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

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

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

Abstract. In this paper, we explore a natural generalization of results on binary 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 $\phi : U \rightarrow V(T)$ 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 conclude with a polynomial time algorithm to find a test for a feasible tree path labeling of a given set system when T is a *k-subdivided star* when all the sets are of size at most $k + 2$.

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

1 Introduction

Consecutive ones property (COP) of binary matrices is a widely studied combinatorial problem. The problem is to rearrange rows (columns) of a binary matrix in such a way that every column (row) has its 1s occur consecutively. If this is possible the matrix is said to have the COP. This problem has several practical applications in diverse fields including scheduling [8], information retrieval [12] and computational biology [1]. Further, it is a tool in graph theory [6] for interval graph recognition, characterization of Hamiltonian graphs, and in integer linear programming [7, 8]. Recognition of COP is polynomial time solvable by several algorithms. PQ trees [3], variations of PQ trees [15, 9, 10, 14], ICPIA [16] are the main ones.

The problem of COP testing can be easily seen as a 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 [16] with a single permutation of the universe (set of row indices) which permutes each set to its interval. The result in [16] characterizes 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 intersection cardinality of the corresponding intervals. While this is naturally a necessary condition, [16] 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 [10] 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 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 set system, target tree pair – $(\mathcal{F}, T)$. As a special case if the tree T is a path, the problem becomes the interval assignment problem. We focus on the question of generalizing the notion of an ICPIA [16] 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 feasible 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 passes a filtering algorithm (described in this paper) successfully. This characterization is proved constructively and it gives a natural data structure that stores all the relevant feasible 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 [16].

Our results also have close connection to recognition of *path graphs* and connection to path graph isomorphism. The reason is that given a hypergraph $\mathcal{F}$, it can be viewed as paths in a tree, if and only if the intersection graph of $\mathcal{F}$ is *path graph.* Path graphs are a subclass of chordal graphs which are combinatorially characterized as the intersection graphs of subtrees of a tree. These results are well studied in the literature in [18, 5, 2, 6]. Chordal graphs which can be represented as the intersection graph of paths in a tree are called path graph [6]. Checking if a graph is a path graph can be done in polynomial time by the results of [5, 19]. However, this is only a necessary condition for our question. Indeed, our work provides a sufficient condition for a related problem: that is whether a path assignment to be feasible? On the other hand, path graph isomorphism is known be isomorphism-complete, see for example [11]. Therefore, it is unlikely that we can solve the for a feasible path labeling ℓ for a given $\mathcal{F}$ and tree T. It is definitely interesting to classify the kinds of trees and hypergraphs for which feasible path labelings can be found efficiently. These results would form a natural generalization of COP testing and interval graph isomorphism, culminating in Graph Isomorphism itself.

Our Results:

1. Given a path labeling ℓ to $\mathcal{F}$ from a tree T, we give a necessary and sufficient condition for it to be a feasible path labeling. This necessary and sufficient condition can be testing in polynomial time. The most interesting consequence is that in our constructive procedure, it is sufficient to iteratively check if three-way intersection cardinalities are preserved. In other words, in each iteration, it is sufficient to check if the intersection any three sets is of the same cardinality as the intersection of the corresponding paths (we refer to this at some places as the test for preservation fo three-way intersection cardinalities). This generalizes the well studied question of the case when the given tree T is a path [10, 16].
2. In the Section 4, we initiate an exploration of finding feasible path labeling of set systems in a special kind of tree which we call the k-subdivided star. This question is an attempt to generalize the problem of testing if a matrix the consecutive ones property. However, we restrict the hypergraph $\mathcal{F}$ to be such that all hyperedges have at most $k+2$ elements. In spite of this restricted case we consider, we believe that our results are of significant interest in understanding the nature of Graph Isomorphism which is polynomial time solvable in interval graphs and is hard on path graphs.

Roadmap. The necessary preliminaries with definitions etc. are presented in Section 2. Section 3 documents the characterization of a feasible path labeling and finally, Section 4 describes a polynomial time algorithm to find the tree path labeling of a given set system from a given k-subdivided tree.

2 Preliminaries

Hypergraph Preliminaries: 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 $\mathcal{F}$; $supp(\mathcal{F}) = \bigcup_{S \in \mathcal{F}} S$. For the purposes of this paper, a set system is required to "cover" the universe; $supp(\mathcal{F}) = U$. A set system $\mathcal{F}$ can also be visualized as a *hypergraph* vertex set is $supp(\mathcal{F})$ and hyperedges are the sets in $\mathcal{F}$. The *intersection graph* $\mathbb{I}(\mathcal{F})$ of a hypergraph $\mathcal{F}$ 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 hyperedges have a non-empty intersection [6]. Two hypergraphs $\mathcal{F}'$, $\mathcal{F}''$ are said to be *isomorphic* to each other, denoted by $\mathcal{F}' \cong \mathcal{F}''$, iff there exists a bijection $\phi : supp(\mathcal{F}') \rightarrow supp(\mathcal{F}'')$ such that for all sets $A \subseteq supp(\mathcal{F}')$, A is a hyperedge in $\mathcal{F}'$ iff B is a hyperedge in $\mathcal{F}''$ where $B = \{\phi(x) \mid x \in A\}$ [11], written as $B = \phi(A)$.

Path Hypergraph from a Tree: We consider trees T such that $|V(T)| = |U| = n$. A *path system* $\mathcal{P}$ is a set system of paths from T; $\mathcal{P} \subseteq \{P \mid P \subseteq V,\ T[P] \text{ is a path}\}$. This generalizes the fact, from the literature [4, 11], that intervals can be viewed as sub-paths of a path. If the intersection graphs of $\mathcal{F}$ and $\mathcal{P}$(a path system) are isomorphic, $\mathbb{I}(\mathcal{F}) \cong \mathbb{I}(\mathcal{P})$, then the associated bijection $\ell : \mathcal{F} \rightarrow \mathcal{P}$ due to this isomorphism is called a *path labeling* of the hypergraph $\mathcal{F}$. Note that there are two kinds of isomorphisms here. We are concerned about the isomrophisms intersection graphs on $\mathcal{F}$ and $\mathcal{P}$, and also the isomorphism between the hypergraph $\mathcal{F}$ and $\mathcal{P}$. If $\mathcal{F} \cong \mathcal{P}$ where $\mathcal{P}$ is a path system, then $\mathcal{F}$ is called a *path hypergraph* and $\mathcal{P}$ is called *path representation* of $\mathcal{F}$. If this isomorphism is $\phi : supp(\mathcal{F}) \rightarrow V(T)$, then it is clear that there is a path labeling $\ell_\phi : \mathcal{F} \rightarrow \mathcal{P}$ to the set system; $\ell_\phi(S) = \{y \mid y = \phi(x), x \in S\}$ for all $S \in \mathcal{F}$. In other words, if $\mathcal{F} \cong \mathcal{P}$, we get a path labeling. Recall that $supp(\mathcal{P}) = V(T)$. In this work, we are given as input $\mathcal{F}$ and a tree T, and the question is whether there is a path labeling ℓ to a set of paths in T. We refer to such a solution path system by $\mathcal{F}^\ell$. A path labeling $(\mathcal{F}, \ell)$ is defined to be *feasible* if there is a hypergraph isomorphism $\phi : supp(\mathcal{F}) \rightarrow supp(\mathcal{F}^\ell) = V(T)$ induces a path labeling $\ell_\phi : \mathcal{F} \rightarrow \mathcal{F}^\ell$ such that $\ell_\phi = \ell$.

Overlap Graphs and Marginal Hyperedges: An *overlap graph* $\mathbb{O}(\mathcal{F})$ of a hypergraph $\mathcal{F}$ is a graph such that its vertex set has a bijection to $\mathcal{F}$ and there exists an edge between two of its vertices iff their corresponding hyperedges overlap. Two hyperedges S and S' are said to *overlap*, denoted by $S \between S'$, if they have a non-empty intersection and neither is contained in the other; $S \between S'$ iff $S \cap S' \neq \emptyset, S \nsubseteq S', S' \nsubseteq S$. Thus $\mathbb{O}(\mathcal{F})$ is a spanning subgraph of $\mathbb{I}(\mathcal{F})$ and not necessarily connected. Each connected component of $\mathbb{O}(\mathcal{F})$ is called an *overlap component*. A hyperedge $S \in \mathcal{F}$ is called *marginal* if for all $S' \between S$, the overlaps $S \cap S'$ form a single inclusion chain [11]. Additionally, if S is such that it is contained in no other marginal hyperedge in $\mathcal{F}$, then it is called *super-marginal.*

k-subdivided star – a special tree A *star* graph is a complete bipartite graph $K_{1,p}$ which is clearly a tree and p is the number of leaves. The vertex with maximum degree is called the *center* of the star and the edges are called *rays* of the star. A *k-subdivided star* is a star with all its rays subdivided exactly k times. The definition of a *ray of a k-subdivided star* is extended to the path from the center to a leaf. It is clear that all rays are of length $k + 2$.

3 Characterization of Feasible Tree Path Labelings

In this section we give an algorithmic characterization of a feasibility of tree path labeling. Consider a path labeling $(\mathcal{F}, \ell)$ on the given tree T. We call $(\mathcal{F}, \ell)$ an *Intersection Cardinality Preserving Path Labeling (ICPPL)* if it has the following properties.

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

The following lemma is useful in subsequent arguments.

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

Proof. Let $P_i = \ell(S_i)$, for all $1 \le i \le 3$. $|S_1 \cap (S_2 \setminus S_3)| = |(S_1 \cap S_2) \setminus S_3| = |S_1 \cap S_2| - |S_1 \cap S_2 \cap S_3|$. Due to properties (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 proved. □

In the remaining part of this section we show that $(\mathcal{F}, \ell)$ is feasible if and only if it is an ICPPL and Algorithm 3 returns a non-empty function. Algorithm 3 recursively does two levels of filtering of $(\mathcal{F}, \ell)$ to make it simpler while retaining the set of isomorphisms, if any, between $\mathcal{F}$ and $\mathcal{F}^\ell$. First, we present Algorithm 1 or `filter_1`, and prove its correctness. This algorithm refines the path labeling by processing pairs of paths in $\mathcal{F}^\ell$ that share a leaf until no two paths in the new path labeling share any leaf.

Algorithm 1 Refine ICPPL `filter_1`$(\mathcal{F}, \ell, T)$

```
1: $\mathcal{F}_0 \leftarrow \mathcal{F}$, $\ell_0(S) \leftarrow \ell(S)$ for all $S \in \mathcal{F}_0$
2: $j \leftarrow 1$
3: 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
4:     $\mathcal{F}_j \leftarrow (\mathcal{F}_{j-1} \setminus \{S_1, S_2\}) \cup \{S_1 \cap S_2, S_1 \setminus S_2, S_2 \setminus S_1\}$    |Remove $S_1$, $S_2$ and add the ``filtered'' sets
5:     for every $S \in \mathcal{F}_{j-1}$ s.t. $S \neq S_1$ and $S \neq S_2$ do $\ell_j(S) \leftarrow \ell_{j-1}(S)$ end for
6:     $\ell_j(S_1 \cap S_2) \leftarrow \ell_{j-1}(S_1) \cap \ell_{j-1}(S_2)$    |Carry forward the path labeling for all existing sets other than $S_1$, $S_2$
7:     $\ell_j(S_1 \setminus S_2) \leftarrow \ell_{j-1}(S_1) \setminus \ell_{j-1}(S_2)$    |Define path labeling for new sets
8:     $\ell_j(S_2 \setminus S_1) \leftarrow \ell_{j-1}(S_2) \setminus \ell_{j-1}(S_1)$
9:     if $(\mathcal{F}_j, \ell_j)$ does not satisfy (Property iii) of ICPPL then
10:        exit
11:    end if
12:    $j \leftarrow j + 1$
13: end while
14: $\mathcal{F}' \leftarrow \mathcal{F}_j$, $\ell' \leftarrow \ell_j$
15: 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, $j \ge 0$, $(\mathcal{F}_j, \ell_j)$ is a feasible path assignment.*

Proof. We will prove this by mathematical induction on the number of iterations. The base case $(\mathcal{F}_0, \ell_0)$ is feasible since it is the input itself which is given to be feasible. Assume the lemma is true till $j-1$th iteration. i.e. every hypergraph isomorphism $\phi : supp\,(\mathcal{F}_{j-1}) \to V\,(T)$ that defines $(\mathcal{F}, \ell)$'s feasibility, is 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 when applying the ϕ induced path labeling on $\mathcal{F}_j$, $\ell_{\phi[\mathcal{F}_j]}(S_1 \setminus S_2) = \ell_{\phi[\mathcal{F}_{j-1}]}(S_1) \setminus \ell_{\phi[\mathcal{F}_{j-1}]}(S_2)$. 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 proved. □

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

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

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

※ *Invariant* I/II

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

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

※ *Invariant* III

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

IIIb | *Only one, say R, is a new set.* Due to invariant IV induction hypothesis, Lemma 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. If $R = S_1 \cap S_2$, $\lvert R \cap R'\rvert = \lvert S_1 \cap S_2 \cap R'\rvert = \lvert\ell_{j-1}(S_1) \cap \ell_{j-1}(S_2) \cap \ell_{j-1}(R')\rvert = \lvert\ell_j(S_1 \cap S_2) \cap \ell_j(R')\rvert = \lvert\ell_j(R) \cap \ell_j(R')\rvert$. If $R = S_1 \setminus S_2$, $\lvert R \cap R'\rvert = \lvert(S_1 \setminus S_2) \cap R'\rvert = \lvert(\ell_{j-1}(S_1) \setminus \ell_{j-1}(S_2)) \cap \ell_{j-1}(R')\rvert = \lvert\ell_j(S_1 \cap S_2) \cap \ell_j(R')\rvert = \lvert\ell_j(R) \cap \ell_j(R')\rvert$. Similarly, if $R = S_2 \setminus S_1$. Note R' is not a new set.

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

※ *Invariant* IV

Due to the condition in line 9, this invariant 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 set of hypergraph isomorphism functions that defines $(\mathcal{F}, \ell)$'s feasibility is the same as the set that defines $(\mathcal{F}_j, \ell_j)$'s feasibility, if any. Secondly, for any iteration $j > 0$ of the* **while** *loop, the* **exit** *statement in line 10 will not execute.*

Proof. Since $(\mathcal{F}, \ell)$ is feasible, by Lemma 3.2 $(\mathcal{F}_j, \ell_j)$ for every iteration $j > 0$ is feasible. Also, every hypergraph isomorphism $\phi : supp(\mathcal{F}) \rightarrow V(T)$ that induces ℓ on $\mathcal{F}$ also induces ℓ_j on $\mathcal{F}_j$, i.e., $\ell_{\phi[\mathcal{F}_j]} = \ell_j$. Thus it can be seen that for all $x \in supp(\mathcal{F})$, for all $v \in V(T)$, if $(x, v) \in \phi$ then $v \in \ell_j(S)$ for all $S \in \mathcal{F}_j$ such that $x \in S$. In other words, filter 1 outputs a filtered path labeling that "preserves" hypergraph isomorphisms of the original path labeling.
Secondly, line 10 will execute iff the exit condition in line 9, i.e. failure of three way intersection preservation, becomes true in any iteration of the **while** loop. Due to Lemma 3.3 Invariant IV, the exit condition does not occur if the input is a feasible ICPPL. □

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

Suppose the input ICPPL $(\mathcal{F}, \ell)$ is feasible, yet set X in Algorithm 2 is empty in some iteration of the **while** loop. This will abort our procedure of finding the hypergraph isomorphism. The following lemma shows that this will not happen.

Lemma 3.5. *If the input ICPPL $(\mathcal{F}, \ell)$ to Algorithm 2 is feasible, then for all iterations $j > 0$ of the* **while** *loop, the* **exit** *statement in line 8 does not execute.*

Proof. Assume X is empty for some iteration $j > 0$. 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 hypergraph isomorphism bijection 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. □

Lemma 3.6. *In Algorithm 2, for all $j > 0$, at the end of the jth iteration of the* **while** *loop the four invariants given in Lemma 3.3 hold.*

Algorithm 2 Leaf labeling from an ICPPL `filter_2`$(\mathcal{F}, \ell, T)$

```
1: $\mathcal{F}_0 \leftarrow \mathcal{F}$, $\ell_0(S) \leftarrow \ell(S)$ for all $S \in \mathcal{F}_0$        |Path images are such that no two path
                                                                                                            |images share a leaf.
2: $j \leftarrow 1$
3: 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
4:     $\mathcal{F}_j \leftarrow \mathcal{F}_{j-1} \setminus \{S_1\}$
5:     for all $S \in \mathcal{F}_{j-1}$ such that $S \neq S_1$ set $\ell_j(S) \leftarrow \ell_{j-1}(S)$
6:     $X \leftarrow S_1 \setminus \bigcup_{S \in \mathcal{F}_{j-1}, S \neq S_1} S$
7:     if $X$ is empty then
8:         exit
9:     end if
10:    $x \leftarrow$ arbitrary element from $X$
11:    $\mathcal{F}_j \leftarrow \mathcal{F}_j \cup \{\{x\}, S_1 \setminus \{x\}\}$
12:    $\ell_j(\{x\}) \leftarrow \{v\}$
13:    $\ell_j(S_1 \setminus \{x\}) \leftarrow \ell_{j-1}(S_1) \setminus \{v\}$
14:    $j \leftarrow j + 1$
15: end while
16: $\mathcal{F}' \leftarrow \mathcal{F}_j$, $\ell' \leftarrow \ell_j$
17: return $(\mathcal{F}', \ell')$
```

Proof. By Lemma 3.5 we know that set X will not be empty in any iteration of the **while** loop if input ICPPL $(\mathcal{F}, \ell)$ is feasible and ℓ_j is always computed for all $j > 0$. Also note that removing a leaf from any path keeps the new path connected. Thus invariant I is obviously true. In every iteration $j > 0$, we remove exactly one element x from one set S in $\mathcal{F}$ and exactly one vertex v which is a leaf from one path $\ell_{j-1}(S)$ in T. This is because as seen in Lemma 3.5, x is exclusive to S and v is exclusive to $\ell_{j-1}(S)$. Due to this fact, it is clear that the intersection cardinality equations do not change, i.e., invariants II, III, IV remain true. On the other hand, if the input ICPPL is not feasible the invariants are vacuously true. $\square$

We have seen two filtering algorithms above, namely, Algorithm 1 `filter_1` and Algorithm 2 `filter_2` which when executed serially respectively 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 indeed output the filtered ICPPL. Now we present the algorithmic characterization of a feasible tree path labeling by way of Algorithm 3.

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

Lemma 3.7. *If $(\mathcal{F}, \ell)$ is an ICPPL from a tree T and Algorithm 3,* `get-hypergraph-isomorphism` *$(\mathcal{F}, \ell, T)$ returns a non-empty function, then there exists a hypergraph isomorphism $\phi : supp(\mathcal{F}) \rightarrow V(T)$ such that the ϕ-induced tree path labeling is equal to ℓ or $\ell_\phi = \ell$.*

Proof. It is clear that in the end of every recursive call to Algorithm 3, the function ϕ' is one to one involving all the leaves in the tree passed as input to that recursive call. Moreover, by Lemma 3.4

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

```
1: if $T$ is empty then
2:     return $\emptyset$
3: end if
4: $L \leftarrow \{v \mid v \text{ is a leaf in } T\}$
5: $(\mathcal{F}_1, \ell_1) \leftarrow$ filter_1$(\mathcal{F}, \ell, T)$
6: $(\mathcal{F}_2, \ell_2) \leftarrow$ filter_2$(\mathcal{F}_1, \ell_1, T)$
7: $(\mathcal{F}', \ell') \leftarrow (\mathcal{F}_2, \ell_2)$
8: $\phi' \leftarrow \emptyset$
9: for every $v \in L$ do
10:     $\phi'(x) \leftarrow v$ where $x \in \ell_2^{-1}(\{v\})$        |Copy the leaf labels to a one to one
                                                                     |function $\phi' : supp(\mathcal{F}) \rightarrow L$
11:     Remove $\{x\}$ and $\{v\}$ from $\mathcal{F}'$, $\ell'$ appropriately
12: end for
13: $T' \leftarrow T \setminus L$
14: $\phi'' \leftarrow$ get-hypergraph-isomorphism$(\mathcal{F}', \ell', T')$
15: $\phi \leftarrow \phi'' \cup \phi'$
16: return $\phi$
```

and Lemma 3.5 it is consistent with the tree path labeling ℓ passed. The tree pruning is done by only removing leaves in each call to the function and is done till the tree becomes empty. Thus the returned function $\phi : supp(\mathcal{F}) \rightarrow V(T)$ is a union of mutually exclusive one to one functions exhausting all vertices of the tree. In other words, it is a bijection from $supp(\mathcal{F})$ to $V(T)$ inducing the given path labeling ℓ and thus a hypergraph isomorphism. □

Theorem 3.8. *A path labeling $(\mathcal{F}, \ell)$ on tree T is feasible iff it is an ICPPL and Algorithm 3 with $(\mathcal{F}, \ell, T)$ as input returns a non-empty function.*

Proof. From Lemma 3.7, we know that if $(\mathcal{F}, \ell)$ is an ICPPL and Algorithm 3 with $(\mathcal{F}, \ell, T)$ as input returns a non-empty function, $(\mathcal{F}, \ell)$ is feasible. Now consider the case where $(\mathcal{F}, \ell)$ is feasible. i.e. there exists a hypergraph isomorphism ϕ such that $\ell_\phi = \ell$. Lemma 3.4 and Lemma 3.5 show us that filter 1 and filter 2 do not exit if input is feasible. Thus Algorithm 3 returns a non-empty function. □

ICPPL when given tree is a path: Consider a special case of ICPPL with the following properties when the tree T is a path. Hence, all path labels are can be viewed as intervals assigned to the sets in $\mathcal{F}$. It is shown, in [16], that the filtering algorithms outlined above need only preserve pairwise intersection cardinalities, and higher level intersection cardinalities are preserved by the Helly Property of intervals. Consequently, the filter algorithms do not need to ever evaluate the additional check to **exit**. This structure and its algorithm is used in the next section for finding tree path labeling from a k-subdivided star due to this graph's close relationship with intervals.

4 Case of a Special Tree – The k-subdivided star

In this section we consider the problem of assigning paths from a k-subdivided star T to a given set system $\mathcal{F}$ such that each set $X \in \mathcal{F}$ is of cardinality at most $k + 2$. Secondly, we present our results only for the case when overlap graph $\mathbb{O}(\mathcal{F})$ is connected. A connected component of $\mathbb{O}(\mathcal{F})$ is called an overlap component of $\mathcal{F}$. An interesting property of the overlap components is that any two distinct overlap components, say $\mathcal{O}_1$ and $\mathcal{O}_2$, are such that any two sets $S_1 \in \mathcal{O}_1$ and $S_2 \in \mathcal{O}_2$

are disjoint, or, w.l.o.g, all the sets in $\mathcal{O}_1$ are contained within one set in $\mathcal{O}_2$. This containment relation naturally determines a decomposition of the overlap components into rooted containment trees. We consider the case when there is only one rooted containment tree, and we first present our algorithm when $\mathbb{O}(\mathcal{F})$ is connected. It is easy to see that once the path labeling to the overlap component in the root of the containment tree is achieved, the path labeling to the other overlap components in the rooted containment tree is essentially finding a path labeling when the target tree is a path: each target path is a path that is allocated to sets in the root overlap component. Therefore, for the rest of this section, $\mathbb{O}(\mathcal{F})$ is a connected graph. Recall that we also consider the special case when all hyperedges are of cardinality at most $k+2$. By definition, a k-subdivided star has a central vertex which we call the *root*, and each root to leaf path is called a *ray*. First, we observe that by removing the root r from T, we get a collection of p vertex disjoint paths of length $k+1$, p being the number of leaves in T. We denote the rays by $R_1, \ldots, R_p$, and the number of vertices in R_i, $i \in [p]$ is $k+2$. Let $\langle v_{i1}, \ldots, v_{i(k+2)} = r \rangle$ denote the sequence of vertices in R_i, where v_{i1} is the leaf. Note that r is a common vertex to all R_i.

4.1 Description of the Algorithm

In this section the given hypergraph $\mathcal{F}$, the k-subdivided star and the root of the star are denoted by $\mathcal{O}$, T and vertex r, respectively. In particular, note that the vertices of $\mathcal{O}$ correspond to the sets in $\mathcal{F}$, and the edges correspond to the overlap relation.
For each hyperedge $X \in \mathcal{O}$, we will maintain a 2-tuple of non-negative numbers $\langle p_1(X), p_2(X) \rangle$. The numbers satisfy the property that $p_1(X) + p_2(X) \leq |X|$, and at the end of path labeling, for each X, $p_1(X) + p_2(X) = |X|$. This signifies the algorithm tracking the lengths of subpaths of the path assigned to X from at most two rays. We also maintain another parameter called the *residue* of X denoted by $s(X) = |X| - p_1(X)$. This signifies the residue path length that must be assigned to X which must be from another ray. For instance, if X is labeled a path from only one ray, then $p_1(X) = |X|$, $p_2(X) = 0$ and $s(X) = 0$.

The algorithm proceeds in iterations, and in the i-th iteration, $i > 1$, a single hyperedge X that overlaps with a hyperedge that has been assigned a path is considered. At the beginning of each iteration hyperedges of $\mathcal{O}$ are classifed into the following disjoint sets.

$\mathcal{L}_1^i$ *Labeled without* r. Those that have been labeled with a path which does not contain r in one of the previous iterations.
$\mathcal{L}_1^i = \{X \mid p_1(X) = |X| \text{ and } p_2(X) = 0 \text{ and } s(X) = 0, X \in \mathcal{O}\}$

$\mathcal{L}_2^i$ *Labeled with* r. Those that have been labeled with two subpaths of $\ell(X)$ containing r from two different rays in two previous iterations.
$\mathcal{L}_2^i = \{X \mid 0 < p_1(X), p_2(X) < |X| = p_1(X) + p_2(X) \text{ and } s(X) = 0, X \in \mathcal{O}\}$

$\mathcal{T}_1^i$ *Type 1 / partially labeled.* Those that have been labeled with one path containing r from a single ray in one of the previous iterations. Here, $p_1(X)$ denotes the length of the subpath of $\ell(X)$ that X has been so far labeled with.
$\mathcal{T}_1^i = \{X \mid 0 < p_1(X) < |X| \text{ and } p_2(X) = 0 \text{ and } s(X) = |X| - p_1(X), X \in \mathcal{O}\}$

$\mathcal{T}_2^i$ *Type 2 / not labeled.* Those that have not been labeled with a path in any previous iteration.
$\mathcal{T}_2^i = \{X \mid p_1(X) = p_2(X) = 0 \text{ and } s(X) = |X|, X \in \mathcal{O}\}$

The set $\mathcal{O}_i$ refers to the set of hyperedges $\mathcal{T}_1^i \cup \mathcal{T}_2^i$ in the ith iteration. Note that $\mathcal{O}_1 = \mathcal{O}$. In the ith iteration, hyperedges from $\mathcal{O}_i$ are assigned paths from T using the following rules. Also the end of the iteration, $\mathcal{L}_1^{i+1}, \mathcal{L}_2^{i+1}, \mathcal{T}_1^{i+1}, \mathcal{T}_2^{i+1}$ are set to $\mathcal{L}_1^i, \mathcal{L}_2^i, \mathcal{T}_1^i, \mathcal{T}_2^i$ respectively, along with some case-specific changes mentioned in the rules below.

I. **Iteration 1:** Let $S = \{X_1, \ldots, X_s\}$ denote the super-marginal hyperedges from $\mathcal{O}_1$. If $|S| = s \neq p$, then exit reporting failure. Else, assign to each $X_j \in S$, the path from R_j such that the path contains the leaf in R_j. This path is refered to as $\ell(X_j)$. Set $p_1(X_j) = |X|, p_2(X_j) = s(X_j) = 0$. Hyperedges in S are not added to $\mathcal{O}_2$ but are added to $\mathcal{L}_1^2$ and all other hyperedges are added to $\mathcal{O}_2$.

II. **Iteration i:** Let X be a hyperedge from $\mathcal{O}_i$ such that there exists $Y \in \mathcal{L}_1^i \cup \mathcal{L}_2^i$ and $X \between Y$. Further let $Z \in \mathcal{L}_1^i \cup L_2^i$ such that $Z \between Y$. If $X \in \mathcal{T}_2^i$, and if there are multiple Y candidates then any Y is selected. On the other hand, if $X \in \mathcal{T}_1^i$, then X has a partial path assignment, $\ell'(X)$ from a previous iteration, say from ray R_j. Then, Y is chosen such that $X \cap Y$ has a non-empty intersection with a ray different from R_j. The key things that are done in assigning a path to X are as follows. The *end* of path $\ell(Y)$ where $\ell(X)$ would overlap is found, and then based on this the existence of a feasible assignment is decided. It is important to note that since $X \between Y$, $\ell(X) \between \ell(Y)$ in any feasible assignment. Therefore, the notion of the *end* at which $\ell(X)$ and $\ell(Y)$ overlap is unambiguous, since for any path, there are two end points.

   (a) *End point of $\ell(Y)$ where $\ell(X)$ overlaps depends on $X \cap Z$:* If $X \cap Z \neq \emptyset$, then $\ell(X)$ has an overlap of $|X \cap Y|$ at that end of $\ell(Y)$ at which $\ell(Y)$ and $\ell(Z)$ overlap. If $X \cap Z = \emptyset$, then $\ell(X)$ has an overlap of $|X \cap Y|$ at that end of $\ell(Y)$ where $\ell(Y)$ and $\ell(Z)$ do not intersect.

   (b) *Any path of length $s(X)$ at the appropriate end contains r:* If $X \in \mathcal{T}_1^i$ then after finding the appropriate end as in step IIa this the unique path of length $s(X)$ should end at r. If not, we exit reporting failure. Else, $\ell(X)$ is computed as union of $\ell'(X)$ and this path. If any three-way intersection cardinality is violated with this new assignment, then exit, reporting failure. Otherwise, X is added to $\mathcal{L}_2^{i+1}$. On the other hand, if $X \in \mathcal{T}_2^i$, then after step IIa, $\ell(X)$ or $\ell'(X)$ is unique up to the root and including it. Clearly, the vertices $\ell(X)$ or $\ell'(X)$ contains depends on $|X|$ and $|X \cap Y|$. If any three way intersection cardinality is violated due to this assignment, exit, reporting failure. Otherwise, $p_1(X)$ is updated as the length of the assigned path, and $s(X) = |X| - p_1(X)$. If $s(X) > 0$, then X is added to $\mathcal{T}_1^{i+1}$. If $s(X) = 0$, then X is added to $\mathcal{L}_1^{i+1}$.

   (c) *The unique path of length $s(X)$ overlapping at the appropriate end of Y does not contain r:* In this case, $\ell(X)$ is updated to include this path. If any three way intersection cardinality is violated, exit, reporting failure. Otherwise, update $p_1(X)$ and $p_2(X)$ are appropriate, X is added to $\mathcal{L}_1^{i+1}$ or $\mathcal{L}_2^{i+1}$, as appropriate.

Proof of Correctness and Analysis of Running Time: It is clear that the algorithm runs in polynomial time, as at each step, at most three-way intersection cardinalities need to be checked. Further, finding super-marginal hyperedges can also be done in polynomial time, as it involves considering the overlap regions and checking if the inclusion partial order contains a single minimal element. In particular, once the super-marginal edges are identified, each iteration involes finding the next hyperedge to consider, and testing for a path to that hyperedge. To identify the next hyperedge to consider, we consider the breadth first layering of the hyperedges with the zeroeth layer consisting of the super-marginal hyperedges. Since $\mathcal{O}$ is connected, it follows that all hyperedges of $\mathcal{O}$ will be considered by the algorithm. Once a hyperedge is considered, the path to be assigned to it can also be computed in constant time. In particular, in the algorithm the path to be assigned to X depends on $\ell(Y), \ell(Z)$, $s(X)$ and the presence or absence of r in the candidate partial path $\ell'(X)$. Therefore, once the super-marginal edges are identified, the running time of the algorithm is linear in the size

of the input. By the technique used for constructing prime matrices [10], the super-marginal edges can be found in linear time in the input size. Therefore, the algorithm can be implemented to run in linear time in the input size.
The proof of correctness uses the following main properties:

1. The k-subdivided star has a very symmetric structure. This symmetry is quantified based on the following observation – either there are no feasible path labelings of $\mathcal{O}$ using paths from T, or there are exactly $p!$ feasible path labelings. In other words, there is either no feasible assignment, or effectively a unique assignment modulo symmetry.
2. The p super-marginal hyperedges, if they exist, will each be assigned a path from distinct rays, and each such path contains the leaf.
3. For a candidate hyperedge X, the partial path assignment $\ell'(X)$ is decided by its overlap with $\ell(Y)$ and cardinality of intersection with $\ell(Z)$.

These properties are formalized as follows:

Lemma 4.1. *If $X \in \mathcal{F}$ is super-marginal and ℓ is a feasible tree path labeling to tree T, then $\ell(X)$ will contain a leaf in T.*

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

Lemma 4.2. *If $\mathcal{O}$ does not have any super-marginal edges, then in any feasible path labeling ℓ of $\mathcal{O}$ with paths from T is such that, for any hyperedge X for which $\ell(X)$ contains a leaf, $|X| \geq k+3$.*

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

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

Lemma 4.3. *If there is a feasible path labeling for $\mathcal{O}$ in T, then there are exactly p super-marginal hyperedges.*

These lemmas now are used to prove the following theorem.

Theorem 4.4. *Given $\mathcal{O}$ and a k-subdivided star T, the above algorithm decides correctly if there is a feasible path labeling ℓ.*

Proof. Outline. If the algorithm outputs a path labeling ℓ, then it is clear that it is an ICPPL. The reason is that the algorithm checks that three-way intersection cardinalities are preserved in

each iteration which ensures ICPPL Property iii. Moreover, it is clear that $\ell(X)$ for any $X \in \mathcal{O}$ is computed by maintaining ICPPL Property i and ICPPL Property ii. For such a labeling ℓ, the proof that it is feasible is by induction on k. What needs to be shown is that Algorithm 3 successfully runs on input $(\mathcal{O}, \ell)$. In base case $k = 0$, T is a star. Also every set is at most size 2 $(k + 2)$ size and thus overlaps are at most 1. If two paths share a leaf in `filter_1` one must be of length 2 and the other of length 1. Thus the exit condition is not met. Further, it is also clear that the exit condition in `filter_2` is also not met. Thus claim proven for base case. Now assume the claim to be true when target tree is a $(k - 1)$-subdivided star. Consider the case of a k-subdivided star. We can show that after `filter_1` and one iteration of a modified `filter_2` *all* leaves are assigned pre-images. Removing the leaves from T and the pre-images from support of $\mathcal{O}$, results in an ICPPL to a $(k - 1)$-subdivided star. Now we apply the induction hypothesis, and we get a isomorphism between the hypergraph $\mathcal{O}$ and $\mathcal{O}^\ell$.
In the reverse direction if there is a feasible path labeling ℓ, then we know that ℓ is unique up to isomorphism. Therefore, again by induction on k it follows that the algorithm finds ℓ. □

References

1. Atkins, J.E., Boman, E.G., Hendrickson, B.: A spectral algorithm for seriation and the consecutive ones problem. SICOMP: SIAM Journal on Computing 28 (1998)
2. Blair, J.R.S., Peyton, B.: An introduction to chordal graphs and clique trees. Tech. rep., Oak Ridge National Laboratory, Oak Ridge, Tennessee 37831 (1992)
3. Booth, K.S., Lueker, G.S.: 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 (Dec 1976)
4. Brandstädt, A., Le, V.B., Spinrad, J.P.: Graph classes: a survey. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (1999)
5. Gavril, F.: A recognition algorithm for the intersection graphs of paths in trees. Discrete Mathematics 23(3), 211 – 227 (1978)
6. Golumbic, M.C.: Algorithmic graph theory and perfect graphs, Annals of Discrete Mathematics, vol. 57. Elsevier Science B.V. (2004), second Edition
7. Hochbaum, Tucker: Minimax problems with bitonic matrices. NETWORKS: Networks: An International Journal 40 (2002)
8. Hochbaum, D.S., Levin, A.: Cyclical scheduling and multi-shift scheduling: Complexity and approximation algorithms. Discrete Optimization 3(4), 327–340 (2006)
9. Hsu, W.L.: PC-trees vs. PQ-trees. Lecture Notes in Computer Science 2108, 207–217 (2001)
10. Hsu, W.L.: A simple test for the consecutive ones property. J. Algorithms 43(1), 1–16 (2002)
11. Köbler, J., Kuhnert, S., Laubner, B., Verbitsky, O.: Interval graphs: Canonical representation in logspace. Electronic Colloquium on Computational Complexity (ECCC) 17, 43 (2010)
12. Kou, L.T.: Polynomial complete consecutive information retrieval problems. SIAM Journal on Computing 6(1), 67–75 (Mar 1977)
13. Kumar, P.S., Madhavan, C.E.V.: Clique tree generalization and new subclasses of chordal graphs. Discrete Applied Mathematics 117, 109–131 (2002)
14. McConnell, R.M.: 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)
15. Meidanis, J., Munuera, E.G.: A theory for the consecutive ones property. In: Proceedings of WSP'96 - Third South American Workshop on String Processing. pp. 194–202 (1996)
16. Narayanaswamy, N.S., Subashini, R.: A new characterization of matrices with the consecutive ones property. Discrete Applied Mathematics 157(18), 3721–3727 (2009)
17. Peyton, B.W., Pothen, A., Yuan, X.: A clique tree algorithm for partitioning a chordal graph into transitive subgraphs. Tech. rep., Old Dominion University, Norfolk, VA, USA (1994)
18. Renz, P.L.: Intersection representations of graphs by arcs. Pacific J. Math. 34(2), 501–510 (1970)
19. Schaffer, A.A.: A faster algorithm to recognize undirected path graphs. Discrete Applied Mathematics 43, 261–295 (1993)