SYNOPSIS OF

Generalization of Consecutive Ones Property of Binary Matrices

A THESIS

submitted by

ANJU SRINIVASAN

for the award of the degree

of

MASTER OF SCIENCE

(by Research)

DEPARTMENT OF COMPUTER SCIENCE
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

OCTOBER 2011

This is to certify that the thesis titled **Generalization of Consecutive Ones Property of Binary Matrices**, submitted by **Anju Srinivasan**, to the Indian Institute of Technology, Madras, for the award of the degree of **Master of Science *by Research***, is a bona fide record of the research work done by her under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. N. S. Narayanaswamy
Research Guide
Associate Professor
Dept. of Computer Science
IIT Madras – 600 036

Chennai
25 October 2011

[a:] [work in progress.]

Abstract

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

[a:] 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 $\phi : U \to V(T)$ 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 conclude with a polynomial time algorithm to find a feasible tree path labeling of a given set system when T is a |k-subdivided star|, set system has a single containment tree of overlap components and set size is limited to at most $k + 2$. [this is the abstract of the paper. expand to fit thesis.]

Contents

List of Tables

List of Figures

COP	Consecutive Ones Property
COT	Consecutive Ones Testing
ICPIA	Intersection Cardinality Preservation Interval Assignment
ICPPL	Intersection Cardinality Preserved Path Labeling

2^U	Powerset of set U

Introduction [a:] [Some things to remember - have this somewhere.

- Examples: Illustration of the problem examples. both COP and extension
- Basic preliminaries: (definitions theorems etc) needed if any - graph theory
- Organization: Outline of document

Abstract must be a brief about what results we have and how it fits in the body of research.

1. *Area:* - Broad to Specialized. i.e. Combinatorial algorithms -> Matrix reorganization -> general data reorganization (interval assignment) -> path assignment
2. *Class of problems:* say, data reorganization.
3. *Nature of results:* Is a generalization. We have a Polynomial algorithm for a subset of the generalization.

Introduction is a magnified version of Abstract with some brief citations etc. mini survey. Here talk about canonization how it connects these problems to graph isomorphism. Introduce our problem, explain why it is important etc. How canonization is a posisbility. Brief about results obtained.

Someone who doesn't want to go through the whole thesis must get the gist of the whole document from the introduction chapter. that should be the outlook.

once this is done, the rest of the chapters will fall into place.

Survey chapter is the full fledged expansion of the survey in introduction with details, observations, theorems etc.

The main themes of the thesis that must not be left out:

1. connection of COP to **set systems** and thus to general data bases. Narayanaswamy and Subashini (2009)
2. **graph isomorphism** by canonization Köbler *et al.* (2010)
3. **certificate for a problem** McConnell (2004)

Also see purple book notes.]

1 to area

2 to problem

3 to your thesis

4 Organization of document

Consecutive Ones property [a: [survey of cop includes mainly

1. Dom (2008) (perhaps minus chap 4?)
2. Narayanaswamy and Subashini (2009)
3. McConnell (2004)
4. Köbler *et al.* (2010)

.. to name a few.

sections will be as follows:]

5 Characterization of COP

[a: [Tucker (1972), McConnell (2004), prime submatrix Hsu (2002)?]

6 Recognition of COP

6.1 [a: **[Polynomial time algorithms]**

[a: [(pq tree, pqr, icpia)]

6.2 [a:] [Tucker's submatrices, Dom's algo to find them]

6.3 [a:] [McConnell's certificate]

7 Alteration to matrices to get COP

[a:] [algorithms to make a matrix into one with COP (min/max-cos-r/c) Dom (2008)]

8 brief analysis of hard problems with input having COP (chap 5, 6 in dom)

[a:] [Do I need this?]

9 Complexity of certain COP variations

[a:] [maybe combine with Section 8

complexity - poly, logspace]

Other problems related to COP [a:] [(see literature)]

Research - Tree path labeling of path hypergraphs [a:] [Extension to trees. (stuff in latin paper,)]

[a:] [UPDATE TO FINAL VERSION. BELOW IS OLDER VERSION WITH INCOMPLETE SEC 4.]

10 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 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) 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 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 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 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 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 runs 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 Narayanaswamy and Subashini (2009).

[a:] [the following text could be in Further Study/Conclusion]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 solvableGavril (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 focus on a new special case of the tree path labeling problem. Here the set system is such that every set is at most $k + 2$ in size and for every pair of sets in it there exists a sequence of sets between them with consecutive sets in this sequence having a strict intersection – i.e., non-empty intersection with neither being contained in the other. Moreover, the given tree is a *k-subdivided star*. We demonstrate a polynomial time algorithm to find a feasible path labeling in this case.

Roadmap. The necessary preliminaries with definitions etc. are presented in Section 11. Section 12 documents the characterization of a feasible path labeling and finally, Section 13 describes a polynomial time algorithm to find the tree path labeling of a given set system from a given k-subdivided tree.

11 Preliminaries

This section states definitions and basic facts necessary in the scope of this document.

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; $supp(\mathcal{F}) = U$.

The graph T represents a *target tree* with same number of vertices as elements in U; $|V(T)| = n$. A *path system* $\mathcal{P}$ is a set system of paths from T; $\mathcal{P} \subseteq \{P \mid P \subseteq V,\ T[P] \text{ is a path}\}$.

A set system $\mathcal{F}$ can be alternatively represented by a *hypergraph* $\mathcal{F}_H$ 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. Due to the equivalence of set system and hypergraph

in the scope of this paper, we drop the subscript $_H$ in the notation and refer to both the structures by $\mathcal{F}$.

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 $B = \{\phi(x) \mid x \in A\}$ Köbler *et al.* (2010). This is called *hypergraph isomorphism*. [a:] [also extend ϕ to hyperedges – see if required]

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 Golumbic (2004).

If the intersection graphs of two hypergraphs are isomorphic, $\mathbb{I}(\mathcal{F}) \cong \mathbb{I}(\mathcal{P})$ where $\mathcal{P}$ is also a path system, then the bijection $\ell : \mathcal{F} \rightarrow \mathcal{P}$ due to this isomorphism is called a *path labeling* of the hypergraph $\mathcal{F}$. To illustrate further, let $g : V(\mathcal{F}) \rightarrow V(\mathcal{P})$ be the above mentioned isomorphism where $V(\mathcal{F})$ and $V(\mathcal{P})$ are the vertex sets that represent the hyperedges for each hypergraph respectively, $V(\mathcal{F}) = \{v_S \mid S \in \mathcal{F}\}$ and $V(\mathcal{P}) = \{v_P \mid P \in \mathcal{P}\}$. Then the path labeling ℓ is defined as follows: $\ell(S_1) = P_1$ iff $g(v_{S_1}) = v_{P_1}$. The path system $\mathcal{P}$ may be alternatively denoted in terms of $\mathcal{F}$ and ℓ as $\mathcal{F}^{\ell}$. In most scenarios in this paper, what is given are the pair $(\mathcal{F}, \ell)$ and the target tree T; hence this notation will be used more often.

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 an *induced 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 $S \in \mathcal{F}$. Recall that $supp(\mathcal{P}) = V(T)$.

A path labeling $(\mathcal{F}, \ell)$ is defined to be *feasible* if $\mathcal{F} \cong \mathcal{F}^{\ell}$ and this hypergraph isomorphism $\phi : supp(\mathcal{F}) \rightarrow supp(\mathcal{F}^{\ell})$ induces a path labeling $\ell_\phi : \mathcal{F} \rightarrow \mathcal{F}^{\ell}$ such that $\ell_\phi = \ell$.

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 $S \cap$

$S' \neq \emptyset, S \not\subseteq S', S' \not\subseteq S$. 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 $S' \not\emptyset S$, the overlaps $S \cap S'$ form a single inclusion chain Köbler *et al.* (2010). Additionally, if S is such that it is contained in no other hyperedge in $\mathcal{F}$, i.e., it is inclusion maximal then it is called *super-marginal*.

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 $k + 2$.

12 Characterization of Feasible Tree Path Labeling

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

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

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`$(\mathcal{F}, \ell, T)$

$\mathcal{F}_0 \leftarrow \mathcal{F}$, $\ell_0(S) \leftarrow \ell(S)$ for all $S \in \mathcal{F}_0$
$j \leftarrow 1$
while there is $S_1, S_2 \in \mathcal{F}_{j-1}$ such that $\ell_{j-1}(S_1)$ and $\ell_{j-1}(S_2)$ have a common leaf in T **do**
  $\mathcal{F}_j \leftarrow (\mathcal{F}_{j-1} \setminus \{S_1, S_2\}) \cup \{S_1 \cap S_2, S_1 \setminus S_2, S_2 \setminus S_1\}$ | `Remove` S_1, S_2 `and add the "filtered" sets`
  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** | `Carry forward the path labeling for all existing sets other than` S_1, S_2
  $\ell_j(S_1 \cap S_2) \leftarrow \ell_{j-1}(S_1) \cap \ell_{j-1}(S_2)$
  $\ell_j(S_1 \setminus S_2) \leftarrow \ell_{j-1}(S_1) \setminus \ell_{j-1}(S_2)$ | `Define path labeling for new sets`
  $\ell_j(S_2 \setminus S_1) \leftarrow \ell_{j-1}(S_2) \setminus \ell_{j-1}(S_1)$
  if $(\mathcal{F}_j, \ell_j)$ does not satisfy (Property iii) of ICPPL **then**
    exit
  end if
  $j \leftarrow j + 1$
end while
$\mathcal{F}' \leftarrow \mathcal{F}_j$, $\ell' \leftarrow \ell_j$
return $(\mathcal{F}', \ell')$

Lemma 12.2 *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.*

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 $\phi : supp\left(\mathcal{F}_{j-1}\right) \to V(T)$ 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 proven. ∎

Lemma 12.3 *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 proven.
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 $R = S_1 \cap S_2$, $|R| = |S_1 \cap S_2| = |\ell_{j-1}(S_1) \cap \ell_{j-1}(S_2)| = |\ell_j(S_1 \cap S_2)| = |\ell_j(R)|$. If $R = S_1 \setminus S_2$, $|R| = |S_1 \setminus S_2| = |S_1| - |S_1 \cap S_2| = |\ell_{j-1}(S_1)| - |\ell_{j-1}(S_1) \cap \ell_{j-1}(S_2)| = |\ell_{j-1}(S_1) \setminus \ell_{j-1}(S_2)| = |\ell_j(S_1 \setminus S_2)| = |\ell_j(R)|$. 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 12.1 and definition of ℓ_j, it follows that invariant III is true no matter which of the new sets R is equal to. If $R = S_1 \cap S_2$, $|R \cap R'| = |S_1 \cap S_2 \cap R'| =$

$|\ell_{j-1}(S_1) \cap \ell_{j-1}(S_2) \cap \ell_{j-1}(R')| = |\ell_j(S_1 \cap S_2) \cap \ell_j(R')| = |\ell_j(R) \cap \ell_j(R')|$. If $R = S_1 \setminus S_2$, $|R \cap R'| = |(S_1 \setminus S_2) \cap R'| = |(\ell_{j-1}(S_1) \setminus \ell_{j-1}(S_2)) \cap \ell_{j-1}(R')| = |\ell_j(S_1 \cap S_2) \cap \ell_j(R')| = |\ell_j(R) \cap \ell_j(R')|$. 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 proven.

※ *Invariant* IV

Due to the condition in line 9, this invariant is ensured at the end of every iteration.

∎

Lemma 12.4 *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 12.2 $(\mathcal{F}_j, \ell_j)$ for every iteration $j > 0$ is feasible. Also, every hypergraph isomorphism $\phi : supp(\mathcal{F}) \to V(T)$ 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 12.3 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.

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: $\quad \mathcal{F}_j \leftarrow \mathcal{F}_{j-1} \setminus \{S_1\}$
5: $\quad$ for all $S \in \mathcal{F}_{j-1}$ such that $S \neq S_1$ set $\ell_j(S) \leftarrow \ell_{j-1}(S)$
6: $\quad X \leftarrow S_1 \setminus \bigcup_{S \in \mathcal{F}_{j-1}, S \neq S_1} S$
7: $\quad$ **if** X is empty **then**
8: $\quad\quad$ **exit**
9: $\quad$ **end if**
10: $\quad x \leftarrow$ arbitrary element from X
11: $\quad \mathcal{F}_j \leftarrow \mathcal{F}_j \cup \{\{x\}, S_1 \setminus \{x\}\}$
12: $\quad \ell_j(\{x\}) \leftarrow \{v\}$
13: $\quad \ell_j(S_1 \setminus \{x\}) \leftarrow \ell_{j-1}(S_1) \setminus \{v\}$
14: $\quad j \leftarrow j + 1$
15: **end while**
16: $\mathcal{F}' \leftarrow \mathcal{F}_j$, $\ell' \leftarrow \ell_j$
17: **return** $(\mathcal{F}', \ell')$

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 12.5 *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 12.6 *In Algorithm 2, for all $j > 0$, at the end of the jth iteration of the* **while** *loop the four invariants given in Lemma 12.3 hold.*

Proof By Lemma 12.5 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 12.5, 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. ∎

We have seen two filtering algorithms above, namely, Algorithm 1 `filter_1` and Algorithm 2 `filter_2` which when executed serially repectively 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.

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

if T is empty **then**
  return $\emptyset$
end if
$L \leftarrow \{v \mid v \text{ is a leaf in } T\}$
$(\mathcal{F}_1, \ell_1) \leftarrow$ `filter_1`$(\mathcal{F}, \ell, T)$
$(\mathcal{F}_2, \ell_2) \leftarrow$ `filter_2`$(\mathcal{F}_1, \ell_1, T)$
$(\mathcal{F}', \ell') \leftarrow (\mathcal{F}_2, \ell_2)$
$\phi' \leftarrow \emptyset$
for every $v \in L$ **do**
  $\phi'(x) \leftarrow v$ where $x \in \ell_2^{-1}(\{v\})$ | `Copy the leaf labels to a one to one function` $\phi' : supp(\mathcal{F}) \to L$
  Remove $\{x\}$ and $\{v\}$ from $\mathcal{F}'$, ℓ' appropriately
end for
$T' \leftarrow T \setminus L$
$\phi'' \leftarrow$ `get-hypergraph-isomorphism`$(\mathcal{F}', \ell', T')$
$\phi \leftarrow \phi'' \cup \phi'$
return ϕ

Lemma 12.7 *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 $\phi : supp(\mathcal{F}) \to V(T)$ 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 12.4 and Lemma 12.5 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 $\phi : supp(\mathcal{F}) \to V(T)$ 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 12.8 *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 12.7, 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 12.4 and Lemma 12.5 show us that filter 1 and filter 2 do not exit if input is feasible. Thus Algorithm 3 returns a non-empty function. ∎

12.1 ICPPL when given tree is a path

Consider a special case of ICPPL with the following properties.

1. Given tree T is a path. Hence, all path labels are interval labels.
2. Only pairwise intersection cardinality preservation is sufficient. i.e. property (iii) in ICPPL is not enforced.
3. The filter algorithms do not have **exit** statements.

This is called an Intersection Cardinality Preservation Interval Assignment (ICPIA) Narayanaswamy and Subashini (2009). 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.

13 Testing for feasible path assignments to k-subdivided star

In this section we consider the problem of assigning paths from a k-subdivided star T to a given set system $\mathcal{F}$. We consider $\mathcal{F}$ for which the overlap graph $\mathbb{O}(\mathcal{F})$ is connected. The overlap graph is well-known from the work of Köbler *et al.* (2010); Narayanaswamy and Subashini (2009); Hsu (2002). We use the notation in Köbler *et al.* (2010). Recall from Section 11 that hyperedges S and S' are said to overlap, denoted by $S \between S'$, if S and S' have a non-empty intersection but neither of them is contained in the other. The overlap graph $\mathbb{O}(\mathcal{F})$ is a graph in which the vertices correspond to the sets in $\mathcal{F}$, and the vertices corresponding to the hyperedges S and S' are adjacent if and only if they overlap. Note that the intersection graph of $\mathcal{F}$, $\mathbb{I}(\mathcal{F})$ is different from $\mathbb{O}(\mathcal{F})$ and $\mathbb{O}(\mathcal{F}) \subseteq \mathbb{I}(\mathcal{F})$. 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. We also assume that all hyperedges are of cardinality at most $k + 2$.

Recall from Section 11 that a k-subdivided star is a star with each edge subdivided k times. Therefore, 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.

In this section the given hypergraph, the k-subdivided star and the root of the star are denoted by $\mathcal{O}$, T and vertex r, respectively.

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 R_i 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.

Suppose the given hypergraph has a feasible path labeling to the given tree T. Let this labeling be ℓ. Now we present an algorithm that lists the sequence of edges that maintain a set of properties necessary for them to have a feasible tree path labeling.

I. **Step 1**
   (a) **There are type 1 edges, $|\mathcal{T}_1^i| > 0$:** If there is only one $\mathcal{T}_1^i$ hyperedge, label it to the unique path in R_i of length $s(X)$ starting at $v_{i(k+2)} = r$. If there are more than one $\mathcal{T}_1^i$ hyperedges, for any pair of hyperedges $S_1, S_2 \in \mathcal{T}_1^i$, one of the following is true. Let $\ell'(X) \subseteq R_j$ denote the subpath of length $p_1(X)$ that has been assigned to X from a previously considered ray R_j.

Case 1. $|\ell'(S_1) \cap \ell'(S_2)| = 1$ and $|S_1 \cap S_2| > 1$ i.e. they are in different rays and their residues must contain their intersection. Hence assign both to R_i.

Case 2. $|\ell'(S_1) \cap \ell'(S_2)| > 1$ and $|S_1 \cap S_2| > |\ell'(S_1) \cap \ell'(S_2)|$ i.e. they are in the same rays and their residues must contain their intersection. Hence assign both to R_i.

Case 3. $|\ell'(S_1) \cap \ell'(S_2)| > 1$ and $|S_1 \cap S_2| = |\ell'(S_1) \cap \ell'(S_2)|$. i.e. the intersection cardinality has been met. Hence assign only S_1 to R_i and S_2 must be assigned to some $i \geq p$.

Remove the labeled edge or edges from O_i and add it to O_{i+1}. Also do not add it to $\mathcal{T}_1^{i+1}$ and add to $\mathcal{L}_2^{i+1}$.

(b) **There are no type-1 edges nor super-marginal type-2 hyperedges:** If all type-2 edges of length at most $k+1$, EXIT by saying there is no ICPPL. Otherwise, Find an X of type-2 such that $|X| \geq k+2$, assign it the unique path starting at $v_{i(k+1)} = r$ of length $k+2$, and set $p_1(X) = k+2$, $s(X) = |X|-(k+2)$, assign it the unique path starting at $v_{i(k+1)} = r$ of length $k+2$, mark this as a type-1 hyperedge, and add it to O_{i+1} after removing it from O_i.

(c) **There are super-marginal type-2 hyperedges and no type-1 edges:** Let X be a super-marginal hyperedge. If $|X| \leq k+2$, then X is assigned the path of length $|X|$ starting at v_{i1}, and $p_1(X) = |X|$, $p_2(X) = 0$. X is removed from O_i and not added to O_{i+1}. If $|X| > k+2$, then $p_1(X) = k+2$, assign it the unique path starting at $v_{i(k+1)} = r$ of length $k+2$ and $s(X) = |X|-(k+2)$. In this case, X is classified as a Type-1 edge, removed from O_i, and added to O_{i+1}.

II. **Step 2:** Iteratively, a hyperedge X is selected from O_i that has an overlap with one of the hyperedges Y such that $\ell(Y) \subseteq R_i$, and a unique path is assigned to X. The path, say $U(X) \subseteq R_i$, that is assigned can be decided unambiguously since the $X \not\parallel Y$, and all intersection cardinalities can be preserved only at one of the ends of $\ell(Y)$.

Let $\ell(X)$ denote the unique path assigned to X. If X is a type-2 hyperedge and if the unique path of length $|X|$ does not contain r, then $p_1(X) = |X|$, $p_2(X) = 0$, and X is removed from O_i. In the case when $\ell(X)$ has to contain r, then $p_1(X)$ is the length of the path, $p_2(X) = 0$, and $s(X) = |X| - p_1(X)$. Further X is classified as a type-1 hyperedge, added to O_{i+1} and removed from O_i. In the case when X is a type-1 hyperedge, then we check if $U(X)$, which is of length $s(X)$ contains r. If it does, then we assign $\ell(X) \leftarrow \ell(X) \cup U(X)$, remove X from O_i and do not add it to O_{i+1}. If not, then we **Exit** reporting that an assignment cannot be found. The iteration ends when no hyperedge in O_i has an overlap with a hyperedge assigned to R_i.

The order in which they are assigned is the sequence of hyperedges with the following properties. Note that if such a sequence cannot be computed, O cannot have a feasible path labeling from T.

[a: [need to formalize the below properties]] In the following lemmas we identify a set of necessary conditions for $\mathcal{F}$ to have an ICPPL in the k-subdivided star T. If during the execution of the above algorithm, one of these necessary conditions is not satisfied, the algorithm exits reporting the non-existence of an ICPPL.

Lemma 13.1 *Let all hyperedges in O_i be type-1 edges. Then there is a maximal subset $\mathcal{T}_i \subseteq O_i$ with the following properties:*

1. *$\mathcal{T}_i$ form an inclusion chain.*
2. *For all $X \in \mathcal{T}_i$, $s(X) \leq k+2$, and There is a an $X \in \mathcal{T}_i$ such that $s(X) = k+2$.*

Lemma 13.2 *If there are no super-marginal type-2 edges in O_i, then there exists at least $p - i$ type-2 hyperedges $X \in O_i$ such that $|X| \geq k+2$.*

Lemma 13.3 *At the end of Step-1 in the i-th iteration, if one hyperedge X of type-2 is such that $\ell(X) \subseteq R_i$, then all other hyperedges in O_i are connected to X in the overlap component.*

Lemma 13.4 *At the end of Step-2, If control has exit at any time, there is no ICPPL. If control has not exit, then R_i is saturated. No hyperedge of O_{i+1} will get a path from R_i in the future iterations. No type-2 hyperedge of O_{i+1} will get a path from R_i. Basically R_i is done.*

Lemma 13.5 *Finally, we need to prove that the assignment is an ICPPL. Secondly, if there is a permutation then maps sets to paths, then it is indeed an ICPPL, and our algorithm will basically find it. It is a unique assignment upto permutation of the leaves.*

Lemma 13.6 *If $X \in \mathcal{F}$ is super-marginal and $(\mathcal{F}, \ell)$ is a feasible tree path labeling to tree T, then $\ell(X)$ will contain a leaf in T.*
On the otherhand, if $\ell(X)$ has a leaf in T, X is marginal but may not be super-marginal.

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 $X \nsubseteq Y_1$ 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 proven. ∎

Consider the overlap graph $\mathbb{O}(\mathcal{F})$ of the given hypergraph $\mathcal{F}$. Let $S_{sm} \in \mathcal{F}$ be such that it is a super-marginal hyperedge. Algorithm 4 uses S_{sm} along with the overlap graph $\mathbb{O}(\mathcal{F})$ to calculate the feasible tree path labeling to the k-subdivded tree T.

Algorithm 4 `compute-ksubtree-path-labeling`$(X, \mathcal{F}, T)$

1: **if** $X = S_{sm}$ **then**
2: – TBD –
3: **else**
4: – TBD –
5: **end if**

Research - [title TBD] [a: [the matrix theory submitted in walcom, any other work - need to see notes.]]

Conclusion

Appendix A

REFERENCES

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

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

3. **Booth, K. S.** and **G. S. Lueker** (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.

4. **Brandstädt, A.**, **V. B. Le**, and **J. P. Spinrad**, *Graph classes: a survey*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1999. ISBN 0-89871-432-X.

5. **Dom, M.** (2008). *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property*. Ph.D. thesis, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany. Published by Cuvillier, 2009.

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

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

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

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

10. **Hsu, W.-L.** (2001). PC-trees vs. PQ-trees. *Lecture Notes in Computer Science*, **2108**, 207–217. ISSN 0302-9743.

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

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

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

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

15. **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.

16. **Meidanis, J.** and **E. G. Munuera**, A theory for the consecutive ones property. *In Proceedings of WSP'96 - Third South American Workshop on String Processing*. 1996.

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

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

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

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

21. **Tucker, A.** (1972). A structure theorem for the consecutive 1's property. *J. Comb. Theory Series B*, **12**, 153–162.

1. Authors.... Title... *Journal*, Volume, Page, (year).