On Extensions of Consecutive Ones Testing

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. Consecutive ones property (COP) of a binary matrix is a combinatorial property of the matrix such that there exists a permutation of the rows of the matrix which makes all the ones in the columns consecutive. Detecting this property in a matrix is polynomial time solvable and there are several algorithms known. This problem is also the same as interval assigments to a set system. In this paper we extend this idea to tree path assignments to a set system and give a characterization for it.

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(rows) has its 1s occur consecutively. If this is possible the matrix is said to have COP. It has several practical applications in diverse fields including scheduling**?**, information retrieval**?** and computational biology**?**. Within theory it is used as a tool in graph theory**?** and integer linear programming**??**. Recognition of COP is polynomial time solvable by several algorithms. PQ trees**?**, variations of PQ trees**????**, ICPIA**?** are the main ones. The problem of COP 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 COP, a reordering of its rows will result in sets that have only consecutive integers. In other words, the sets are intervals. Indeed now the problem now becomes finding interval assignments to the given set system **?** with a single permutation of the universe (set of row indices) permutes each set to its interval. The idea of this algorithm is similar to that in **?** but much simpler. 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 from a universe U and a tree T, can there be a bijection from the U to the vertices of T such that each set in the system maps to a path in T. As a special case if T is a path the problem becomes the interval assignment problem or the ICPIA as it is called in **?**. It can be noted that intersection graphs of ICPIA set systems can be characterized by chordal graphs. Chordal graphs are of great significance and has several applications. One of the well known and interesting properties of a chordal graphs is its connection with intersection graphs**?**. 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**???**. Certain format of these trees are called clique trees**?** 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**?**. In the chordal graph as intersection graphs, if all the subtrees that map to the vertices of the graph in the isomorphism are paths on the tree, it is called a path graph**?** or path chordal graph. If a set system is an ICPIA or has feasible tree path assignment, its the intersection graph is an interval graph or a path graph respectively. This is because every vertex represents a set that can be permuted to an interval or a path in a tree. Path graph recognition is polynomial solvable**??**. If a given set system has an intersection graph that is isomorphic to a path graph, it is not necessary that the set system has feasible tree path assignment. It would be interesting to study if feasible tree path assignments can be used in path graph isomorphism.

2 Preliminaries (WIP)

In this paper, the collection $\mathcal{F} = \{S_i \mid S_i \subseteq U, S_i \neq \emptyset, i \in I\}$ is a *set system* of universe, $U = \{1, \ldots, n\}$. Moreoever, a set system is assumed to “cover” the universe, i.e. $\bigcup_{i \in I} S_i = U$.

The graph $T = (V, E)$ represents a given tree with $|V| = n$. Further, for simplicity, V is defined as $\{1, \ldots, n\}$. All paths referred to in this paper are paths on a tree.

Generalizing the definition of *feasibility* in **?** to a set assignment, a set assignment $\mathcal{A}_s$ is defined to be *feasible* if there exists a bijection defined as follows.

$$\sigma : U \to Z, \text{ such that } \sigma(S_i) = Q_i \text{ for all } i \in I, \sigma \text{ is a bijection} \tag{1}$$

The assignment $\mathcal{A}_s$ is then called *Set Translation Feasible.*

A *path assignment* $\mathcal{A}$ to $\mathcal{F}$ is defined as a set assignment where second universe is the vertex set V of a given tree T and every second subset in the ordered pairs is a path in this tree. Formally, the definition is as follows.

$$\mathcal{A} = \{(S_i, P_i) \mid S_i \in \mathcal{F}, P_i \subseteq V \text{ s.t. } T[P_i] \text{ is a path}, i \in I\}$$

In other words, P_i is the path on the tree T assigned to S_i in $\mathcal{A}$. As mentioned before for set systems, the paths cover the whole tree, i.e. $\bigcup_{i \in I} P_i = V$

Let X be a partially ordered set. The element $mub(X)$ represents the maximal upper bound of a poset X. A maximal upper bound X_m of a poset X is such that, $\nexists X_q \mid X_m, X_q \in X, X_m \preccurlyeq X_q$. $\preccurlyeq$ being the partial order.

The set I represents the index set $[m]$. If index i is used without further qualification, it is meant to be $i \in I$. Any function, if not defined on a domain of sets, when applied on a set is understood as the function applied to each of its elements. i.e. for any function f defined with domain U, the abuse of notation is as follows; $f(S)$ is used instead of $\hat{f}(S)$ where $\hat{f}(S) = \{y \mid y = f(x), x \in S\}$.

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 obvious from the context.

An in-tree is an oriented tree in which a single vertex is reachable from every other vertex.

3 Tree path assignment

A natural extension of the interval assignment problem is assignment of paths from a tree to a set system. Consider a path assignment $\mathcal{A} = \{(S_i, P_i) \mid S_i \in \mathcal{F}, P_i \text{ is a path from } T, i \in [m]\}$ to a set system $\mathcal{F} = \{S_i \mid S_i \subseteq U, i \in [m]\}$, were T is a given tree, U is the set system's universe and m is the number of sets in $\mathcal{F}$. We call $\mathcal{A}$ an *Intersection Cardinality Preserving Path Assignment (ICPPA)* if it has the following properties.

i. $|S_i| = |P_i|$ for all $i \in [m]$
ii. $|S_i \cap S_j| = |P_i \cap P_j|$ for all $i, j \in [m]$
iii. $|S_i \cap S_j \cap S_k| = |P_i \cap P_j \cap P_k|$ for all $i, j, k \in [m]$

Lemma 1. *If $\mathcal{A}$ is an ICPPA, and $(S_1, P_1), (S_2, P_2), (S_3, P_3) \in \mathcal{A}$, then $|S_1 \cap (S_2 \setminus S_3)| = |P_1 \cap (P_2 \setminus P_3)|$.*

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 ICPPA, $|S_1 \cap S_2| - |S_1 \cap S_2 \cap S_3| = |P_1 \cap P_2| - |P_1 \cap P_2 \cap P_3| = |(P_1 \cap P_2) \setminus P_3| = |P_1 \cap (P_2 \setminus P_3)|$. Thus lemma is proven. ☐

Lemma 2. *Consider four paths in a tree Q_1, Q_2, Q_3, Q_4 such that they have nonempty pairwise intersection and Q_1, Q_2 share a leaf. Then there exists $i, j, k \in \{1, 2, 3, 4\}$ with no two of i, j, k being equal such that, $Q_1 \cap Q_2 \cap Q_3 \cap Q_4 = Q_i \cap Q_j \cap Q_k$.*

Proof. *Case 1:* w.l.o.g, consider $Q_3 \cap Q_4$ and let us call it Q. This is clearly a path (intersection of two paths is a path). Suppose Q does not intersect with $Q_1 \setminus Q_2$, i.e. $Q \cap (Q_1 \setminus Q_2) = \emptyset$. Then $Q \cap Q_1 \cap Q_2 = Q \cap Q_2$. Similarly, if $Q \cap (Q_2 \setminus Q_1) = \emptyset$, $Q \cap Q_1 \cap Q_2 = Q \cap Q_1$. 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:* Let us consider the compliment of the previous case. i.e. the intersection of any two paths intersects with both the set differences of the other two. First let us consider $Q \cap (Q_1 \setminus Q_2) \neq \emptyset$ and $Q \cap (Q_1 \setminus Q_2) \neq \emptyset$, where $Q = Q_3 \cap Q_4$. Since Q_1 and Q_2 share a leaf, there is exactly one vertex at which they branch off from the path $Q_1 \cap Q_2$ into two paths $Q_1 \setminus Q_2$ and $Q_2 \setminus Q_1$. Let this vertex be v. It is clear that if path $Q_3 \cap Q_4$, must intersect with paths $Q_1 \setminus Q_2$ and $Q_2 \setminus Q_1$, it must contain v since these are paths from a tree. Moreover, $Q_3 \cap Q_4$ intersects with $Q_1 \cap Q_2$ at exactly v and only at v which means that $Q_1 \cap Q_2$ does not intersect with $Q_3 \setminus Q_4$ or $Q_4 \setminus Q_3$ which contradicts initial condition of this case. Thus this case cannot occur and case 1 is the only possible scenario.
Thus lemma is proven □

Following are two algorithms which will be used later to prove a property of the ICPPA. First we present and then prove the correctness of Algorithm **??**.

Algorithm 1 Permutations from an ICPPA $\{(S_i, P_i) | i \in I\}$

```
Let $\Pi_0 = \{(S_i, P_i) | i \in I\}$
$j = 1$;
while There is $(P_1, Q_1), (P_2, Q_2) \in \Pi_{j-1}$ with $Q_1$ and $Q_2$ having a common leaf do
    $\Pi_j = \Pi_{j-1} \setminus \{(P_1, Q_1), (P_2, Q_2)\}$;
    $\Pi_j = \Pi_j \cup \{(P_1 \cap P_2, Q_1 \cap Q_2), (P_1 \setminus P_2, Q_1 \setminus Q_2), (P_2 \setminus P_1, Q_2 \setminus Q_1)\}$;
    $j = j + 1$;
end while
$\Pi = \Pi_j$;
Return $\Pi$;
```

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

- Invariant I: Q *is a path in* T *for each* $(P, Q) \in \Pi_j$
- Invariant II: $|P| = |Q|$ *for each* $(P, Q) \in \Pi_j$
- Invariant III: *For any two* $(P, Q), (P', Q') \in \Pi_j$, $|P' \cap P''| = |Q' \cap Q''|$.
- Invariant IV: *For any three,* $(P', Q'), (P'', Q''), (P, Q) \in \Pi_j$, $|P' \cap P'' \cap P| = |Q' \cap Q'' \cap Q|$.

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 new sets added in jth iteration as defined in line **??** of Algorithm **??**, i.e. the following three assignment pairs for some $(P_1, Q_1), (P_2, Q_2) \in \Pi_{j-1}$ where Q_1 and Q_2 intersect and share a leaf: $(P_1 \cap P_2, Q_1 \cap Q_2)$, or $(P_1 \setminus P_2, Q_1 \setminus Q_2)$, or $(P_2 \setminus P_1, Q_2 \setminus Q_1)$.
The base case, $\Pi_0 = \{(S_i, P_i) \mid i \in [m]\}$, is trivially true since it is the input which is an ICPPA. Assume the lemma is true till the $j-1$ iteration. Consider jth iteration:
If (P, Q), (P', Q') and (P'', Q'') are in Π_j and Π_{j-1}, all the invariants are clearly true because they are from $j-1$ iteration.
If (P, Q) is in Π_j and not in Π_{j-1}, then it must be one of the new sets added in Π_j. Since (P_1, Q_1) and (P_2, Q_2) are from Π_{j-1} and Q_1, Q_2 intersect and have a common leaf, it can be verified that the new sets are also paths.
By hypothesis for invariant III, invariant II also holds for (P, Q) no matter which new set in Π_j it is.
To prove invariant III, if (P, Q) and (P', Q') are not in Π_{j-1}, then they are both new sets and invariant III holds trivially (new sets are disjoint). Next consider $(P, Q), (P', Q') \in \Pi_j$ with only one of them, say

(P', Q'), in Π_{j-1}. Then (P, Q) is one of the new sets added in line **??**. It is easy to see that if (P, Q) is $(P_1 \cap P_2, Q_1 \cap Q_2)$, then due to invariant IV in hypothesis, invariant III becomes true in this iteration. Similarly, using lemma **??** invariant III is proven if (P, Q) is $(P_1 \setminus P_2, Q_1 \setminus Q_2)$, or $(P_2 \setminus P_1, Q_2 \setminus Q_1)$.
To prove invariant IV, consider three assignments $(P, Q), (P', Q'), (P'', Q'')$. If at least two of these pairs are in not Π_{j-1}, then they are any two of the new sets. Note that these new sets are disjoint and hence if $(P', Q'), (P'', Q'')$ are any of these sets, $|P \cap P' \cap P''| = |Q \cap Q' \cap Q''| = 0$ and invariant IV is true. Now we consider the case if at most one of $(P, Q), (P', Q'), (P'', Q'')$ is not in Π_{j-1}. If none of them are not in Π_{j-1} (i.e. all of them are in Π_{j-1}), invariant IV is clearly true. 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 **??** 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 **??**. □

It can be observed that the output of algorithm **??** is such that every leaf is incident on at most a single path in the new set of assignments. This is due to the loop condition at line **??**. Let v_1 be the leaf incident on path P_i. Assign to it any one element from $S_i \setminus \bigcup_{i \neq j} S_j$. Remove (S_i, P_i) from assignments and add $(\{x_1\}, \{v_1\}), (S_i \setminus \{x_1\}, P_i \setminus \{v_1\})$. Now all assignments except single leaf assignments are paths from the subtree $T_0 = T \setminus \{v \mid v \text{ is a leaf in } T\}$.

Algorithm 2 Leaf assignments from an ICPPA $\{(S_i, P_i) | i \in I\}$

```
Let $\Pi_0 = \{(S_i, P_i) | i \in [m]\}$. Paths are such that no two paths $P_i, P_j, i \neq j$ share a leaf.
$j = 1$
while there is a leaf $v$ and a unique $(S_{i_1}, P_{i_1})$ such that $v \in P_{i_1}$ do
    $\Pi_j = \Pi_{j-1} \setminus \{(S_{i_1}, P_{i_1})\}$
    $X = S_{i_1} \setminus \bigcup_{i \neq i_1, i \in I} S_i$
    if $X$ is empty then
        exit
    end if;
    Let $x$ = arbitrary element from $X$

    $\Pi_j = \Pi_j \cup \{(S_{i_1} \setminus \{x\}), (P_{i_1} \setminus \{v\}), (\{x\}, \{v\})\}$
    $j = j + 1$
end while
$\Pi = \Pi_j$
Return $\Pi$
```

Lemma 4. *In Algorithm **??**, at the end of the jth iteration, $j \geq 0$, of the while loop of Algorithm **??**, the invariants given in lemma **??** hold. Moreover, X as defined in the algorithm is non empty if this is an ICPPA.*

Proof. First we see that $X = S_{i_1} \setminus \bigcup_{i \neq i_1, i \in I} S_i$ is non empty in every iteration for an ICPPA. Suppose X is empty. We know that $v \in P_{i_1} \setminus \bigcup_{i \neq i_1, i \in I} P_i$ since v is in the unique path P_{i_1}. Since this is an ICPPA $|S_{i_1}| = |P_{i_1}|$. For any $x \in S_{i_1}$ it is contained in at least two sets due to our assumption. Let S_{i_2} be a second set that contains x. We know $v \notin P_{i_2}$. 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.

We use mathematical induction on the number of iterations for this proof. The term "new sets" will refer to the sets added in Π_j in the jth iteration, i.e. $(P' \setminus \{x\}, Q' \setminus \{v\})$ and $(\{x\}, \{v\})$ for some (P', Q') in Π_{j-1} such that v is a leaf and Q' is the unique path incident on it.
For Π_0 all invariants hold because it is output from algorithm **??** which is an ICPPA. Hence base case is proved. Assume the lemma holds for Π_{j-1}. Consider Π_j and any $(P, Q) \in \Pi_j$. If (P, Q) is in Π_j and Π_{j-1} invariants I and II are true because of induction assumption. If it is only in Π_j, then it is $\{(P' \setminus \{x\}), (Q' \setminus \{v\})$ or $(\{x\}, \{v\})$ for some (P', Q') in Π_{j-1}. By definition, x is an element in P' (as defined in the algorithm)

and v is a leaf in Q'. If (P, Q) is $\{(P' \setminus \{x\}), (Q' \setminus \{v\})$, Q is a path since only a leaf is removed from path Q'. We know $|P'| = |Q'|$, therefore $|P' \setminus \{x\}| = |Q' \setminus \{v\}|$. Hence in this case invariants I and II are obvious. It is easy to see these invariants hold if (P, Q) is $(\{x\}, \{v\})$.

For invariant III consider $(P_1, Q_1), (P_2, Q_2)$ in Π_j. If both of them are also in Π_{j-1}, claim is proved. If one of them is not in Π_{j-1} then it has to be $\{(P' \setminus \{x\}), (Q' \setminus \{v\})$ or $(\{x\}, \{v\})$ for some (P', Q') in Π_{j-1}. Since by definition, Q' is the only path with v and P' the only set with x in the previous iteration, $|P_1 \cap (P' \setminus \{x\})| = |P_1 \cap P'|$ and $|Q_1 \cap (Q' \setminus \{v\})| = |Q_1 \cap Q'|$ and $|P_1 \cap \{x\}| = 0, Q_1 \cap \{v\} = 0$. Thus invariant III is also proven.
To prove invariant IV, consider $(P_1, Q_1), (P_2, Q_2), (P_3, Q_3)$ in Π_j. 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 $|P_1 \cap P_2 \cap \{x\}| = 0, Q_1 \cap Q_2 \cap \{v\} = 0$. 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 **??** and **??** we prove the following theorem.

Theorem 1. *If $\mathcal{A}$ is an ICPPA, then there exists a bijection $\sigma : U \to V(T)$ 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 **??**. This yields a "filtered" ICPPA as the output which is input to Algorithm **??**. It can be observed that the output of Algorithm **??** is a set of interval 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 **??**. The output of this algorithm is given to Algorithm **??** 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**?**. 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 **??** and **??** 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. $\square$

4 Finding an assignment of tree paths to a set system

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.
Consider the strict intersection graph or overlap graph of $\mathcal{F}$. It is constructed with its vertices denoting a unique subset in the set system and 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, intersection graph is $G_f = (V_f, E_f)$ such that $V_f = \{v_i \mid S_i \in \mathcal{F}\}$ and $E_f = \{(v_i, v_j) \mid S_i \cap S_j \neq \emptyset$ and $S_i \nsubseteq S_j, S_j \nsubseteq S_i\}$. We use this graph to decompose M as described in **??**. 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 $supp(X) = \bigcup\limits_{M \in X} supp(M)$.

Now we define a partial order on the prime submatrices. 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 partial order is the same as that defined in **?**. The prime submatrices and the above partial order can be defined for any set system. We will use this theory of prime submatrices to find an ICPPA for a set system $\mathcal{F}$. Recall the following lemmas and theorems.

Lemma 5 (Lemma 3, ?). *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 (Lemma 4, ?). *For each pair of prime sub-matrices, either $(M_i, M_j) \notin \preccurlyeq$ or $(M_j, M_i) \notin \preccurlyeq$.*

Lemma 7 (Lemma 5, ?). *If $(M_i, M_j) \in \preccurlyeq$ and $(M_j, M_k) \in \preccurlyeq$, then $(M_i, M_k) \in \preccurlyeq$.*

Lemma 8 (Lemma 6, ?). *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 (Theorem 4, ?). *$\preccurlyeq$ 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.*

Now we define another relation $\ll$ on the set of prime submatrices as follows:

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

Lemma 9. *Relation $\ll$ is a partial order*

Proof. Reflexive by definition. Antisymmetric and transitive because $\preccurlyeq$ is antisymmetric and transitive.

- Hasse diagram of $\ll$ is an intree.

Theorem 3. *Relation $\ll$ 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 $\ll$ induces intrees. and the intrees partition into total orders.*

Proof. Relation $\ll$ is relective, antisymmetric and transitive because so is $\preccurlyeq$. Consider the Hasse diagram of $\ll$. By definition mubs have outdegree 0
Claim 1: every other node has out degree 1.
we wil prove this by induction on the level of the tree. level 1 has this true since root(mub) is the only possible parent. assume this is true for all nodes till level k-1 . Consider level k. assume there is a node, M_i in level k that has out degree 2 going into two nodes say. M_{i1} and M_{i2}. i.e. $(M_i, M_{i1}$ and $(M_i, M_{i2}$. We also know that $(M_{i1}, M_p) \in \ll$. Therefore by transitivity, $(M_i, M_p) \in \ll$. Then M_i should have been related in $\ll$ to M_i instead of M_{i1}. This is a contradiction to the assumption above.

Claim 2: Every node other than the root has a path to the root proof: tbd.

claim: matrices in each level can be partitioned s.t. each partition set is a total order. - suggested construction: matrices of the level($leveli$) with common parent ($leveli - 1$) is one partition.
claim: these created partitions are disjoint disjoint. there is total order in the partition.

From Theorem **??** we can see that the set of prime submatrices can be partitioned into sets of matrices that form in-trees.

Let X be an in-tree and T be a tree. It is said X has ICPPA on T iff the prime matrix $mub(X)$ has an ICPPA with some subtree $T_1 \subseteq T$ and prime matrices $X \setminus mub(X)$ have ICPIA.

Lemma 10. *A matrix M has an ICPPA iff there exists a tree T that has subtrees $T_1, T_2, \ldots T_r \in T, T_i \cap T_j \neq \emptyset, i \neq j, i, j \in [r]$ such that, intrees $X_1, X_2, \ldots X_r$ have an ICPPA in trees $T_1, T_2, \ldots T_r$. $X_1, X_2, \ldots X_r$ represent in-trees of prime submatrices of M due to the partial order $\ll$.*

Proof. First we prove that if a matrix M has an ICPPA, then there exists tree T as described in lemma statement. If M has an ICPPA, then there exists a tree T with an assignment $\mathcal{A}$ of paths from T to sets in M. From Theorem **??** we know that M can be decomposed into a partition of prime submatrices $X_1, X_2, \ldots X_r$. Let $M_{Xi} = mub(X_i)$. The assignment of sets in M_{Xi} are paths by definition and being a prime matrices these sets strictly overlap. Thus it is clear that the vertices $\{\mathcal{A}(x) \mid x \in supp(mub(X_i))\ \}$ for any $i \in [r]$ induce a subtree in T, let us call it T_i. Since $X_i \cap X_j = \emptyset, i \neq j$ and $\mathcal{A}$ is a bijection[1], $T_i \cap T_j = \emptyset, i \neq j$. Thus claim proven.

Now we need to prove the if condition of lemma. We prove by construction of an algorithm to find the ICPPA for a given matrix M and a tree T.

TO PROVE: IF there exists a tree T which has subtrees $T_1, T_2, \ldots T_r \in T, T_i \cap T_j \neq \emptyset, i \neq j, i, j \in [r]$ such that, the intrees rooted at $mub(X_1), mub(X_2), \ldots mub(X_r)$ have an ICPPA from $T_1, T_2, \ldots T_r$ resp. $X_1, X_2, \ldots X_r$ are the partition sets of prime submatrices of M due to the partial order $\preccurlyeq$. $mub(X)$ denotes the maximal upper bound of a poset X. THEN then sets in M have path assignments that preserve ICPPA conditions

Since $mub(X_i)$ has ICPPA from T_i, every set in the matrix mub has a path in T_i that form an ICPPA. By definition of mub and X_i, all other sets in partition X_i are descendants of X_i. there is a unique path from any node mi in hasse of X_i to mub(X_i). The matrix adjacent to mub(X_i) on this path let it be M_{xi1},

[1] sloppy tbd

and path P. M_{xi1} is completely contained in a set in $mub(X_i)$, say S_{xi1}. P_{xil} is a path in T_i assigned to S_{xil}. Effectively $Pxil$ is an interval. Thus if M_{xi2} is matrix adjacent to M_{xi1} in the path p, to get path assignments to M_{xi2} from P_{xil} is an ICPIA problem. Similarly $supp(Mxi2) \subseteq S_{xi1}$ a set in M_{xi1}. P'_{xil} is again a path and path assignments to M_{xi3}, next in the path P, from P'_{xil} is an ICPIA problem and so on till the mlbs.

to prove: all sets have paths assigned to them.

The following is the outline of the algorithm. Let $\mathcal{F}$ be the given set system and T be the given tree. Consider one of the partitions X_i that is induced by $\preccurlyeq$. By theorem **??**, we know that the submatrices in X_i can be ordered in the following manner- $M_{ik} \preccurlyeq ... \preccurlyeq M_{i2} \preccurlyeq M_{i1}$ where $\{i_1, i_2, ..i_k\} \subseteq [p]$ and p is the number of prime submatrices, $M_{i_j} \in X_i, j \in [k]$. From the definition of $\preccurlyeq$, for each r, $2 \leq r \leq k$, $supp(M_{ir})$ is contained in at least one set in $M_{i(r-1)}$. Therefore, it follows that $supp(X_i) = supp(M_{i1})$. We find a subtree $T_i \subseteq T$, such that $supp(M_{i1}) = |T_i|$, and associate it with M_{i1}. Now we assign paths from T_i to the sets $col(M_{i1})$ maintaining the ICPPA condition. If we are unable to do so, we try the next subtree of size $|supp(M_{i1})|$ and so on. Suppose paths have been successfully assigned to the sets in M_{i1}. We know that $supp(M_{i2})$ is contained in at least one column/set $S \in col(M_{i1})$. Thus the subtree associated with $supp(M_{i2})$ is contained in the path associated with S. Now this becomes an interval assignment problem and ICPIA algorithms can be used to assign paths to all $S' \in M_{i2}$. Similarly ICPIA can be used to assign paths to $col(M_{i_j})$ for all $2 \leq j \leq k$.

For simplicity, we consider T to be a rooted tree with an arbitrary node as its root.

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 $X_1, \dots, X_l$ of the prime sub-matrices induced by $\preccurlyeq$
Construct the total order on each set in the partition. Let $M_{k1}, M_{k2}, ..., M_{kj_k}$ be the total order of partition $X_k$ in
reverse order.
$\mathcal{A}_M = \emptyset$
**for** $(k = 1; k \leq l; k++)$ **do**
  Assign some subtree $T' \subseteq T$ to $M_{k1}$ s.t. $|T'| = supp(M_{k1})$
  Assign paths from $T'$ to sets in $M_{k1}$. Let this assignment be $\mathcal{A}_{k1} =$
  $\{(S_{ki}, P_{ki}) | S_{ki}$ is a set in $M_{k1}, P_{ki}$ is a path from $T'\}$
  $\mathcal{A}_k = \mathcal{A}_{k1}$
  **for** $2 \leq r \leq j_k$ **do**
    **if** parent of $M_{kr}$ in total order is $M_{k1}$ **then**
      Find the set in $M_{k1}$ that contains $supp(M_{kr})$. Let it be $S_{kr}$
      $\mathcal{A}_{kr} = ICPPA(M_{kr}, P_{kr}, k, r)$
      **if** $\mathcal{A}_{kr}$ is invalid **then**
        exit
      **end if**
      $\mathcal{A}_k = \mathcal{A}_k \cup \mathcal{A}_{kr}$
    **end if**
  **end for**
  $\mathcal{A}_M = \mathcal{A}_M \cup \mathcal{A}_{kr}$
**end for**
return $\mathcal{A}_M$
```

The ranking in algorithm **??** of the tree path T_x is as follows. Assume one of the ends of path as the left vertex and the other as right vertex. Rank the vertices of T_x from left to right to get $I_x = \{r \mid r = rank(v), v \in T_x\}$. i.e. $I_x = [1, |T_x|]$

Algorithm 4 Recursive algorithm ICPPA(M_{km}, T_x, k, m), M is the matrix which to which paths from T_x must be assigned. k is the partition being processed and m is index of the submatrix in the total order for M_{km}. For simplicity we refer to M_{km} as M_x below
.

```
if $|col(M_x) = 1|$ and $T_x$ is a path then
  return $\{(S, T_x)\}$ where $S$ is the only set in $M_x$
else
  if $|col(M_x) = 1|$ then
    Report failure and return
  end if
end if
Rank the vertices of $T_x$ to get interval $I_x$.
$\mathcal{A}' =$ ICPIA $(M_x, I_x)$
Convert intervals back to tree paths. $P_i = \{v \mid rank(v) = r, r \in I_i\}$ for every interval $I_i$ assigned in $\mathcal{A}'\}$
$\mathcal{A} = \{(S_i, P_i) \mid$ for all $S_i$ represented in $M_x\}$
$\mathcal{A}_f = \mathcal{A}$
for $s = m + 1$ to $j_k$ do
  if $M_x$ is the parent of $M_{ks}$ then
    Find $S_j$ in $M_x$ that contains $supp(M_{ks})$
    $\mathcal{A}_s = ICPPA(M_{ks}, P_j, k, s)$
    $\mathcal{A}_f = \mathcal{A}_f \cup \mathcal{A}_f$
  end if
end for
return $\mathcal{A}_f$
```

5 COP in logspace

5.1 Inverval hypergraphs

? showed that interval graphs 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 **?**. Finally the canon for the whole graph is created using logspace tree canonization algorithm from **?**. 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**?**.

Theorem 4 (Theorem 4.7, ?). *Given an interval hypergraph $\mathcal{H}$, a canonical interval labeling l_H for H can be computed in FL.*

Using the following construction it can be seen 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 COP, then $\mathcal{H}$ is an interval hypergraph. From theorem **??**, 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 COP.

Now we have the following corollary.

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

5.2 Generalized PQ tree

The first linear time algorithm for testing COP for a binary matrix was using a data structure called PQ trees invented by **?**. There is a PQ tree for a matrix iff the matrix has COP. A PQ tree is a "partly" ordered tree which is created for a binary matrix M with COP. A PQ tree represents all COP orderings of M.

The PQ tree is constructed by considering one column at a time and at each iteration, either the algorithm reports M has no COP or changes the tree to create a PQ tree for the matrix induced by the columns considered so far. The construction is very elaborate and complicated.

A closely related data structure is the generalized PQ tree in **?**. 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 COP.

Theorem 5 (? Theorem 3.6). *If $\mathcal{F}$ has COP, its generalized PQ tree has no degenerate nodes*

Using the idea of generalized PQ tree, this result proves that checking for bipartiteness in the certain incomparability graph is sufficient to check for COP. **?** invented a certificate to confirm when a binary matrix does not have COP. This is, to the best of our knowledge, the first result that came up with such a verification strategy. Earlier work only involved recognition of COP. If a matrix is recognized to have COP, this is easy to verify because the algorithm outputs the COP ordering (permutation of rows for COP in columns) and it is only a matter of applying this ordering for verification.

? 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 6 (Theorem 6.1, ?). *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 $n + 3$*[2]

Thus to check for COP of a set system (or matrix), one can check if its incompatibility graph is bipartite. **?** showed that checking for bipartiteness can be done in logspace. Thus we arrive at corollary **??** using this approach of COP testing as well.

[2] original theorem says $n + 2$ but this is an error and was corrected by **?**