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

N.S. Narayanaswamy[1] **Anju Srinivasan**[2,3]

Computer Science and Engineering Department,
Indian Institute of Technology Madras, Chennai - 600036

[1]swamy@cse.iitm.ernet.in, [2]asz@cse.iitm.ac.in, [3]anjuzabil@gmail.com

Abstract

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

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 [Hochbaum and Levin, 2006], information retrieval [Kou, 1977] and computational biology [Atkins et al., 1998]. Further, it is a tool in graph theory [Golumbic, 2004] for interval graph recognition, characterization of hamiltonian graphs, and in integer linear programming [Hochbaum and Tucker, 2002; Hochbaum and Levin, 2006]. Recognition of COP is polynomial time solvable by several algorithms. PQ trees [Booth and Lueker, 1976], variations of PQ trees [Meidanis and Munuera, 1996; Hsu, 2001, 2002; McConnell, 2004], ICPIA [Narayanaswamy and Subashini, 2009] are the main ones.

The problem of COP testing can be easily seen as a simple constraint satisfaction problem involving a system of sets from a universe. Every column of the binary matrix can be converted into a set of integers which are the indices of the rows with 1s in that column. When observed in this context, if the matrix has the COP, a reordering of its rows will result in sets that have only consecutive integers. In other words, the sets after reordering are intervals. Indeed the problem now becomes finding interval assignments to the given set system [Narayanaswamy and Subashini, 2009] with a single permutation of the universe (set of row indices) which permutes each set to its interval. The result in [Narayanaswamy and Subashini, 2009] characterize interval assignments to the sets which can be obtained from a single permutation of the rows. They show that for each set, the cardinality of the interval assigned to it must be same as the cardinality of the set, and the intersection cardinality of any two sets must be same as the interesction cardinality of the corresponding intervals. While this is naturally a necessary condition, [Narayanaswamy and Subashini, 2009] shows this is indeed sufficient. Such an interval assignment is called an Intersection Cardinality Preserving Interval Assignment (ICPIA). Finally, the idea of decomposing a given 0-1 matrix into prime matrices to check for COP is adopted from [Hsu, 2002] to test if an ICPIA exists for a given set system.

Our Work. A natural generalization of the interval assignment problem is feasible tree path labeling problem of a set system. The problem is defined as follows - given a set system $\mathcal{F}$ from a universe U and a tree T, does there exist a bijection from the U to the vertices of T such that each set in the system maps to a path in T. We refer to this as the tree path labeling problem for an input $(\mathcal{F}, T)$ pair. As a special case if T is a path the problem becomes the interval assignment problem. We focus on the question of generalizing the notion of an ICPIA [Narayanaswamy and Subashini, 2009] to characterize feasible path assignments. We show that for a given set system $\mathcal{F}$, a tree T, and an assignment of paths from T to the sets, there is a bijection between U and $V(T)$ if and only if all intersection cardinalities among any three sets (not necessarily distinct) is same as the intersection cardinality of the paths assigned to them and the input runs a filtering algorithm (described in this paper) without prematurely exiting. This characterization is proved constructively and it gives a natural data structure that stores all the relevant bijections between U and $V(T)$. Further, the filtering algorithm is also an efficient algorithm to test if a tree path labeling to the set system is feasible. This generalizes the result in [Narayanaswamy and Subashini, 2009]. It is an interesting fact that for a matrix with the COP, the intersection graph of the corresponding set system is an interval graph. A similar connection to a subclass of chordal graphs and a superclass of interval graphs exists for the generalization of COP. In this case, the intersection graph of the corresponding set system must be a *path graph*. Chordal graphs are of great significance, extensively studied, and have several applications. One of the well known and interesting properties of a chordal graphs is its connection with intersection graphs [Golumbic, 2004]. 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 [Renz, 1970; Gavril, 1978; Blair and Peyton, 1992]. These trees when in a certain format, are called clique trees [Peyton et al., 1994] of the chordal graph. This is a compact representation of the chordal graph. There has also been work done on the generalization of clique trees to clique hypergraphs [Kumar and Madhavan, 2002]. If the chordal graph can be represented as the intersection graph of paths in a tree, then the graph is called path graph [Golumbic, 2004]. Therefore, it is clear that if there is a bijection from U to $V(T)$ such that for every set, the elements in it map to vertices of a unique path in T, then the intersection graph of the set system is indeed a path graph. However, this is only a necessary condition and can be checked efficiently because path graph recognition is polynomial time solvable[Gavril, 1978; Schaffer, 1993]. Indeed, it is possible to construct a set system and tree, such that the intersection graph is a path graph, but there is no bijection between U and $V(T)$ such that the sets map to paths. Path graph isomorphism is known be isomorphism-complete, see for example [Köbler et al., 2010]. An interesting area of research would be to see what this result tells us about the complexity of the tree path labeling problem (not covered in this paper). In the later part of this paper, we decompose our search for a bijection between U and $V(T)$ into subproblems. Each subproblem is on a set subsystem in which for each set, there is another set in the set subsystem with which the intersection is *strict* - i.e., there is a non-empty intersection, but neither is contained in the other. This is in the spirit of results in [Hsu, 2002; Narayanaswamy and Subashini, 2009] where to test for the COP in a given matrix, the COP problem is solved on an equivalent set of prime matrices.

Roadmap. In Section 2 we present the necessary preliminaries. In Section 4 we present a polynomial time algorithm to find the tree path labeling of a given set system from a given k-subdivided tree.

2 Preliminaries

In this paper, the set $\mathcal{F} \subseteq (2^U \setminus \emptyset)$ is a *set system* of a universe U with $|U| = n$. The *support* of a set system $\mathcal{F}$ denoted by $supp(\mathcal{F})$ is the union of all the sets in $\mathcal{F}$, i.e., $supp(\mathcal{F}) = \bigcup_{S \in \mathcal{F}} S$. For the purposes of this paper, a set system is required to "cover" the universe, i.e. $supp(\mathcal{F}) = U$.

The graph T represents a given tree with $|V(T)| = n$. A *path system* $\mathcal{P}$ is a set system of paths from T i.e., $\mathcal{P} = \{P \mid P \subseteq V,\ T[P] \text{ is a path.}\}$

A set system $\mathcal{F}$ can be alternatively represented by a *hypergraph* $\mathcal{H}_\mathcal{F}$ whose vertex set is $supp(\mathcal{F})$ and hyperedges are the sets in $\mathcal{F}$. This is a known representation for interval systems in literature [Brandstädt et al., 1999; Köbler et al., 2010]. We extend this definition here to path systems.

The *intersection graph* $\mathbb{I}(\mathcal{H})$ of a hypergraph $\mathcal{H}$ is a graph such that its vertex set has a bijection to (the set of hyperdeges of) $\mathcal{H}$ and there exists an edge between two vertices iff their corresponding hyperedges have a non-empty intersection [Golumbic, 2004].

Two hypergraphs $\mathcal{H}$, $\mathcal{K}$ are said to be isomorphic *(hypergraph isomorphism)* to each other, denoted by $\mathcal{H} \cong \mathcal{K}$, iff there exists a bijection $\phi : supp(\mathcal{H}) \to supp(\mathcal{K})$ such that for all sets $H \subseteq supp(\mathcal{H})$, H is a hyperedge in $\mathcal{H}$ iff K is a hyperedge in $\mathcal{K}$ where $K = \{\phi(x) \mid x \in H\}$.

If $\mathcal{H}_\mathcal{F} \cong \mathcal{H}_\mathcal{P}$ where $\mathcal{P}$ is a path system, then $\mathcal{H}_\mathcal{F}$ is called a *path hypergraph* and $\mathcal{P}$ is called *path representation* of $\mathcal{H}_\mathcal{F}$. If isomorphism is $\phi : supp(\mathcal{H}_\mathcal{F}) \to supp(\mathcal{H}_\mathcal{P})$, then it is clear that there is an induced path labeling $l_\phi : \mathcal{F} \to \mathcal{P}$ to the set system. In the rest of the document, we may use $\mathcal{F}$ and/or $\mathcal{H}_\mathcal{F}$ interchangeably to refer the set system and/or its hypergraph.

A graph G that is isomorphic to the intersection graph $\mathbb{I}(\mathcal{P})$ of a path system $\mathcal{P}$ of T, is a *path graph.* This isomorphism gives a bijection $\ell' : V(G) \to \mathcal{P}$. Moreover, for the purposes of this paper, we require that in a path labeling, $supp(\mathcal{P}) = V(T)$. If graph $G = \mathbb{I}(\mathcal{H})$ for some hypergraph $\mathcal{H}$, then clearly there is a bijection $\ell : \mathcal{H} \to \mathcal{P}$ such that $\ell(S) = \ell'(v_S)$ where v_S is the vertex corresponding to set S in $\mathbb{I}(\mathcal{H})$ for any $S \in \mathcal{H}$. This bijection ℓ is called the *path labeling* of the hypergraph $\mathcal{H}$ and the path system $\mathcal{P}$ may be alternatively denoted as $\mathcal{H}^\ell$.

An *overlap graph* $\mathbb{O}(\mathcal{H})$ of a hypergraph $\mathcal{H}$ is a graph such that its vertex set has a bijection to $\mathcal{H}$ and there exists an edge between two vertices iff their corresponding hyperedges overlap. Two sets (hyperedges) 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 $\mathbb{O}(\mathcal{H})$ is a subgraph of $\mathbb{I}(\mathcal{H})$ and not necessarily connected. Each connected component of $\mathbb{O}(\mathcal{H})$ is called an *overlap component.*

A path labeling $\ell : \mathcal{F} \to \mathcal{P}$ of setsystem $\mathcal{F}$ is defined to be *feasible* if their hypergraphs are isomorphic to each other, $\mathcal{H}_\mathcal{F} \cong \mathcal{H}_\mathcal{P}$ and if this isomorphism $\phi : supp(\mathcal{F}) \to supp(\mathcal{P})$ induces a path labeling $\ell_\phi : \mathcal{F} \to \mathcal{P}$ such that $\ell_\phi = \ell$.

A *star* graph is a complete bipartite graph $K_{1,l}$ which is clearly a tree and p is the number of leaves. The vertex with maximum degree is called the *center* of the star and the edges are called *rays* of the star.

A *k-subdivided star* is a star with all its rays subdivided exactly k times. The path from the center to a leaf is called a ray of a k-subdivided star and they are all of length $k + 2$.

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 $S_1, S_2, S_3 \in \mathcal{F}$

The following lemma is useful in characterizing feasible tree path labelings. The proof is in the appendix.

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

In the remaining part of this section we describe an algorithmic characterization for a feasible tree path labeling. We show that a path labeling is feasible if and only if it is an ICPPL and it successfully passes the filtering algorithms 1 and 2. One direction of this claim is clear: that if a path labeling is feasible, then all intersection cardinalities are preserved, i.e. the path labeling is an ICPPL. Algorithm 1 has no premature exit condition hence any input will go through it. Algorithm 2 has an exit condition at line 8. It can be easily verified that X cannot be empty if ℓ is a feasible path labeling. The reason is that a feasible path labeling has an associated bijection between $supp(\mathcal{F})$ and $V(T)$ i.e. $supp(\mathcal{F}^\ell)$ such that the sets map to paths, "preserving" the path labeling. The rest of the section is devoted to constructively proving that it is sufficient for a path labeling to be an ICPPL and pass the two filtering algorithms. To describe in brief, the constructive approaches refine an ICPPL iteratively, such that at the end of each iteration we have a "filtered" path labeling, and finally we have a path labeling that defines a family of bijections from $supp(\mathcal{F})$ to $V(T)$ i.e. $supp(\mathcal{F}^\ell)$.

First we present Algorithm 1 or Filter 1, and prove its correctness. This algorithm refines the path labeling by considering pairs of paths that share a leaf.

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

1: Let $\mathcal{F}_0 = \mathcal{F}$
2: Let $\ell_0(S) = \ell(S)$ for all $S \in \mathcal{F}_0$
3: $j = 1$
4: **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**
5: $\quad \mathcal{F}_j = (\mathcal{F}_{j-1} \setminus \{S_1, S_2\}) \cup \{S_1 \cap S_2, S_1 \setminus S_2, S_2 \setminus S_1\}$ | `Remove` S_1`,` S_2 `and add the ''filtered'' sets`
6: $\quad$ for all $S \in \mathcal{F}_{j-1}$ such that $S \neq S_1$ and $S \neq S_2$, set $\ell_j(S) = \ell_{j-1}(S)$ | `Do not change path labeling for any set other than` S_1`,` S_2
7: $\quad \ell_j(S_1 \cap S_2) = \ell_{j-1}(S_1) \cap \ell_{j-1}(S_2)$
8: $\quad \ell_j(S_1 \setminus S_2) = \ell_{j-1}(S_1) \setminus \ell_{j-1}(S_2)$
9: $\quad \ell_j(S_2 \setminus S_1) = \ell_{j-1}(S_2) \setminus \ell_{j-1}(S_1)$
10: $\quad$ **if** $(\mathcal{F}_j, \ell_j)$ does not satisfy condition (iii) of ICPPL **then**
11: $\quad\quad$ **exit**
12: $\quad$ **end if**
13: $\quad j = j + 1$
14: **end while**
15: $\mathcal{F}' = \mathcal{F}_j$, $\ell' = \ell_j$
16: **return** $\mathcal{F}', \ell'$

Lemma 3.2 *In Algorithm 1, if input $(\mathcal{F}, \ell)$ is a feasible path assignment then at the end of jth iteration of the* **while** *loop, $j \geq 0$, $(\mathcal{F}_j, \ell_j)$ is a feasible path assignment.*

Lemma 3.3 *In Algorithm 1, at the end of jth iteration, $j \geq 0$, of the* **while** *loop of Algorithm 1, the following invariants are maintained.*

I ※ $\ell_j(R)$ *is a path in* T, *for all* $R \in \mathcal{F}_j$

II ※ $|R| = |\ell_j(R)|$, *for all* $R \in \mathcal{F}_j$

III ※ $|R \cap R'| = |\ell_j(R) \cap \ell_j(R')|$, *for all* $R, R' \in \mathcal{F}_j$

IV ※ $|R \cap R' \cap R''| = |\ell_j(R) \cap \ell_j(R') \cap \ell_j(R'')|$,
\- *for all* $R, R', R'' \in \mathcal{F}_j$

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.

※ *Invariant* I/II

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

I/IIb | *R is a new set.* If R is in $\mathcal{F}_j$ and not in $\mathcal{F}_{j-1}$, then it must be one of the new sets added in $\mathcal{F}_j$. In this case, it is clear that for each new set, the image under ℓ_j is a path since by definition the chosen sets S_1, S_2 are from $\mathcal{F}_{j-1}$ and due to the while loop condition, $\ell_{j-1}(S_1)$, $\ell_{j-1}(S_2)$ have a common leaf. Thus invariant I is proven.
Moreover, due to induction hypothesis of invariant III ($j-1$th iteration) and the definition of l_j in terms of l_{j-1}, invariant II is indeed true in the jth iteration for any of the new sets.

※ *Invariant* III

IIIa | *R and R' are not new sets.* Trivially true by induction hypothesis.

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

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

※ *Invariant* IV

Due to the condition in line 10, invariant IV is ensured at the end of every iteration. ∎

Lemma 3.4 *If the input ICPPL $(\mathcal{F}, \ell)$ to Algorithm 1 is feasible, then the algorithm executes without a premature exit.*

Proof A premature exit occurs iff the exit condition in line 10, i.e. failure of three way intersection preservation, becomes true in any iteration of the **while** loop. Since $(\mathcal{F}, \ell)$ is feasible, there exists a hypergraph isomorphism $\phi : supp(\mathcal{F}) \rightarrow supp(\mathcal{F}^\ell)$ such that $\phi(x) = v$. Clearly, ϕ is a renaming of vertices in hypergraph $\mathcal{F}$ to those in hypergraph $\mathcal{F}^\ell$. Thus the following facts can be observed in every iteration of the loop.

i. all intersection cardinalities are preserved in this path labeling
ii. element x is exclusive in a hyperedge in $\mathcal{F}$ since v is exclusive in a hyperedge in $\mathcal{F}^\ell$.

Thus the exit condition is never rendered true after x and v are removed from their respective hyperedges. ∎

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

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

1: Let $\mathcal{F}_0 = \mathcal{F}$
2: Let $\ell_0(S) = \ell(S)$ for all $S \in \mathcal{F}_0$. Note: Path images are such that no two path images share a leaf.
3: $j = 1$
4: **while** there is a leaf v in T and a unique $S_1 \in \mathcal{F}_{j-1}$ such that $v \in \ell_{j-1}(S_1)$ **do**
5: $\quad \mathcal{F}_j = \mathcal{F}_{j-1} \setminus \{S_1\}$
6: $\quad$ for all $S \in \mathcal{F}_{j-1}$ such that $S \neq S_1$ set $\ell_j(S) = \ell_{j-1}(S)$
7: $\quad X = S_1 \setminus \bigcup_{S \in \mathcal{F}_{j-1}, S \neq S_1} S$
8: $\quad$ **if** X is empty **then**
9: $\quad\quad$ **exit**
10: $\quad$ **end if**
11: $\quad$ Let $x =$ arbitrary element from X
12: $\quad \mathcal{F}_j = \mathcal{F}_j \cup \{\{x\}, S_1 \setminus \{x\}\}$
13: $\quad \ell_j(\{x\}) = \{v\}$
14: $\quad \ell_j(S_1 \setminus \{x\}) = \ell_{j-1}(S_1) \setminus \{v\}$
15: $\quad j = j + 1$
16: **end while**
17: $\mathcal{F}' = \mathcal{F}_j$
18: $\ell' = \ell_j$
19: **return** $\mathcal{F}', \ell'$

Lemma 3.5 *In Algorithm 2, for all $j \geq 0$, at the end of the jth iteration the four invariants given in lemma 3.3 are valid.*

Proof Consider the false negative case of the input ICPPL $(\mathcal{F}, \ell)$ also being feasible, but set X is empty in some iteration of the **while** loop at line 8. This will prematurely exit the algorithm and thus prevent us from finding the permutation. We will now show by contradiction that this cannot happen. Assume X is empty for some iteration $j \geq 0$. 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 $v \in \ell_{j-1}(S_1) \setminus \bigcup_{(S \in \mathcal{F}_{j-1}) \wedge (S \neq S_1)} \ell_{j-1}(S)$. 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 in S_2 to those in $\ell_{j-1}(S_2)$. This contradicts our assumption that the input is feasible. Thus X cannot be empty if input is ICPPL and feasible. ∎

We have seen two filtering algorithms above, namely, algorithms 1 and 2 which finally result in a new ICPPL on the same universe U and tree T. We also proved that if the input is indeed feasible, these algorithms do not exit prematurely thus never outputs a false negative. Using these algorithms we now prove the following theorem.

Theorem 3.6 *If $(\mathcal{F}, \ell)$ is an ICPPL from a tree T that passes filter 1 and filter 2 (algorithm 1, algorithm 2, then there exists a hypergraph isomorphism $\phi : supp(\mathcal{F}) \to V(T)$ such that the ϕ-induced tree path labeling is equal to ℓ, $\ell_\phi = \ell$.*

Proof This is a contructive proof. We find ϕ part by part by running algorithms 1 and 2 one after the other in a loop. After each iteration we calculate an exclusive subset of the bijection ϕ, namely that which involves all the leaves of the tree in that iteration. Then all the leaves are pruned off the tree before the next iteration. The loop terminates when the pruned tree has no nodes at which point the rest of the bijection is obvious, thus completing ϕ. This is the brief outline of the algorithm. Now we see it in detail below.

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

$$\ell_1(S) = \ell'(S), \text{ when } \ell'(S) \text{ has non-leaf vertices}$$
$$\phi_1(x) = v, \text{ when } \ell'(S) = \{v\}, v \in V_1,$$
$$\text{and } S = \{x\}$$

Consider the tree $T_1 = T[V(T) \setminus V_0]$, i.e. it isomorphic to T with its leaves removed. Let U_1 be the universe of the subsystem that is not mapped to a leaf of T, i.e. $U_1 = supp(\mathcal{F}) \setminus Y_1$.

Let $\mathcal{F}_1$ be the set system induced by $\mathcal{F}'$ on universe U_1. Now we have a subproblem of finding the hypergraph isomorphism for $(\mathcal{F}_1, \ell_1)$ with tree T_1. Now we repeat Algorithm 1 followed by Algorithm 2 on $(\mathcal{F}_1, \ell_1)$ with tree T_1. As before we define l_2 in terms of l_1, ϕ_2 in terms of $V_2 = \{v \mid v \text{ is a leaf in } T_1\}$, prune the tree T_1 to get T_2 and so on. Thus in the ith iteration, T_i is the pruned tree, ℓ_i is a feasible path labeling to $\mathcal{F}_i$ if $(\mathcal{F}_{i-1}, \ell_{i-1})$ is feasible, ϕ_i is the leaf labeling of leaves of T_{i-1}. Continue this until some dth iteration for the smallest value d such that T_d has no nodes. From the lemma 3.3 and 3.5 we know that $(\mathcal{F}_d, \ell_d)$ is an ICPPL.

Now we have exactly one bijection ϕ_j, $j \in [d]$ defined for every element $x \in U$ into some vertex $v \in V(T)$. The bijection for the ICPPL, $\phi : U \to V(T)$ is constructed by the following definition.

$$\phi(x) = \phi_i(x)$$
$$\text{where } x \text{ is in the domain of } \phi_i, i \in [d]$$

It can be verified easily that ϕ is the required hypergraph isomorphism. Thus the theorem is proven. ∎

4 Finding tree path labeling from k-subdivided stars

When the given tree is restricted to a smaller class, namely k-subdivided stars and when the set system has only one overlap component, we have an algorithm which has polynomial time complexity.

For ease of notation and due to our focus here being only on a set system that is a single overlap component, we will denote the set system, k-subdivided star and the root of the star by $\mathcal{O}$, T and vertex r, respectively. We generalize the interval assignment algorithm for an overlap component from a prime matrix in [Narayanaswamy and Subashini, 2009] (algorithm 4 in their paper) to find tree path labeling for overlap component $\mathcal{O}$.

The outline of the algorithm is as follows. Notice that the path between a leaf v and r has the property that none of the vertices except r has degree greater than 2. Thus each ray excluding vertex r can be considered as independent intervals. So we begin by labeling of hyperedges to paths that have vertices from a single ray only and vertex r (the dependency due to r will be resolved in the algorithm eventually). Clearly this can be done using an interval assignment algorithm alone. This is done for each ray one after another till a condition for a *crossing hyperedge* is reached for each ray which is described below. This part of the algorithm is called *saturation of rays* and is repeated several times till all hyperedges are labeled. Prior to that, to begin this procedure a similar set of steps are carried out in *initialization of path labeling* for each ray when its leaves are arbitrarily assigned to marginal hyperedges (as defined in [Köbler et al., 2010]).

During the saturation procedure for any particular ray $R_i, i \in [p]$, we will reach a point in the algorithm where we cannot proceed further with an interval assignment procedure alone because in order to maintain the ICPPL properties of the hyperedge, one will require a path that "crosses" the center of the star to another ray R_j, $i \neq j, j \in [p]$ and maintining interval assignment properties cannot always deterministically tell us which ray that should be. Such a hyperedge is called *crossing hyperedge* of that ray. At this point we make the following observation about the classification of the hyperedges in $\mathcal{O}$ that have already been assigned to or can only be assigned to (due to the algorithmic choices made so far in constructing a feasible path label ℓ) paths from a particular ray $R_i = \{v \mid v \in V(T), v \text{ is in } i\text{th ray or } v = r\}$ only in T, $i \in [p]$ (recall that p is the number of rays in T).

Type 0/ labeled hyperedges: The hyperedges that have been labeled with paths contained in R_i.
$\mathcal{H}_0^i = \{H \mid \ell(H) \subseteq R_i \text{ where } H \in \mathcal{O}\}$

Type 1/ unlabeled contained hyperedges: The hyperedges that are contained in type 0 hyperedges of R_i.
$\mathcal{H}_1^i = \{H \mid H \subseteq H' \text{ where } H \in \mathcal{O}, H' \in \mathcal{H}_0^i\}$

Type 2/ unlabeled overlaping hyperedges: The hyperedges that overlap with at least one type 0 hyperedge associated with R_i but cannot be labeled with a path that is contained in R_i. It requires vertices from another ray also in its labeling if $\mathcal{O}$ has a feasible path labeling ℓ from T.
$\mathcal{H}_2^i = \{H \mid H \cap H' \neq \emptyset \text{ and } \nexists \ell(H) \subseteq R_i, \text{ where } H \in \mathcal{O}, H' \in \mathcal{H}_0^i\}$.

It must be noted that a *crossing hyperedge* is of type 2 which is encountered in every call to the saturation of rays procedure.

Since $\mathcal{O}$ is an overlap component, the type 1 hyperedges overlap with some type 2 hyperedge and can be handled as and when type 2 hyperedges are labeled using the intersection cardinality properties of ICPPL. Note that in the algorithm outlined above, we find a single crossing hyperedge per ray per call to saturation procedure. Consider a ray R_i and its corresponding crossing hyperedge B_i. Now we try to make a partial path labeling as follows. We partition the crossing hyperedge into two subsets $B_i = B_i' \cup B_i''$ such that $\ell(B_i') = P_i', \ell(B_i'') = P_i''$ which are defined as follows.

$$
\begin{aligned}
&P_i' \subseteq R_i \text{ such that,}\\
&\qquad r \in P_i',\\
&\qquad |P_i'| = k + 2 - |supp(\mathcal{H}_0^i)| + |supp(\mathcal{H}_0^i) \cap B_i|\\
&P_i'' \in \mathcal{Z}_i \text{ such that,}\\
&\qquad \mathcal{Z}_i = \{P_{i,j} \mid j \neq i, j \in [p]\} \text{ where,}\\
&\qquad P_{i,j} = \{v_{j,0} \mid v_{j,0} \text{ is adj. to } r \text{ on } R_j\} \cup\\
&\qquad\qquad \{v_{j,q} \mid \text{ vertex } v_{j,q} \text{ is adjacent to } v_{j,q-1},\\
&\qquad\qquad \text{where } q \in [|B_i| - |P_i'| - 1]\}\\
&P_i' \cup P_i'' \text{ is a path in } T
\end{aligned}
$$

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

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

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

Note that $P_b'' \neq P_{b,a}$ in observation 2 else it would have been covered in observation 1. Now we continue to find new crossing hyperedges on all rays until the path labeling is complete.

The algorithm is formally described as follows. Algorithm 3 is the main algorithm which uses algorithms 4, 5 and 6 as subroutines. The function $dist(u, v)$ returns the number of vertices between the vertices u and v on the path that connects them (including u and v).

5 Acknowlegements

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

References

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

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

Booth, K. S. and Lueker, G. S. [1976], '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.

Brandstädt, A., Le, V. B. and Spinrad, J. P. [1999], *Graph classes: a survey*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

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

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

$\mathcal{L}$ — `$\mathcal{L} \subseteq \mathcal{O}$ is a global variable for the set subsystem that has a path labeling so far. It is the domain of the feasible path labeling $\ell$ at any point in the algorithm.`

ℓ — `$\ell : \mathcal{L} \rightarrow \mathcal{P}$, is a global variable representing a feasible path labeling of $\mathcal{L}$ to some path system $\mathcal{P}$ of $T$. It is the partial feasible path labeling of $\mathcal{O}$ at any point in the algorithm.`

$initialize\_rays(\mathcal{O}, T)$

`Call algorithm 4 for initialization of rays. This is when a hyperedge is assigned to a path with the ray's leaf.`

while $\mathcal{L} \neq \mathcal{O}$ **do**

  $saturate\_rays\_and\_find\_crossing\_hyperedges(\mathcal{O}, T)$

  `Saturate all rays of $T$ by using algorithm 5. This subroutine also finds the crossing hyperedge $\mathcal{B}_i$ of each ray $i$. A crossing hyperedge is one that needs to be labeled to a path that has vertices from exactly two rays.`

  $partial\_path\_labeling\_of\_crossing\_hyperedges(\mathcal{O}, T)$

  `Find path labeling of crossing hyperedges by using algorithm 6. This subroutine finds the part of the blocked hyperedge's path label that comes from the second ray.`

end while

Algorithm 4 $initialize\_rays(\mathcal{O}, T)$

Let $\{v_i \mid i \in [p], p$ is number of leaves of $T\}$

`Also note $k + 2$ is the length of the path from the center to any leaf since $T$ is $k$-subdivided star.`

$\mathcal{K} \leftarrow \{H \mid H \in \mathcal{O},\ N(H)$ in $\mathcal{O}$ is a clique $\}$

`Local variable to hold the marginal hyperedges. A marginal hyperedge is one that has exactly one inclusion chain of interections with every set it overlaps with, i.e., its neighbours in the overlap graph form a clique.`

for every inclusion chain $C \subseteq \mathcal{K}$ **do**

  Remove from $\mathcal{K}$ all sets in C except the set $H_{C-icpia-max}$ which is the set closest to the maximal inclusion set H_{C-max} such that $|H_{C-icpia-max}| \leq k + 2$.

end for

if $|\mathcal{K}| > l$ **then**

  Exit. — `No labeling possible since $\mathcal{O}$ is an overlap component and $T$ does not have enough rays.`

end if

`$H_{C-icpia-max}$ does not exist for at least one ray. Labeling could still be possible because $H_{C-max}$ could be a viable crossing hyperedge itself. Hence proceed.`

$i \leftarrow 0$

for every hyperedge $H \in \mathcal{K}$ **do**

  $i \leftarrow i + 1$

  $\ell(H) \leftarrow P_i$ where P_i is the path in T containing leaf v_i such that $|P_i| = |H|$.

  $\mathcal{L} \leftarrow \mathcal{L} \cup \{H\}$

end for

Return the number of initialized rays, i.

Golumbic, M. C. [2004], *Algorithmic graph theory and perfect graphs*, Vol. 57 of *Annals of Discrete Mathematics*, Elsevier Science B.V. Second Edition.

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

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

Hsu, W.-L. [2001], 'PC-trees vs. PQ-trees', *Lecture Notes in Computer Science* **2108**, 207–217.

Hsu, W.-L. [2002], 'A simple test for the consecutive ones property', *J. Algorithms* **43**(1), 1–16.

Köbler, J., Kuhnert, S., Laubner, B. and Verbitsky, O. [2010], 'Interval graphs: Canonical representation in logspace', *Electronic Colloquium on Computational Complexity (ECCC)* **17**, 43.

Kou, L. T. [1977], 'Polynomial complete consecutive information retrieval problems', *SIAM Journal on Computing* **6**(1), 67–75.

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

McConnell, R. M. [2004], 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)'.

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

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

Peyton, B. W., Pothen, A. and Yuan, X. [1994], A clique tree algorithm for partitioning a chordal graph into transitive subgraphs, Technical report, Old Dominion University, Norfolk, VA, USA.

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

Algorithm 5 *saturate_ rays_ and_ find_ crossing_ hyperedge*$(\mathcal{O}, T)$

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

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

A Detailed proofs

Proof of Lemma 3.1 Let $P_i = \ell(S_i)$, for all $1 \leq i \leq 3$. $|S_1 \cap (S_2 \setminus S_3)| = |(S_1 \cap S_2) \setminus S_3| = |S_1 \cap S_2| - |S_1 \cap S_2 \cap S_3|$. Due to conditions (ii) and (iii) of ICPPL, $|S_1 \cap S_2| - |S_1 \cap S_2 \cap S_3| = |P_1 \cap P_2| - |P_1 \cap P_2 \cap P_3| = |(P_1 \cap P_2) \setminus P_3| = |P_1 \cap (P_2 \setminus P_3)|$. Thus lemma is proven. ∎

Proof of Lemma 3.2 We will prove this by mathematical induction on the number of iterations. The base case $(\mathcal{F}_0, \ell_0)$ is feasible since it is the input itself. Assume the lemma is true till $j-1$th iteration. i.e. there is a bijection $\phi : supp(\mathcal{F}_{j-1}) \rightarrow V(T)$ such that the induced path labeling on $\mathcal{F}_{j-1}$, $\ell_{\phi[\mathcal{F}_{j-1}]}$ is equal to ℓ_{j-1}. We will prove that ϕ is also the bijection that makes $(\mathcal{F}_j, \ell_j)$ feasible. Note that $supp(\mathcal{F}_{j-1}) = supp(\mathcal{F}_j)$ since the new sets in $\mathcal{F}_j$ are created from basic set operations to the sets in $\mathcal{F}_{j-1}$. For the same reason and ϕ being a bijection, it is clear that $\ell_{\phi[\mathcal{F}_j]}(S_1 \setminus S_2) = \ell_{\phi[\mathcal{F}_{j-1}]}(S_1) \setminus \ell_{\phi[\mathcal{F}_{j-1}]}(S_2)$. Now observe that $\ell_j(S_1 \setminus S_2) = \ell_{j-1}(S_1) \setminus \ell_{j-1}(S_2) = \ell_{\phi[\mathcal{F}_{j-1}]}(S_1) \setminus \ell_{\phi[\mathcal{F}_{j-1}]}(S_2)$. Thus the induced path labeling $\ell_{\phi[\mathcal{F}_j]} = \ell_j$. Therefore lemma is proven. ∎

Algorithm 6 *partial_ path_ labeling_ of_ crossing_ hyperedges*$(\mathcal{O}, T)$

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

Proof of Lemma 3.3 :

※ *Invariant* I/II

I/IIb | *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.

※ *Invariant III*

IIIb | *Only one, say 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 3.1
[8] ℓ_j definition. Note R' is not a new set

Proof of lemma 3.5 (alternate) 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 4.
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.

※ *Invariant* I/II

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

I/IIb | *R is a new set.* If R is in $\mathcal{F}_j$ and not in $\mathcal{F}_{j-1}$, then it must be one of the new sets added in $\mathcal{F}_j$. 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.

※ *Invariant* III

IIIa | *R and R' are not new sets.* Trivially true by induction hypothesis.

IIIb | *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.

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

※ *Invariant* IV

IVa | *R, R' and R'' are not new sets.* Trivially true by induction hypothesis.

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

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

∎