Tree Path Assignments to Sets-A Generalization of the Consecutive Ones Property

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

Indian Institute of Technology Madras, Chennai, swamy@cse.iitm.ernet.in,
Indian Institute of Technology Madras, Chennai, anjuzabil@gmail.com

Abstract. We consider the following constraint satisfaction problem: Given a set $\mathcal{F}$ of subsets of a finite set U of cardinality n, a tree T on n vertices, and an assignment of paths from T to each of the subsets, does there exist a bijection $f : U \rightarrow \{v_1, \ldots, v_n\}$ such that for each element of $\mathcal{F}$, its image under f is same as the path assigned to it? A path assignment to a given set of subsets is called *feasible* if there exists such a bijection. In this paper, we characterize feasible path assignments to a given set of subsets and a tree. This result is a natural generalization of results on matrices with the Consecutive Ones Property(COP) which can be viewed as a special instance of the problem in which the given tree is a path on n vertices. We also present a characterization of set systems and trees which have a feasible path assignment. We also show that testing for a feasible path assignment is isomorphism-complete. On the other hand, it is known that if the given tree is a path a feasible assignment can be found in polynomial time, and we observe that it can actually be done in logspace.

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[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]. Recognition of COP is polynomial time solvable by several algorithms. PQ trees[BL76], variations of PQ trees[MM96, Hsu01, Hsu02, McC04], ICPIA[NS09] 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 are intervals. Indeed the problem now becomes finding interval assignments to the given set system [NS09] with a single permutation of the universe (set of row indices) which permutes each set to its interval. The result in [NS09] characterize interval assignments to the sets which can be obtained from a single permutation of the rows. They show that for each set, the interval cardinality 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, it is shown that it is indeed sufficient. Such an interval assignment is called an Intersection Cardinality Preserving Interval Assignment (ICPIA). The idea of decomposing a given 0-1 matrix into prime matrices is then taken from [Hsu02] 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 assignments to a set system which is the topic of this paper. 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 Assignment 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 [NS09] to characterize feasible path assignments. We show that for a given set system, 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 3 sets (not necessarily distinct) is same as the intersection cardinality of the paths assigned to them. This characterization is proved constructively and it gives a natural data structure that stores all the relevant bijections between U and $V(T)$. Further, it also gives an efficient algorithm to test if a path assignment to the sets is feasible. This also naturally generalizes the result in [NS09].

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 this subclass contains 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[Gol04]. 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[Ren70, Gav78, BP92]. Certain format of these trees are called clique trees[PPY94] of the graph which is a compact representation of the chordal graph. There has also been work done on the generalization of clique trees to clique hypergraphs[KM02]. If the chordal graph can be represented as the intersection graph of paths in a tree, then the graph is called path graph[Gol04]. Therefore, it is clear that if there is a bijection from U to $V(T)$ such that the sets map to paths, then the intersection graph of the set system is indeed a path graph. This is, however, only a necessary condition and can be checked efficiently, as path graph recognition is polynomial solvable[Gav78, Sch93]. 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. This connection indeed suggests that our problem is indeed as hard as path graph isomorphism. Further path graph isomorphism is known be isomorphism-complete, see for example [KKLV10]. In the second part of this paper, we decompose our search for a bijection between U and $V(T)$ into subproblems. Each subproblem is on a set system in which for each set, there is another set in the set system with which the intersection is *strict*- there is a non-empty intersection, but neither is contained in the other. This is in the spirit of results in [Hsu02, NS09] where to test for the COP in a given matrix, the COP problem is solved on an equivalent set of prime matrices. Our decomposition localizes the challenge of path graph isomorphism to two problems.

Finally, we show that Tree Path Assignment is isomorphism-complete. We also point out Consecutive Ones Testing is in Logspace from two different results in the literature [KKLV10, McC04]. To the best of our knowledge this observation has not been made earlier.

Roadmap: In Section 2 we present the necessary preliminaries, in Section 3 we present our characterization of feasible tree path assignments, and in Section 4 we present the characterizing subproblems for finding a bijection between U and $V(T)$ such that sets map to tree paths. Finally, in Section 5 we conclude by showing that Tree Path Assignment is GI-Complete, and also observe that Consecutive Ones Testing is in Logspace.

2 Preliminaries

In this paper, the collection $\mathcal{F} = \{S \mid S \subseteq U, S \neq \emptyset\}$ is a *set system* of a universe U with $|U| = n$. Moreoever, a set system is assumed to "cover" the universe, i.e. $\bigcup_{S \in \mathcal{F}} S = U$ and the set system is not a multiset (for all pairs of sets $S_1, S_2 \in \mathcal{F}$, $S_1 \neq S_2$).

The *support* of a set system $\mathcal{F}$ denoted by *supp*($\mathcal{F}$) is the union of all the sets in $\mathcal{F}$. Formally, *supp*($\mathcal{F}$) = $\bigcup_{S \in \mathcal{F}} S$.

To state in simple terms, 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 (see [Gol04] for more on intersection graphs).

The graph $T = (V, E)$ represents a given tree with $|V| = n$. All paths referred to in this paper are paths from T unless explicitly specified. A *path system* $\mathcal{P}$ is a set system of paths from T. More precisely,

$\mathcal{P} = \{P \mid P \subseteq V,\ T[P] \text{ is a path.}\}$

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 $\ell : V(G) \rightarrow \mathcal{P}$ is called a *path labeling* of G. Moreover, for the purposes of this paper, we require that in a path labeling, $supp(\mathcal{P}) = V(T)$. This path system $\mathcal{P}$ is called a *path representation* of G and may also be denoted by G^{ℓ}. If $G = \mathbb{I}(\mathcal{F})$ where $\mathcal{F}$ is any set system, then clearly ℓ is a bijection from $\mathcal{F}$ to $\mathcal{P}$, ℓ is called the path labeling of set system $\mathcal{F}$ and the path system $\mathcal{P}$ may be alternatively denoted as $\mathcal{F}^{\ell}$.

path labeling is "almost" as same as path assignment. difference is that path labeling is defined only if the pairwise intersection non emptiness is preserved (due to the intersection graph isomorphism). in path assignment it is any assignment. ?????? does it matter if we take TPAs one step down to pairwise inteserction nonemptiness preservation?

A set system $\mathcal{F}$ can be represented as a hypergraph $\mathcal{H}_{\mathcal{F}}$ whose vertex set is $supp(\mathcal{F})$ and hyperedges are sets in $\mathcal{F}$. This is a known representation for interval systems in literature (
*** CITATION **** BLS99, Section 8.7 from KKLV10). We extend this definition here to path systems.

Two hypergraphs $\mathcal{H}, \mathcal{K}$ are said to be isomorphic to each other $\mathcal{H} \cong \mathcal{K}$ if there exists a bijection $\phi : supp(\mathcal{H}) \rightarrow supp(\mathcal{K})$ 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 $K = \{y \mid y = \phi(x), x \in H\}$.
If $\mathcal{H}_{\mathcal{F}}$ is isomorphic to hypergraph $\mathcal{H}_{\mathcal{P}}$ of a path system $\mathcal{P}$, then $\mathcal{H}_{\mathcal{F}}$ is called a *path hypergraph* (of course, $\mathcal{H}_{\mathcal{P}}$ is trivially a path hypergraph). Then $\mathcal{P}$ is called *path representation* of $\mathcal{H}_{\mathcal{F}}$. Clearly, by definition there exists an isomorphism $\phi : supp(\mathcal{H}_{\mathcal{F}}) \rightarrow supp(\mathcal{H}_{\mathcal{P}})$.

define hypergraph

Several path labelings could result in the same path representation. EXPAND?

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 k overlap components in $\mathbb{O}(\mathcal{F})$, the set subsystems are denoted by $\mathcal{F}_1, \mathcal{F}_2, \dots \mathcal{F}_k$. Clearly $\mathcal{F}_i \subseteq \mathcal{F}, i \in [k]$. For any $i, j \in [k]$, it can be verified that one of the following is true.

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

CHANGE the above for hypergraphs instead of set systems?

The terms overlap graph and overlap components are analogously defined for hypergraphs as well.
A path labeling $\ell : \mathcal{F} \rightarrow \mathcal{P}$ on $\mathcal{F}$ is defined to be *feasible* if the hypergraphs $\mathcal{H}_{\mathcal{F}}$ and $\mathcal{H}_{\mathcal{P}}$ are isomorphic to each other, and thus there exists an isomorphism $\phi : supp(\mathcal{F}) \rightarrow supp(\mathcal{P})$. Clearly, this implies $|supp(\mathcal{F})| = |supp(\mathcal{P})|$.

Let X be a partially ordered set with $\preccurlyeq$ being the partial order on X. $mub(X)$ represents an element in X which is a maximal upper bound on X. $X_m \in X$ is a maximal upper bound of X if $\nexists X_q \in X$ such that $X_m \preccurlyeq X_q$.

When refering to a tree as T it could be a reference to the tree itself, or the vertices of the tree. This will be clear from the context.

Finally, 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 $\ell : \mathcal{F} \to \mathcal{P}$ 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 $S_1, S_2, S_3 \in \mathcal{F}$

Lemma 1. *If ℓ is an ICPPL, and $S_1, S_2, S_3 \in \mathcal{F}$, then $|S_1 \cap (S_2 \setminus S_3)| = |P_1 \cap (P_2 \setminus P_3)|$ where $P_i = \ell(S_i), i \in \{1, 2, 3\}$.*

Proof. $|S_1 \cap (S_2 \setminus S_3)| = |(S_1 \cap S_2) \setminus S_3| = |S_1 \cap S_2| - |S_1 \cap S_2 \cap S_3|$. 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. □

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

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. □

In the remaining part of this section we show that a path labeling is feasible if and only if it is an ICPPL. 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. The reason is that a feasible path labeling has an associated bijection between *supp*($\mathcal{F}$) and $V(T)$ such that the sets map to paths. The rest of the section is devoted to constructively proving that it is sufficient, for a path labeling to be an ICPPL. At a top-level, the constructive approaches refine the path labeling 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*($\mathcal{F}$) or U to *supp*($\mathcal{P}$) or $V(T)$. First we present and then prove the correctness of Algorithm 1. This algorithm refines the path labeling by considering pairs of paths that share a leaf.

Algorithm 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**
  $\mathcal{F}_j = (\mathcal{F}_{j-1} \setminus \{S_1, S_2\}) \cup \{S_1 \cap S_2, S_1 \setminus S_2, S_2 \setminus S_1\}$
  for all $S \in \mathcal{F}_{j-1}$ such that $S \neq S_1$ and $S \neq S_2$, set $\ell_j(S) = \ell_{j-1}(S)$
  $\ell_j(S_1 \cap S_2) = \ell_{j-1}(S_1) \cap \ell_{j-1}(S_2)$
  $\ell_j(S_1 \setminus S_2) = \ell_{j-1}(S_1) \setminus \ell_{j-1}(S_2)$
  $\ell_j(S_2 \setminus S_1) = \ell_{j-1}(S_2) \setminus \ell_{j-1}(S_1)$
  $j = j + 1$
end while
$\mathcal{F}' = \mathcal{F}_j$, $\ell' = \ell_j$
Return $\mathcal{F}', \ell'$

Lemma 3. *In Algorithm 1, at the end of jth iteration, $j \geq 0$, of the while loop of Algorithm 1, the following invariants are maintained.*

- 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. 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 can be easily verified that all the new sets are also paths since by definition the chosen sets S_1 and S_1 are from $\mathcal{F}_{j-1}$ and by 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), invariant II can be verified easily for jth iteration by using the definition of l_j in terms of l_{j-1} 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 induction IV hypothesis, lemma 1 and definition of l_j with the fact that R' is not a new set, it can be verified that invariant III is true no matter which of the new sets R is equal to.
  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.*
  ***************** Consider the case where exactly one of them is not in Π_{j-1}. w.l.o.g let that be (P, Q) and it could be any one of the new sets. If (P, Q) is $(P_1 \cap P_2, Q_1 \cap Q_2)$, from lemma 2 and invariant III hypothesis, invariant IV is proven. Similarly if (P, Q) is any of the other new sets, invariant IV is proven by also using lemma 1. ********************
  Case 3.3: *At least two of R, R', R'' are new sets.* The new sets are disjoint hence this case is vacuously true.

□

It can be observed that the output of algorithm 1 is such that every leaf of T is incident on at most one path in the path images of the returned set system $\mathcal{F}$ on the returned path labeling ℓ. This is due to the loop condition at line 3. The next algorithm refines the path labeling and the set system further as follows. This is done to help reduce the size of the problem by pruning the tree off its leaves. 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 $\ell^{-1}(P) \setminus \{x\}$. After doing this for all leaves in T (that is part of a path image), 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.

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

```
Let F_0 = F
Let ℓ_0(S) = ℓ(S) for all S ∈ 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 ∈ F_{j−1} such that v ∈ ℓ_{j−1}(S_1) do
  F_j = F_{j−1} \ {S_1}
  for all S ∈ F_{j−1} such that S ≠ S_1 set ℓ_j(S) = ℓ_{j−1}(S)
  X = S_1 \ ⋃_{S ∈ F_{j−1}, S ≠ S_1} S
  if X is empty then
    exit
  end if
  Let x = arbitrary element from X
  F_j = F_j ∪ {{x}, S_1 \ {x}}
  ℓ_j({x}) = {v}
  ℓ_j(S_1 \ {x}) = ℓ_{j−1}(S_1) \ {v}
  j = j + 1
end while
F' = F_j
ℓ' = ℓ_j
Return F', ℓ'
```

Lemma 4. *In Algorithm 2, for all $j \geq 0$, at the end of the jth iteration the four invariants given in lemma 3 are valid.*

Proof. First we see that $X = S_1 \setminus \bigcup_{S \neq S_1} S$ is non-empty for an ICPPL in every iteration of this algorithm. Suppose X is empty. 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 there cannot exist a permutation that maps elements of S_{i_2} to P_{i_2}. This contradicts our assumption that this is an ICPPA. Thus X cannot be empty.**********

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.* ***************** If exactly one of them, say $P_3 \notin \Pi_{j-1}$, it is one of the new sets. By the same argument used to prove invariant III, $|P_1 \cap P_2 \cap (P' \setminus \{x\})| = |P_1 \cap P_2 \cap P'|$ and $|Q_1 \cap Q_2 \cap (Q' \setminus \{x\})| = |Q_1 \cap Q_2 \cap Q'|$. Since P_1, P_2, P' are all in Π_{j-1}, by induction hypothesis $|P_1 \cap P_2 \cap P'| = |Q_1 \cap Q_2 \cap Q'|$. Also $|P_1 \cap P_2 \cap \{x\}| = 0, Q_1 \cap Q_2 \cap \{v\} = 0$.

Case 3.3: *At least two of R, R', R'' are new sets.* *********** If two or more of them are not in Π_{j-1}, then it can be verified that $|P_1 \cap P_2 \cap P_3| = |Q_1 \cap Q_2 \cap Q_3|$ since the new sets in Π_j are either disjoint or as follows: assuming $P_1, P_2 \notin \Pi_{j-1}$ and new sets are derived from $(P', Q'), (P'', Q'') \in \Pi_{j-1}$ with x_1, x_2 exclusively in P_1, P_2, $(\{x_1\}, \{v_1\}), (\{x_2\}, \{v_2\}) \in \Pi_j$ thus v_1, v_2 are exclusively in Q_1, Q_2 resp. it follows that $|P_1 \cap P_2 \cap P_3| = |(P' \setminus \{x_1\}) \cap (P'' \setminus \{x_2\}) \cap P_3| = |P' \cap P'' \cap P_3| = |Q' \cap Q'' \cap Q_3| = |(Q' \setminus \{v_1\} \cap Q'' \setminus \{v_2\} \cap Q_3| = |Q_1 \cap Q_2 \cap Q_3|$. Thus invariant IV is also proven.

□

Using algorithms 1 and 2 we prove the following theorem.

Theorem 1. *If $\mathcal{F}$ has an ICPPL ℓ, then there exists a hypergraph isomorphism $\sigma : \mathcal{H}_{\mathcal{F}} \to \mathcal{H}$ such that $\sigma(S_i) = P_i$ for all $i \in I$*

Proof. This is a contructive proof. First, the given ICPPA $\mathcal{A}$ and tree T are given as input to Algorithm 1. This yields a "filtered" ICPPA as the output which is input to Algorithm 2. 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. To be precise, it would be of the form $\mathcal{B}_0 = \mathcal{A}_0 \cup \mathcal{L}_0$. The leaf assignments are defined in $\mathcal{L}_0 = \{(x_i, v_i) \mid x_i \in U, v_i \in T, x_i \neq x_j, v_i \neq v_j, i \neq j, i, j \in [k]\}$ where k is the number of leaves in T. The path assignments are defined in $\mathcal{A}_0 \subseteq \{(S'_i, P'_i) \mid S'_i \subseteq U_0, P'_i$ is a path from $T_0\}$ where T_0 is the tree obtained by removing all the leaves in T and $U_0 = U \setminus \{x \mid x$ is assigned to a leaf in $\mathcal{L}_0\}$. Now we have a subproblem of finding the permutation for the path assignment $\mathcal{A}_0$ which has paths from tree T_0 and sets from universe U_0. Now we repeat the procedure and the path assignment $\mathcal{A}_0$ and tree T_0 is given as input to Algorithm 1. The output of this algorithm is given to Algorithm 2 to get a new union of path and leaf assignments $\mathcal{B}_1 = \mathcal{A}_1 \cup \mathcal{L}_1$ defined similar to $\mathcal{B}_0, \mathcal{L}_0, \mathcal{A}_0$. In general, the two algorithms are run on path assignment $\mathcal{A}_{i-1}$ with paths from tree T_{i-1} to get a new subproblem with path assignment $\mathcal{A}_i$ and tree T_i. T_i is the subtree of T_{i-1} obtained by removing all its leaves. More importantly, it gives leaf assignments $\mathcal{L}_i$ to the leaves in tree T_{i-1}. This is continued until we get a subproblem with path assignment $\mathcal{A}_{d-1}$ and tree T_{d-1} for some $d \leq n$ which is just a path. From the last lemma we know that $\mathcal{A}_{d-1}$ is an ICPPA. Another observation is that an ICPPA with all its tree paths being intervals (subpaths from a path) is nothing but an ICPIA[NS09]. Let $\mathcal{A}_{d-1}$ be equal to $\{(S''_i, P''_i) \mid S''_i \subseteq U_{d-1}, P''_i$ is a path from $T_{d-1}\}$. It is true that the paths P''_is may not be precisely an interval in the sense of consecutive integers because they are some nodes from a tree. However, it is easy to see that the nodes of T_{d-1} can be ordered from left to right and ranked to get intervals

I_i for every path P_i'' as follows. $I_i = \{[l, r] \mid l =$ the lowest rank of the nodes in $P_i'', r = l + |P_i''| - 1\}$. Let asssignment $\mathcal{A}_d$ be with the renamed paths. $\mathcal{A}_d = \{(S_i'', I_i) \mid (S_i'', P_i'') \in \mathcal{A}_{d-1}\}$. What has been effectively done is renaming the nodes in T_{d-1} to get a tree T_d. The ICPIA $\mathcal{A}_d$ is now in the format that the ICPIA algorithm requires which gives us the permutation $\sigma' : U_{d-1} \to T_{d-1}$
σ' along with all the leaf assignments $\mathcal{L}_i$ gives us the permutation for the original path assignment $\mathcal{A}$. More precisely, the permutation for tree path assignment $\mathcal{A}$ is defined as follows. $\sigma : U \to T$ such that the following is maintained.

$$\begin{aligned}\sigma(x) &= \sigma'(x), \text{ if } x \in U_{d-1}\\ &= \mathcal{L}_i(x), \text{ where } x \text{ is assigned to a leaf in a subproblem } \mathcal{A}_{i-1}, T_{i-1}\end{aligned}$$

To summarize, run algorithm 1 and 2 on T. After the leaves have been assigned to specific elements from U, remove all leaves from T to get new tree T_0. The leaf assignments are in $\mathcal{L}_0$. Since only leaves were removed T_0 is indeed a tree. Repeat the algorithms on T_0 to get leaf assignments $\mathcal{L}_1$. Remove the leaves in T_0 to get T_1 and so on until the pruned tree T_d is a single path. Now run ICPIA algorithm on T_d to get permutation σ'. The relation $\mathcal{L}_0 \cup \mathcal{L}_1 \cup .. \cup \mathcal{L}_d \cup \sigma'$ gives the bijection required in the original problem. □

4 Finding an assignment of tree paths to a set system

In the previous section we have shown that the problem of finding a Tree Path Asssignment to an input $(\mathcal{F}, T)$ is equivalent to finding an ICPPA to $\mathcal{F}$ in tree T. In this section we characterize those set systems that have an ICPPA in a given tree. As a consequence of this characterization we identify two sub-problems that must be solved to obtain an ICPPA. We do not solve the problem and in the next section show that finding an ICPPA in a given tree is GI-Complete.
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 $U = \{x_1, \ldots, x_n\}$. If M represents $\mathcal{F}$, $|U| = n, |\mathcal{F}| = m$. Thus (i, j)th element of M, $M_{ij} = 1$ iff $x_i \in S_j$. If $\mathcal{F}$ has a feasible tree path assignment (ICPPA) $\mathcal{A} = \{(S_i, P_i) \mid i \in [m]\}$, then we say its corresponding matrix M has an ICPPA. Conversly we say that a matrix M has an ICPPA if there exists an ICPPA $\mathcal{A}$ as defined above.
We now define the strict intersection graph or overlap graph of $\mathcal{F}$. This graph occurs at many places in the literature, see for example [KKLV10, Hsu02, NS09]. The vertices of the graph correspond to the sets in $\mathcal{F}$. An edge is present between vertices of two sets iff the corresponding sets have a nonempty intersection and none is contained in the other. Formally, strict intersection graph is $G_f = (V_f, E_f)$ such that $V_f = \{v_i \mid S_i \in \mathcal{F}\}$ and $E_f = \{(v_i, v_j) \mid S_i \cap S_j \neq \emptyset \text{ and } S_i \not\subseteq S_j, S_j \not\subseteq S_i\}$. The usage of overlap graph to decompose the problem of consecutive ones testing was first introduced by [FG65]. 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 [Hsu02, NS09]. 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 G_f. Let us denote the prime sub-matrices by $M_1, \ldots, M_p$ each corresponding to one of the p components of G_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 $supp(M_i) = \bigcup\limits_{j \in col(M_i)} S_j$. Note that for each i, $supp(M_i) \subseteq U$. For a set of prime sub-matrices X we define $supp(X) = \bigcup\limits_{M \in X} supp(M)$.

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

$$\{(M_i, M_j) | \text{ A set } S \in M_i \text{ is contained in a set } S' \in M_j\} \cup \{(M_i, M_i) | 1 \leq i \leq p\}$$

This relation is the same as that defined in [NS09]. 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 an ICPPA for a set system $\mathcal{F}$. Recall the following lemmas, and theorem that $\preccurlyeq$ is a partial order, from [NS09].

Lemma 5. *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 6. *For each pair of prime sub-matrices, either $(M_i, M_j) \notin \preccurlyeq$ or $(M_j, M_i) \notin \preccurlyeq$.*

Lemma 7. *If $(M_i, M_j) \in \preccurlyeq$ and $(M_j, M_k) \in \preccurlyeq$, then $(M_i, M_k) \in \preccurlyeq$.*

Lemma 8. *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 2. *$\preccurlyeq$ 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 2. 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:

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

Theorem 3. *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 8 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. □

Let the partition of X given by Theorem 3 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 9. *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 ICPPA 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 ICPPA 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 ICPPA in T_i. It is clear from the properties of the partial order $\preccurlyeq$ that these ICPPAs naturally yield an ICPPA 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 ICPPA, 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 ICPPA in T_i. For each $1 \leq i \leq r$, 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 subtree T_i. All other prime-submatrices in X_i are assigned paths in T_i since $\mathcal{A}$ is an ICPPA, 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 ICPPA, 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 ICPPA in T_i. Hence the lemma. □

The essence of the following lemma is that an ICPPA 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 [NS09] 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 ICPPA in T.

Lemma 10. *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 ICPPA in T_i if and only if the matrix in $X_{i,0}$ has an ICPPA in T_i and all other matrices in X_i have an* ***ICPIA****.*

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 coresonding sub-matrices. Therefore, any ICPPA 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 ICPPA to the prime sub-matrix that is not at the root is an ICPIA, and any ICPIA can be used to construct an ICPPA to the matrices corresponding to nodes in X_i provided the matrix in the root has an ICPPA in T_i. Hence the lemma. □

Lemma 9 and Lemma 10 point out two algorithmic challenges in finding an ICPPA 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 9 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 ICPPA 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 10 which is to assign an ICPPA from a given tree to the matrix associated with the root node of X_i.

Algorithm 3 Algorithm to find an ICPPA for a matrix M on tree T: $main\_ICPPA(M, T)$

Identify the prime sub-matrices. This is done by constructing the strict overlap graph and identify 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 [NS09].
Find a partition of $T_1, \ldots, T_r$ such that matrices in $X_{i,0}$ has an ICPPA in T_i. If not such partition exists, exit with negative answer.

5 Complexity of Tree Path Assignment-A Discussion

Recall that the input to the Tree Path Assignment question is an order pair $(\mathcal{F}, T)$ where $\mathcal{F}$ is a family of subsets of an universe U, and T is a tree such that $|V(T)| = |U|$. The question is to come up with a bijection from U to $V(T)$ such that the image of each set in $\mathcal{F}$ is a path in T. We show that this problem is at least as hard as the problem of testing if two given path graphs are isomorphic.

Theorem 4. *Tree Path Assignment is isomorphism-complete.*

Proof. It is well known (see for example [KKLV10]) that testing isomorphism of path graphs is isomorphism complete. We show a reduction of path graph isomorphism to tree path assignment. Given G_1 and G_2 two path graphs, let T_2 be the clique tree of G_2 obtained from say [Gav78]. The nodes of T_2 correspond to the maximal cliques of G_2 and each vertex of G_2 corresponds to a path in G_2. This is a well-known characterization of path graphs and T_2 can be computed in polynomial time. In G_1, let S_v denote the maximal cliques of G_1 that contain v. This can be computed in polynomial time as G_1 is a path graph, and all chordal graph only have a linear number of maximal cliques. The universe U corresponds bijectively to the set of maximal cliques in G_1, and $\mathcal{F} = \{S_v | v \in V(G_1)\}$. Now, we claim that $(\mathcal{F}, T_2)$ has a tree path assignment if and only if G_1 and G_2 are isomorphic. This is clear since for each vertex $v \in G_1$, there is an associated S_v which is the set of maximal cliques containing v. In G_2, each vertex corresponds to a path in T_2, and the nodes on this path corresponds to the maximal cliques in G_2. Consequently, a tree path assignment will naturally yield an isomorphism between G_1 and G_2, and vice versa. Therefore, Tree Path Assignment is isomorphism-complete. □

5.1 Consecutive Ones Testing is in Logspace

While Tree Path Assignment is isomorphism-complete, it is polynomial time solvable when the given tree is a path. Indeed, in this case we encounter a restatement of matrices with the COP. The known approaches to testing for COP fall into two categories: those that provide a witness when the input matrix does not have the COP, and those that do not provide a witness. The first linear time algorithm for testing COP for a binary matrix was using a data structure called PQ trees, which represent all COP orderings of M, invented by [BL76]. There is a PQ tree for a matrix iff the matrix has the COP. Indeed, this is an algorithmic characterization of the consecutive ones property and the absence of the PQ-tree does not yield any witness to the reason for failure. A closely related data structure is the generalized PQ tree in [McC04]. In generalized PQ tree the P and Q nodes are called prime and linear nodes. Aside from that, it has a third type of node called degenerate nodes which is present only if the set system does not have the COP [McC04]. Using the idea of generalized PQ tree, [McC04] proves that checking for bipartiteness in the certain incomparability graph is sufficient to check for COP. [McC04] invented a certificate to confirm when a binary matrix does not have the COP. [McC04] describes a graph called incompatibility graph of a set system $\mathcal{F}$ which has vertices $(a, b), a \neq b$ for every $a, b \in U$, U being the universe of the set system. There are edges $((a, b), (b, c))$ and $((b, a), (c, b))$ if there is a set $S \in \mathcal{F}$ such that $a, c \in S$ and $b \notin S$. In other words the vertices of an edge in this graph represents two orderings that cannot occur in a consecutive ones ordering of $\mathcal{F}$.

Theorem 5 (Theorem 6.1, [McC04]). *Let $\mathcal{F}$ be an arbitrary set family on domain V. Then $\mathcal{F}$ has the consecutive ones property if and only if its incompatibility graph is bipartite, and if it does not have the consecutive ones property, the incompatibility graph has an odd cycle of length at most $n + 3$.*

This theorem gives a certificate as to why a given matrix does not have the COP. Similarly, the approach of testing for an ICPIA in [NS09] also gives a different certificate- a prime sub-matrix that does not have an ICPIA. Further, the above theorem can be used to check if a given matrix has the COP in logspace by checking if its incompatibility graph is bipartite. [Rei84] showed that checking for bipartiteness can be done in logspace. Thus we conclude that consecutive ones testing can be done in logspace.

More recently, [KKLV10] showed that interval graph isomorphism can be done in logspace. Their paper proves that a canon for interval graphs can be calculated in logspace using an interval hypergraph representation of the interval graph with each hyperedge being a set to which an interval shall be assigned by the canonization algorithm. An overlap graph (subgraph of intersection graph, edges define only strict intersections and no containment) of the hyperedges of the hypergraph is created and canons are computed for each overlap component. The overlap components define a tree like relation due to the fact that two overlap components are such that either all the hyperedges of one is disjoint from all in the other, or all of them are contained in one hyperedge in the other. This is similar to the containment tree defined in [NS09] and in this paper. Finally the canon for the whole graph is created using logspace tree canonization algorithm from [Lin92].

The interval labelling done in this process of canonization is exactly the same as the problem of assigning feasible intervals to a set system, and thus the problem of finding a COP ordering in a binary matrix [NS09].

Theorem 6 (Theorem 4.7, [KKLV10]). *Given an interval hypergraph $\mathcal{H}$, a canonical interval labeling l_H for H can be computed in FL.*

We present the following reduction to see that COP testing is indeed in logspace. Given a binary matrix M of order $n \times m$, let $S_i = \{j \mid M[j,i] = 1\}$. Let $\mathcal{F} = \{S_i \mid i \in [m]\}$ be this set system. Construct a hypergraph $\mathcal{H}$ with its vertex set being $\{1, 2, \dots n\}$. The edge set of $\mathcal{H}$ is isomorphic to $\mathcal{F}$. Thus every edge in $\mathcal{H}$ represents a set in the given set system $\mathcal{F}$. Let this mapping be $\pi : E(\mathcal{H}) \to \mathcal{F}$. It is easy to see that if M has the COP, then $\mathcal{H}$ is an interval hypergraph. From theorem 6, it is clear that the interval labeling $l_{\mathcal{H}} : V(\mathcal{H}) \to [n]$ can be calculated in logspace. Construct sets $I_i = \{l_{\mathcal{H}}(x) \mid x \in E, E \in E(\mathcal{H}), \pi(E) = S_i\}$, for all $i \in [m]$. Since $\mathcal{H}$ is an interval hypergraph, I_i is an interval for all $i \in [m]$, and is the interval assigned to S_i if M has the COP.

Now we have the following corollary.

Corollary 1. *If a binary matrix M has the COP then the interval assignments to each of its columns can be calculated in FL.*

Finally, we conclude by asking about the complexity of Tree Path Assignment restricted to other subclasses of trees. In particular, is Tree Path Assignment in caterpillars easier than Tree Path assignment in general trees.

A Detailed proofs

Proof (Lemma 3).

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

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.

[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
[7] Lemma 1
[8] ℓ_j definition. Note R' is not a new set

B TPA on 3 leaf trees

Try the (X_{i0}, T_i) tree path assignment on tree with three leaves (one node with deg 3) using the idea of slots from [KKLV10]

Backdrop: Algorithm 3 line 4 leaves an unsolved problem in the main ICPPA algorithm where ICPPA needs to be found out for the mub of each partition X_i i.e, X_{i0} on subtree T_i. In this section we assume the partitioning of the tree T into subtrees $\{T_i \mid T_i \subset T, 1 \leq i \leq t\}$ has been done. This is a problem that needs to be addressed separately.

Let $X_{i0} = \mathcal{F}, T_i = T$ for ease of notation.
Slots are sets of elements in the universe *supp*($\mathcal{F}$) which either appear together in a set or do not T_i appear at all (as defined in [KKLV10]). There is exactly one vertex r in that has degree 3, $d(r) = 3$. Which is the set(s) that maps to a path(s) that has r? In [KKLV10] they find the *side slot* by identifying the maximal marginal set from $\mathcal{F}$ (or hyperedge from hypergraph $\mathcal{H}$ in their set up). Using this side slot a partial order is defined that strictly orders the slots which is also the order in which their assigned intervals appear in the interval labeling.

Basic idea is to find the this set that gets assigned to deg 3 vertex path and then assign sub paths from the 3 paths incident on r.

Other observations:

- there exists exactly one set such that the sets that overlap with this set form 3 maximal inclusion chains.
- there exists exactly three sets such that for each of them, the sets that overlap with it forms a single maximal inclusion chain
- for every other set, there are exactly 2 inclusion maximal chain

`TRY AN EXAMPLE`

Let the identified slots be $Y_1, Y_2 ... Y_k$ and let slot that gets assigned to the path with r be Y_r. If T_i is rooted at r we have exactly three paths from r to the three leaves which are vertex disjoint except at r.

Use the slot ordering idea in [KKLV10] (page 10 para 1) $\leq_{Y_r}$. Now we get three (or 2?) orderings with only Y_r in common. Assign paths as in pg 10 eq (3).
Finding Y_r involves something similar to the marginal hyperedge idea in [KKLV10].
Or pick any one of the 3 “prongs” of the rooted version of T and delete it. Consider the remaining as an interval. Rename nodes to make it interval. Find the 3 maximal marginal sets. Arbitrarily choose one and use the ordering formula as in page 10 eq (2) in [KKLV10].

`TRY AN EXAMPLE`

Bibliography

[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.

[BP92] 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.

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

[Gav78] Fanica Gavril. A recognition algorithm for the intersection graphs of paths in trees. *Discrete Mathematics*, 23(3):211 – 227, 1978.

[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.

[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.

[KM02] P. S. Kumar and C. E. Veni Madhavan. Clique tree generalization and new subclasses of chordal graphs. *Discrete Applied Mathematics*, 117:109–131, 2002.

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

[Lin92] Steven Lindell. A logspace algorithm for tree canonization (extended abstract). In *STOC*, pages 400–404. ACM, 1992.

[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.

[PPY94] 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.

[Rei84] John H. Reif. Symmetric complementation. *JACM: Journal of the ACM*, 31(2):401–421, 1984.

[Ren70] Peter L. Renz. Intersection representations of graphs by arcs. *Pacific J. Math.*, 34(2):501–510, 1970.

[Sch93] Alejandro A. Schaffer. A faster algorithm to recognize undirected path graphs. *Discrete Applied Mathematics*, 43:261–295, 1993.