have a subproblem of finding the hypergraph isomorphism for $(\mathcal{F}_1, \ell_1)$ with tree T_1. Now we repeat Algorithm 1 and Algorithm 2 on $(\mathcal{F}_1, \ell_1)$ with tree T_1. As before we define l_2 in terms of l_1, ϕ_2 in terms of leaves of T_1, prune the tree T_1 to get T_2 and so on. Thus in the ith iteration, T_i is the pruned tree, ℓ_i is a feasible path labeling to $\mathcal{F}_i$ if $(\mathcal{F}_{i-1}, \ell_{i-1})$ is feasible, ϕ_i is the leaf labeling of leaves of T_{i-1}. Continue this until some dth iteration for the smallest value d such that T_d is a path. From the lemma 3.4 and 3.6 we know that $(\mathcal{F}_d, \ell_d)$ is an ICPPL. We also know that the special case of ICPPL when the tree is a path is the interval assignment (ICPIA) problem. We now run the ICPIA algorithms [17] on $(\mathcal{F}_d, \ell_d)$.

It is true that T_d is not precisely an interval in the sense of consecutive integers because they could be arbitrarily named nodes a tree. However, it is easy to see that the nodes of T_d can be ordered from left to right and ranked to get intervals I_i for every path $S_i \in \mathcal{F}_d$ as follows. $I_i = \{[l, r] \mid l =$ the lowest rank of the nodes in $\ell_d(S_i)$, $r = l + |\ell_d(S_i)| - 1\}$, where $S_i \in \mathcal{F}_d$. We define an interval assignment $\mathcal{A} = \{(S_i, I_i) \mid S_i \in \mathcal{F}_d\}$ which is an ICPIA and also in the format ICPIA algorithm requires. The ICPIA algorithms give us $\mathcal{A}$ and the bijection $\phi_{d+1} : U_d \to T_d$. The bijection $\phi : U \to V(T)$ defined as follows is the bijection for the ICPPL.

$\phi(x) = \phi_i(x)$ where x is in the domain of $\phi_i, i \in [d+1]$

It can be verified that ϕ is a bijection on $supp(\mathcal{F})$ into $V(T)$ which is the path hypergraph isomorphism between $\mathcal{F}$ and $\mathcal{F}^\ell$ such that $\ell_\phi = \ell$. Thus the theorem is proven. ◀ ◀

4 Finding tree path labeling from k-subdivided stars

As we saw earlier, the algorithm 7 for the problem of tree path labeling to a path system in a general tree is not polynomial time. Algorithm 7 line 4 leaves an unsolved problem in the main ICPPL algorithm where ICPPL needs to be found out for the mub of each partition X_i i.e, X_{i0} on subtree T_i. Essentially this is the problem of finding a path labeling to an overlap component of $\mathcal{H}_\mathcal{F}$ from a subtree of T. When the subtrees are restricted to a smaller class, namely k-subdivided stars, we have an algorithm which has better time complexity.
Following the notation in the previous section, the subtree assigned to the partition X_i is T_i. We saw that it is sufficient to find the ICPPL for X_{i0} from T_i to find the ICPPL for the set subsystems corresponding to the whole partition X_i. Hence in this section, we are interested in the mub X_{i0} of partition X_i. Let the set subsystem corresponding to X_{i0} be $\mathcal{O}_{i0}$. For ease of notation and due to our focus here being only on the overlap subsystem of the mub and the assigned subtree, we will drop the subscripts, and call $\mathcal{O}$ and T as the set system and tree (rather than set subsystem and subtree) respectively.
Note that here we assume the partitioning of the tree T into subtrees $\{T_i \mid T_i$ assigned to X_i, $T_i \subseteq T$, $i \in [t]\}$ has been done. The problem of partitioning T is a problem that needs to be addressed separately and is not covered in this paper at the moment.
We generalize the interval assignment algorithm for an overlap component from a prime matrix in [17] (algorithm 4 in their paper) to find tree path labeling for overlap component $\mathcal{O}$. The tree T is a k-subdivided star. The vertex r is the center of the star.
The outline of the algorithm is as follows. Notice that the path between a leaf and the center vertex has the property that none of the vertices except the center has degree greater than 2. Thus each ray excluding the center can be considered as independent intervals. So we begin by labeling of hyperedges to paths that have vertices from a single ray only and the center vertex. Clearly this can be done using ICPIA alone. This is done for each ray one after another till a condition for a blocking hyperedge is reached for each ray which is described below. This part of the algorithm is called the *initialization of path labeling*.
When considering labeling from any particular ray, we will reach a point in the algorithm were we cannot proceed further with ICPIA alone because the overlap properties of the hyperedge will require a path that will cross the center of the star to another ray and ICPIA cannot tell us which ray that would be. Such a hyperedge is called *blocking hyperedge* of that ray. At this point we make the following observation about the classification of the hyperedges in $\mathcal{O}$.

i *Type 0/ labeled hyperedges*: The hyperedges that have been labeled.
ii *Type 1/ unlabeled non-overlapping hyperedges*: The hyperedges that are either contained or disjoint from type 0 hyperedges.

iii *Type 2/ unlabeled overlaping hyperedges*: The hyperedges that overlap with at least one labeled hyperedge, say H, but cannot be labeled to a path in the same ray as $\ell(H)$ alone. It requires verices from another ray also in its labeling. A *blocking hyperedge* is one of this kind which is encountered in each iteration of the initialization of rays algorithm.

Since $\mathcal{O}$ is an overlap component, the type 1 hyperedges overlap with some type 2 hyperedge and can be handled after type 2 hyperedges. Note that in the algorithm outlined above, we find a single blocking hyperedge and it is a type 2 hyperedge, per ray. Consider a ray $R_i = \{v \mid v \in V(T), v \text{ is in } i\text{th ray or is the center}\}$ and its corresponding blocking hyperedge B_i. Now we try to make a partial path labeling such that for every $i \in [l]$. We partition the blocking hyperedge into two subsets $B_i = B_i' \cup B_i''$ such that B_i', B_i'' map to paths P_i', P_i'' respectively which are defined as follows.

$$
\begin{aligned}
P_i' \quad & \subseteq R_i \text{ such that } r \in P_i' \\
& \text{and } |P_i'| = k + 2 - |supp(\{P \mid P \text{ is a path from } R_i \text{ assigned to type 0 hyperedges}\})| \\
P_i'' \quad & \in \{P_{i,j} \mid j \in [l], j \neq i\} \\
& \text{where } P_{i,j} = \{v_{j,p} \mid v_{j,0} \text{ is adjacent to } r \text{ on } R_j, \\
& \text{for all } 0 < p \leq |B_i \setminus P_i'| - 1, v_{j,p} \text{ is adjacent to } v_{j,p-1}\} \\
P_i' \cup P_i'' \quad & \text{ is a path in } T
\end{aligned}
$$

The path P_i' is obvious and the following procedure is used to find P_i''. It is clear that a hyperedge cannot be blocking more than two rays, since a path cannot have vertices from more than two rays.

▶ Observation 1. If the blocking hyperedge B_a of ray R_a is also the blocking hyperedge for another ray R_b (i.e. $B_a = B_b$), then clearly $P_a'' = P_{a,b}$ (and $P_b'' = P_{b,a}$).

▶ Observation 2. If B_a does not block any other rays of the star other than R_a, then we find that it must intersect with exactly one other blocking hyperedge, say B_b. Once we find the second ray, then clearly $P_a'' = P_{a,b}$.

Note that $P_b'' \neq P_{b,a}$ in observation 2 else it would have been covered in observation 1. Now we continue to find new blocking hyperedges on all rays until the path labeling is complete. The algorithm is formally described as follows. Algorithm 3 is the main algorithm which uses algorithms 4, 5 and 6 as subroutines. The function $dist(u, v)$ returns the number of vertices between the vertices u and v on the path that connects them (including u and v).

5 Finding an assignment of tree paths to a set system

observation: T[V-c] is a collection of independent paths. c is the center.

In the previous section we have shown that the problem of finding a Tree Path Labeling to an input $(\mathcal{F}, T)$ is equivalent to finding an ICPPL to $\mathcal{F}$ in tree T. In this section we characterize those set systems that have an ICPPL in a given tree. As a consequence of this characterization we identify two sub-problems that must be solved to obtain an ICPPL. We do not solve these subproblems but use them as blackboxes to describe the rest of the algorithm. In the next section, we solve one of these subproblems for a smaller class of trees, k-subdivided stars.

A set system can be concisely represented by a binary matrix where the row indices denote the universe of the set system and the column indices denote each of the sets. Let the binary matrix be M with order $n \times m$, the set system be $\mathcal{F} = \{S_i \mid i \in [m]\}$, universe of set system $U = \{x_i \mid i \in [n]\}$. We say M represents $\mathcal{F}$, if (i, j)th element of M, $M_{ij} = 1$ iff $x_i \in S_j$. If $\mathcal{F}$ has a feasible tree path labeling $\ell : \mathcal{F} \to \mathcal{P}$, where $\mathcal{P}$ is a set of paths from a given tree T

Algorithm 3 Algorithm (main subroutine) to find an ICPPL ℓ for an overlap component $\mathcal{O}$ from k-subdivided star graph T: $overlap\_ICPPL\_l\_leaves\_symstarlike3(\mathcal{O}, T)$

```
1: 𝓛   /* 𝓛 ⊆ 𝓞 is a global variable for the set subsystem that has a path labeling
   so far.  It is the domain of the feasible path labeling ℓ at any point in the
   algorithm.  */
2: ℓ   /* ℓ : 𝓛 → 𝓟, is a global variable representing a feasible path labeling of 𝓛
   to some path system 𝓟 of T.  It is the partial feasible path labeling of 𝓞 at
   any point in the algorithm.  */
3: initialize_rays(𝓞, T)    /* Call algorithm 4 for initialization of rays.  This is
   when a hyperedge is assigned to a path with the ray's leaf.  */
4: while 𝓛 ≠ 𝓞 do
5:   saturate_rays_and_find_blocking_hyperedges(𝓞, T)   /* Saturate all rays of
     T by using algorithm 5.  This subroutine also finds the blocking hyperedge 𝓑_i
     of each ray i.  A blocking hyperedge is one that needs to be labeled to a path
     that has vertices from exactly two rays.  */
6:   partial_path_labeling_of_blocking_hyperedges(𝓞, T)  /* Find path labeling of
     blocking hyperedges by using algorithm 6.  This subroutine finds the part of
     the blocked hyperedge's path label that comes from the second ray.  */
7: end while
```

then we say its corresponding matrix M has an ICPPL. Conversely, we say that a matrix M has an ICPPL if there exists an ICPPL ℓ as defined above.

We now consider the overlap graph of $\mathcal{F}$. The usage of overlap graph to decompose the problem of consecutive ones testing was first introduced by [5]. They showed that a binary matrix or its corresponding set system has the COP iff each connected component of the overlap graph (the sets corresponding to this component or its corresponding submatrix) has the COP. The same approach is also described in [11, 17]. We use this idea to decompose M and construct a partial order on the components similarly. The resulting structural observations are used to come up with the required algorithm for tree path assignment.

A prime sub-matrix of M is defined as the matrix formed by a set of columns of M which correspond to a connected component of the graph $\mathbb{O}(\mathcal{F})$. Let us denote the prime sub-matrices by $M_1, \ldots, M_p$ each corresponding to one of the p components of $\mathbb{O}(\mathcal{F})$. Clearly, two distinct matrices have a distinct set of columns. Let $col(M_i)$ be the set of columns in the sub-matrix M_i. The support of a prime sub-matrix M_i is defined as $supp(M_i) = \bigcup_{j \in col(M_i)} S_j$. Note that for each i, $supp(M_i) \subseteq U$. For a set of prime sub-matrices X we define $supp(X) = \bigcup_{M \in X} supp(M)$.

Consider the relation $\preccurlyeq$ on the prime sub-matrices $M_1, \ldots, M_p$ defined as follows:

$$\{(M_i, M_j) \mid \text{ a set } S \in M_i \text{ is contained in a set } S' \in M_j\} \cup \{(M_i, M_i) \mid i \in [p]\}$$

This relation is the same as that defined in [17]. The prime submatrices and the above relation can be defined for any set system. We will use this structure of prime submatrices to present our results on an ICPPL for a set system $\mathcal{F}$. Recall the following lemmas and theorem that $\preccurlyeq$ is a partial order, from [17].

► **Lemma 5.1.** *Let $(M_i, M_j) \in \preccurlyeq$. Then there is a set $S' \in M_j$ such that for each $S \in M_i$, $S \subseteq S'$.*

► **Lemma 5.2.** *For each pair of prime sub-matrices, either $(M_i, M_j) \notin \preccurlyeq$ or $(M_j, M_i) \notin \preccurlyeq$.*

► **Lemma 5.3.** *If $(M_i, M_j) \in \preccurlyeq$ and $(M_j, M_k) \in \preccurlyeq$, then $(M_i, M_k) \in \preccurlyeq$.*

Algorithm 4 *initialize_rays*$(\mathcal{O}, T)$

Let $\{v_i \mid i \in [l], l$ is number of leaves of $T\}$ `/* Also note` $k+2$ `is the length of the path from the center to any leaf since` T `is` k`-subdivided star. */`
$\mathcal{K} \leftarrow \{H \mid H \in \mathcal{O}, N(H)$ in $\mathcal{O}$ is a clique $\}$ `/* Local variable to hold the marginal hyperedges. A marginal hyperedge is one that has exactly one inclusion chain of interections with every set it overlaps with, i.e., its neighbours in the overlap graph form a clique. */`
for every inclusion chain $C \subseteq \mathcal{K}$ **do**
  Remove from $\mathcal{K}$ all sets in C except the set $H_{C-icpia-max}$ which is the set closest to the maximal inclusion set H_{C-max} such that $|H_{C-icpia-max}| \leq k+2$.
end for
if $|\mathcal{K}| > l$ **then**
  Exit. `/* No labeling possible since` $\mathcal{O}$ `is an overlap component and` T `does not have enough rays. */`
end if
`/*` $H_{C-icpia-max}$ `does not exist for at least one ray. Labeling could still be possible because` H_{C-max} `could be a viable blocking hyperedge itself. Hence proceed. */`
$i \leftarrow 0$
for every hyperedge $H \in \mathcal{K}$ **do**
  $i \leftarrow i+1$
  $\ell(H) \leftarrow P_i$ where P_i is the path in T containing leaf v_i such that $|P_i| = |H|$.
  $\mathcal{L} \leftarrow \mathcal{L} \cup \{H\}$
end for
Return the number of initialized rays, i.

▶ **Lemma 5.4.** *If $(M_i, M_j) \in \preccurlyeq$ and $(M_i, M_k) \in \preccurlyeq$, then either $(M_j, M_k) \in \preccurlyeq$ or $(M_k, M_j) \in \preccurlyeq$.*

▶ **Theorem 5.5.** *$\preccurlyeq$ is a partial order on the set of prime sub-matrices of M. Further, it uniquely partitions the prime sub-matrices of M such that on each set in the partition $\preccurlyeq$ induces a total order.*

For the purposes of this paper, we refine the total order mentioned in Theorem 5.5. We do this by identifying an in-tree rooted at each maximal upper bound under $\preccurlyeq$. Each of these in-trees will be on disjoint vertex sets, which in this case would be disjoint sets of prime-submatrices. The in-trees are specified by selecting the appropriate edges from the Hasse diagram associated with $\preccurlyeq$. Let $\mathcal{I}$ be the following set:

$$\mathcal{I} = \{(M_i, M_j) \in \preccurlyeq \mid \nexists M_k \text{ s.t. } M_i \preccurlyeq M_k, M_k \preccurlyeq M_j\} \cup \{(M_i, M_i), i \in [p]\}$$

▶ **Theorem 5.6.** *Consider the directed graph X whose vertices correspond to the prime sub-matrices, and the edges are given by $\mathcal{I}$. Then, X is a vertex disjoint collection of in-trees and the root of each in-tree is a maximal upper bound in $\preccurlyeq$.*

Proof. To observe that X is a collection of in-trees, we observe that for vertices corresponding to maximal upper bounds, no out-going edge is present in X. Secondly, for each other element, exactly one out-going edge is chosen (due to lemma 5.4 and the condition in set $\mathcal{I}$ definition), and for the minimal lower bound, there is no in-coming edge. Consequently, X is acyclic, and since each vertex has at most one edge leaving it, it follows that X is a collection of in-trees, and for each in-tree, the root is a maximal upper bound in $\preccurlyeq$. Hence the theorem. ◀ ◀

Algorithm 5 *saturate_rays_and_find_blocking_hyperedge*$(\mathcal{O}, T)$

Variable $\mathcal{B}_i$ shall store the blocking hyperedge for ith ray. Init variables: for every $i \in [l]$, $\mathcal{B}_i \leftarrow \emptyset$
Let $\mathcal{L}_i \subseteq \mathcal{L}$ containing hyperedges labeled to ith ray i.e. $\mathcal{L}_i = \{H \mid \ell(H) \subseteq R_i\}$
for every $i \in [l]$ **do**
  `/* for each ray */`
  if $L_i = \emptyset$ **then**
    `/* Due to the condition 8 in algorithm 4 */`
    $\mathcal{K} \leftarrow \{H \mid H \in \mathcal{O} \setminus \mathcal{L}$ s.t. neighbours of H in the overlap graph of $\mathcal{O}$ form a clique$\}$
    Pick an inclusion chain $C \subseteq \mathcal{K}$ and let H_{C-max} be the maximal inclusion hyperedge in C.
    $\mathcal{B}_i \leftarrow H_{C-max}$ `/* Since` $H_{C-max} \in \mathcal{L}$`, and due to earlier subroutines,` $|H_{C-max}| > k+2$ `*/`
  end if
  while $\mathcal{B}_i = \emptyset$ and there exists $H \in \mathcal{O} \setminus \mathcal{L}$, such that H overlaps with some hyperedge $H' \in \mathcal{L}_i$ **do**
    $d \leftarrow |H \setminus H'|$
    Let u be the end vertex of the path $\ell(H')$ that is closer to the center r, than its other end vertex
    if $d \leq dist(u, r) + 1$ **then**
      Use ICPIA to assign path $P \subseteq R_i$ to H
      $\ell(H) \leftarrow P$ `/* Update variables */`
      $\mathcal{L} \leftarrow \mathcal{L} \cup \{H\}$, $\mathcal{L}_i \leftarrow \mathcal{L}_i \cup \{H\}$
    else
      $\mathcal{B}_i \leftarrow H$
      Continue `/* Found the blocking hyperedge for this ray; move on to next ray */`
    end if
  end while
end for

Let the partition of X given by Theorem 5.6 be $\{X_1, \ldots, X_r\}$. Further, each in-tree itself can be layered based on the distance from the root. The root is considered to be at level zero. For $j \geq 0$, Let $X_{i,j}$ denote the set of prime matrices in level j of in-tree X_i.

▶ **Lemma 5.7.** *Let M be a matrix and let X be the directed graph whose vertices are in correspondence with the prime submatrices of M. Further let $\{X_1, \ldots, X_r\}$ be the partition of X into in-trees as defined above. Then, matrix M has an ICPPL in tree T iff T can be partitioned into vertex disjoint subtrees $\{T_1, T_2, \ldots T_r\}$ such that, for each $1 \leq i \leq r$, the set of prime sub-matrices corresponding to vertices in X_i has an ICPPL in T_i.*

Proof. Let us consider the reverse direction first. Let us assume that T can be partitioned into $T_1, \ldots, T_r$ such that for each $1 \leq i \leq r$, the set of prime sub-matrices corresponding to vertices in X_i has an ICPPL in T_i. It is clear from the properties of the partial order $\preccurlyeq$ that these ICPPLs naturally yield an ICPPL of M in T. The main property used in this inference is that for each $1 \leq i \neq j \leq r$, $supp(X_i) \cap supp(X_j) = \emptyset$.
To prove the forward direction, we show that if M has an ICPPL, say $\mathcal{A}$, in T, then there exists a partition of T into vertex disjoint subtree $T_1, \ldots, T_r$ such that for each $1 \leq i \leq r$, the set of prime sub-matrices corresponding to vertices in X_i has an ICPPL in T_i. For each

Algorithm 6 *partial_path_labeling_of_blocking_hyperedges*$(\mathcal{O}, T)$

/* Process equal blocking hyperedges. At this point for all $i \in [l]$, $\mathcal{B}_i \neq \emptyset$. */
for every $i \in [l], \mathcal{B}_i \neq \emptyset$ **do**
  for every $j \in [l]$ **do**
    if $\mathcal{B}_i = \mathcal{B}_j$ **then**
      /* Blocking hyperedges of ith and jth rays are same */
      Let $H \leftarrow \mathcal{B}_i$ /* or $\mathcal{B}_j$ */
      Find path P on the path $R_i \cup R_j$ to assign to H using ICPIA
      $\ell(H) \leftarrow P$
      $\mathcal{L} \leftarrow \mathcal{L} \cup \{H\}$
      $\mathcal{B}_i \leftarrow \emptyset$, $\mathcal{B}_j \leftarrow \emptyset$ /* Reset blocking hyperedges for ith and jth rays */
    end if
  end for
end for
/* Process intersecting blocking hyperedges */
for every $i \in [l], \mathcal{B}_i \neq \emptyset$ **do**
  for every $j \in [l]$ **do**
    if $\mathcal{B}_i \cap (supp(\mathcal{L}_j \cup \mathcal{B}_j)) \neq \emptyset$ **then**
      /* Blocking hyperedge of ith ray intersects with hyperedge associated with jth ray */
      Find interval P_i for $\mathcal{B}_i$, on the path $R_i \cup R_j$ that satisfies ICPIA.
      $\ell(\mathcal{B}_i) \leftarrow P_i$
      $\mathcal{B}_i \leftarrow \emptyset$
      $\mathcal{L} \leftarrow \mathcal{L} \cup \{\mathcal{B}_i\}$
    end if
  end for
end for

$1 \leq i \leq r$, we define based on $\mathcal{A}$ a subtree T_i corresponding to X_i. We then argue that the trees thus defined are vertex disjoint, and complete the proof. Consider X_i and consider the prime sub-matrix in $X_{i,0}$. Consider the paths assigned under $\mathcal{A}$ to the sets in the prime sub-matrix in $X_{i,0}$. Since the component in G_f corresponding to this matrix is a connected component, it follows that union of paths assigned to this prime-submatrix is a subtree of T. We call this sub-tree T_i. All other prime-submatrices in X_i are assigned paths in T_i since $\mathcal{A}$ is an ICPPL, and the support of other prime sub-matrices in X_i are contained in the support of the matrix in $X_{i,0}$. Secondly, for each $1 \leq i \neq j \leq r$, $supp(X_i) \cap supp(X_j) = \emptyset$, and since $\mathcal{A}$ is an ICPPL, it follows that T_i and T_j are vertex disjoint. Finally, since $|U| = |V(T)|$, it follows that $T_1, \ldots, T_r$ is a partition of T into vertex disjoint sub-trees such that for each $1 \leq i \leq r$, the set of matrices corresponding to nodes in X_i has an ICPPL in T_i. Hence the lemma. ◀ ◀

The essence of the following lemma is that an ICPPL only needs to be assigned to the prime sub-matrix corresponding to the root of each in-tree, and all the other prime sub-matrices only need to have an Intersection Cardinality Preserving *Interval* Assignments (ICPIA). Recall, an ICPIA is an assignment of intervals to sets such that the cardinality of an assigned interval is same as the cardinality of the interval, and the cardinality of intersection of any two sets is same as the cardinality of the intersection of the corresponding intervals. It is

shown in [17] that the existence of an ICPIA is a necessary and sufficient condition for a matrix to have the COP.
We present the pseudo-code to test if M has an ICPPL in T.

▶ **Lemma 5.8.** *Let M be a matrix and let X be the directed graph whose vertices are in correspondence with the prime submatrices of M. Further let $\{X_1, \ldots, X_r\}$ be the partition of X into in-trees as defined earlier in this section. Let T be the given tree and let $\{T_1, \ldots, T_r\}$ be a given partition of T into vertex disjoint sub-trees. Then, for each $1 \leq i \leq r$, the set of matrices corresponding to vertices of X_i has an ICPPL in T_i if and only if the matrix in $X_{i,0}$ has an ICPPL in T_i and all other matrices in X_i have an* ***ICPIA*** *on their path in T_i.*

Proof. The proof is based on the following fact - $\preccurlyeq$ is a partial order and X is a digraph which is the disjoint union of in-trees. Each edge in the in-tree is a containment relationship among the supports of the corresonding sub-matrices. Therefore, any ICPPL to a prime sub-matrix that is not the root is contained in a path assigned to the sets in the parent matrix. Consequently, any ICPPL to the prime sub-matrix that is not at the root is an ICPIA, and any ICPIA can be used to construct an ICPPL to the matrices corresponding to nodes in X_i provided the matrix in the root has an ICPPL in T_i. Hence the lemma. ◀ ◀

Lemma 5.7 and Lemma 5.8 point out two algorithmic challenges in finding an ICPPL for a given set system $\mathcal{F}$ in a tree T. Given $\mathcal{F}$, finding X and its partition $\{X_1, \ldots, X_r\}$ into in-trees can be done in polynomial time. On the other hand, as per lemma 5.7 we need to parition T into vertex disjoint sub-trees $\{T_1, \ldots, T_r\}$ such that for each i, the set of matrices corresponding to nodes in X_i have an ICPPL in T_i. This seems to be a challenging step, and it must be remarked that this step is easy when T itself is a path, as each individual T_i would be sub-paths. The second algorithmic challenge is identified by lemma 5.8 which is to assign an ICPPL from a given tree to the matrix associated with the root node of X_i.

Algorithm 7 Algorithm to find an ICPPL for a matrix M on tree T: $main\_ICPPL(M,T)$

Identify the prime sub-matrices. This is done by constructing the strict overlap graph and identifying the connected components. Each connected component yields a prime sub-matrix.
Construct the partial order $\preccurlyeq$ on the set of prime sub-matrices.
Construct the partition $X_1, \ldots, X_r$ of the prime sub-matrices induced by $\preccurlyeq$
For each $1 \leq i \leq r$, Check if all matrices except those in $X_{i,0}$ has an ICPIA. If a matrix does not have ICPIA exit with a negative answer. To check for the existence of ICPIA, use the result in [17].
Find a partition of $T_1, \ldots, T_r$ such that matrices in $X_{i,0}$ has an ICPPL in T_i. If not such partition exists, exit with negative answer.

---

6 Acknowlegements

We thank the anonymous referees of the WG 2011 committee and our colleagues who helped us with the readability of this document.

References

1 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.

2 J. R. S. Blair and B. Peyton. An introduction to chordal graphs and clique trees. Technical report, Oak Ridge National Laboratory, Oak Ridge, Tennessee 37831, 1992.

3 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.

4 Andreas Brandstädt, Van Bang Le, and Jeremy P. Spinrad. *Graph classes: a survey*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1999.

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

6 Fanica Gavril. A recognition algorithm for the intersection graphs of paths in trees. *Discrete Mathematics*, 23(3):211 – 227, 1978.

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

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

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

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

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

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

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

14 P. S. Kumar and C. E. Veni Madhavan. Clique tree generalization and new subclasses of chordal graphs. *Discrete Applied Mathematics*, 117:109–131, 2002.

15 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.

16 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.

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

18 Barry W. Peyton, Alex Pothen, and Xiaoqing Yuan. A clique tree algorithm for partitioning a chordal graph into transitive subgraphs. Technical report, Old Dominion University, Norfolk, VA, USA, 1994.

19 Peter L. Renz. Intersection representations of graphs by arcs. *Pacific J. Math.*, 34(2):501–510, 1970.

20 Alejandro A. Schaffer. A faster algorithm to recognize undirected path graphs. *Discrete Applied Mathematics*, 43:261–295, 1993.

A Detailed proofs

Proof of Lemma 3.1

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

Proof of Lemma 3.2

Proof. *Case 1:* Consider the path $P = P_3 \cap P_4$ (intersection of two paths is a path). Suppose in this case, P does not intersect with $P_1 \setminus P_2$, i.e. $P \cap (P_1 \setminus P_2) = \emptyset$. Then $P \cap P_1 \cap P_2 = P \cap P_2$. Similarly, if $P \cap (P_2 \setminus P_1) = \emptyset$, $P \cap P_1 \cap P_2 = P \cap P_1$. Thus it is clear that if the intersection of any two paths does not intersect with any of the set differences of the remaining two paths, the claim in the lemma is true.

Case 2: The other possibilty is the compliment of the previous case which is as follows. So let us assume that the intersection of any two paths intersects with both the set differences of the other two. First let us consider $P \cap (P_1 \setminus P_2) \neq \emptyset$ and $P \cap (P_2 \setminus P_1) \neq \emptyset$, where $P = P_3 \cap P_4$. Since P_1 and P_2 share a leaf, there is exactly one vertex at which they branch off from the path $P_1 \cap P_2$ into two paths $P_1 \setminus P_2$ and $P_2 \setminus P_1$. Let this vertex be v. It is clear that if path $P_3 \cap P_4$, must intersect with paths $P_1 \setminus P_2$ and $P_2 \setminus P_1$, it must contain v since these are paths from a tree. Moreover, $P_3 \cap P_4$ intersects with $P_1 \cap P_2$ at exactly v and only at v which means that $P_1 \cap P_2$ does not intersect with $P_3 \setminus P_4$ or $P_4 \setminus P_3$ which contradicts the assumption of this case. Thus this case cannot occur and case 1 is the only possible scenario.

Thus lemma is proven. ◀ ◀

Proof of Lemma 3.4

Proof.

Case 1: *Invariant I and II*

Case 1.2: *R is a new set:*

If $R = S_1 \cap S_2$, $|R| = |S_1 \cap S_2| = |\ell_{j-1}(S_1) \cap \ell_{j-1}(S_2)|$[1] $= |\ell_j(S_1 \cap S_2)|$[2] $= |\ell_j(R)|$

If $R = S_1 \setminus S_2$, $|R| = |S_1 \setminus S_2| = |S_1| - |S_1 \cap S_2| = |\ell_{j-1}(S_1)| - |\ell_{j-1}(S_1) \cap \ell_{j-1}(S_2)|$[3] $= |\ell_{j-1}(S_1) \setminus \ell_{j-1}(S_2)| = |\ell_j(S_1 \setminus S_2)|$[4] $= |\ell_j(R)|$.

Thus Invariant II proven.

Case 2: *Invariant III*

Case 2.2: *Only R is a new set:*

If $R = S_1 \cap S_2$, $|R \cap R'| = |S_1 \cap S_2 \cap R'| = |\ell_{j-1}(S_1) \cap \ell_{j-1}(S_2) \cap \ell_{j-1}(R')|$[5] $= |\ell_j(S_1 \cap S_2) \cap \ell_j(R')|$[6] $= |\ell_j(R) \cap \ell_j(R')|$

[1] Inv III hypothesis
[2] ℓ_j definition
[3] Inv II and III hypothesis
[4] ℓ_j definition
[5] Inv IV hypothesis
[6] ℓ_j definition. Note that R' is not a new set

If $R = S_1 \setminus S_2$, $|R \cap R'| = |(S_1 \setminus S_2) \cap R'| = |(\ell_{j-1}(S_1) \setminus \ell_{j-1}(S_2)) \cap \ell_{j-1}(R')|$[7] $= |\ell_j(R) \cap \ell_j(R')|$[8]
Thus Invariant III proven.

Case 3: *Invariant IV*

Case 3.2: *Only R is a new set:*
If $R = S_1 \cap S_2$, Consider, $|\ell_{j-1}(S_1) \cap \ell_{j-1}(S_2) \cap \ell_{j-1}(R') \cap \ell_{j-1}(R'')|$. We know from lemma 3.2 that the intersection of these four paths is same as the intersection of three distinct paths among the four. Let us call these four paths P_1, P_2, P_3, P_4 and without loss of generality, let it be that $\cap_{i=1}^{4} P_i = \cap_{i=1}^{3} P_i$. Further $| \cap_{i=1}^{4} P_i| = |S_1 \cap S_2 \cap R'|$ by the invariant IV of the induction hypothesis. Therefore, it follows that $| \cap_{i=1}^{4} P_i| \geq |S_1 \cap S_2 \cap R' \cap R''|$.
If $R = S_1 \setminus S_2$, a similar argument using Lemma 3.1 and the induction hypothesis completes the proof of this case.
Thus Invariant IV proven.

◀

7 Lemma 3.1
8 ℓ_j definition. Note R' is not a new set