Generalization of the Consecutive-ones Property

A THESIS

submitted by

ANJU SRINIVASAN

for the award of the degree of

MASTER OF SCIENCE *by Research*

from the department of

COMPUTER SCIENCE AND ENGINEERING

at

INDIAN INSTITUTE OF TECHNOLOGY MADRAS

Guindy, Chennai - 600036

JANUARY 2012

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

Consecutive-ones property is a non-trivial property of binary matrices that has been studied widely in the literature for over past 50 years. Detection of COP in a matrix is possible efficiently and there are several algorithms that achieve the same. This thesis documents the work done on an extension of COP extended from the equivalent interval assignment problem in [NS09]. These new results rigorously prove a natural extension (to trees) of their characterization as well as makes connections to graph isomorphism, namely path graph isomorphism.

1.1 Organization of the document

Chapter 1 introduces the area of research and the problems addressed in this thesis. Chapter 2 gives a more detailed survey briefed in Section 1.4. Chapter 3 details all the results obtained to the problems of this thesis and finally the conclusion of the thesis is discussed in Chapter 4.

In this chapter, Section 1.2 introduces the main problem of this thesis by way of an illustration. Section 1.3 lays out a few general definitions that are helpful in understanding the rest of the chapter. Section 1.4 gives a brief survey of COP and optimization problems related to it followed by motivation for the thesis in Section 1.5. Section 1.6 presents a

$$
\begin{array}{rcl}
U & = & \{\textbf{Pa}, \textbf{Pi}, \textbf{Sn}, \textbf{Wo}, \textbf{Vi}, \textbf{Li}, \textbf{Ch}, \textbf{Sa}, \textbf{Fr}, \textbf{Sc}, \textbf{Lu}\} \\
\mathcal{F} & = & \{\mathbb{B}, \mathbb{T}, \mathbb{W}, \mathbb{F}\} \\
\mathbb{B} & = & \{\textbf{Ch}, \textbf{Sa}, \textbf{Fr}, \textbf{Sc}, \textbf{Lu}\} \\
\mathbb{T} & = & \{\textbf{Pa}, \textbf{Pi}, \textbf{Vi}, \textbf{Ch}\} \\
\mathbb{W} & = & \{\textbf{Sn}, \textbf{Pi}, \textbf{Wo}\} \\
\mathbb{F} & = & \{\textbf{Vi}, \textbf{Li}, \textbf{Ch}, \textbf{Fr}\} \\
n & = & |U| = 11 \\
m & = & |\mathcal{F}| = 4
\end{array}
$$

Table 1.1: Students and study groups in *Wallace Studies Institute*

summary of our results on the extension of COP namely, the tree path labeling problem.

1.2 Illustration of the problem

A group of students, **Pa***tricia*, **Pi***gpen*, **Sn***oopy*, **Wo***odstock*, **Vi***olet*, **Li***nus*, **Ch***arlie*, **Sa***lly*, **Fr***anklin*, **Sc***hröeder* and **Lu***cy* enroll at the *Wallace Studies Institute* for a liberal arts programme. As part of their semester thesis, they pick a body of work to study and form the namesake study groups, *"*$\mathbb{B}$*rief Interviews with Hideous Men"* [Wal99], *"The String* $\mathbb{T}$*heory"* [Wal96], *"[*$\mathbb{W}$*]Rhetoric and the Math Melodrama"* [Wal00] and *"*$\mathbb{F}$*ate, Time, and Language: An Essay on Free Will"* [Wal10]. A student will be in at least one study group and may be in more than one. For instance, as will be seen later, **Fr***anklin* studies both *"*$\mathbb{B}$*rief Interviews with Hideous Men"* and *"*$\mathbb{F}$*ate, Time, and Language: An Essay on Free Will"* while **Wo***odstock* studies only *"[*$\mathbb{W}$*]Rhetoric and the Math Melodrama"*.

Let U and $\mathcal{F}$ represent the set of students and the set of study groups respectively and the integers n and m denote the total number students and study groups respectively. In relation to this example, these are defined in Table 1.1. Also given there is the study group allocation to students.

The campus has a residential area *Infinite Loop* that has n single occupancy apartments reserved for the study groups' accommodation. All these apartments are located such that the streets connecting them do *not* form loops. Figure 1.1 shows the street map for *Infinite Loop*. It may be noted that as a graph, it classifies as a tree.

A natural question would be to find how the students should be allocated apartments such that each study group has the least distance to travel for a discussion? More specifically, we are interested in the problem with additional conditions, namely, that all the students in a study group must be next to each other; in other words, for one student to reach another fellow study group member's apartment (for all study groups the student

is part of), she must not have to pass the apartment of any student who is not in that study group. To further elucidate, the apartments of students of any study group must be arranged in an exclusive unfragmented path on the street map. Exclusivity here means that the path must not have apartments from other study groups (unless that apartment is also part of *this* study group).

An intuitive approach to this problem would be to first find the paths that each study group decides to inhabit and then refine the allocation to individual students. A feasible allocation of exclusive routes to study groups is illustrated in Figure 1.1. The students' allocation of apartments that obeys this route allocation is shown in Figure 1.3. Table 1.2 shows the same solution set theoretically. How this is algorithmically computed is the focus of this thesis.

1.2.1 Special case

As a special case of the study group accommodation problem, suppose all the apartments are on the same street or if they are all lined up on a single path, the street map becomes a tree that is just a path. Then the problem becomes what is called an *interval assignment problem*. The idea of interval assignment may not be obvious here; hence to see this, consider a different problem in *Wallace Studies Institute* where the classes for these study groups courses need to be scheduled during a day (or a week or any time period). Each study group has a bunch of courses associated with it some of which may be shared by two or more study groups. It is mandatory that a student who is a member of a study group takes all the courses associated with that group. There are slots during the day for classes to be held and the problem is to allocate class slots to courses such that all the classes of a study group are consecutive. The parallels between this class allocation problem and the accommodation problem can be seen as follows. The set U

			Apartment allocation (ϕ)	
T	=	*Street map tree of* Infinite Loop	1	**Sa**
$V(T)$	=	$\{1,2,3,4,5,6,7,8,9,10,11\}$	2	**Pi**
$\mathcal{P}$	=	$\{R\mathbb{B}, R\mathbb{T}, R\mathbb{W}, R\mathbb{F}\}$	3	**Fr**
$R\mathbb{B}$	=	$\{9,1,5,3,11\}$	4	**Wo**
$R\mathbb{T}$	=	$\{7,2,6,5\}$	5	**Ch**
$R\mathbb{W}$	=	$\{8,2,4\}$	6	**Vi**
$R\mathbb{F}$	=	$\{10,6,5,3\}$	7	**Pa**
n	=	$\lvert V\rvert = 11$	8	**Sn**
m	=	$\lvert\mathcal{P}\rvert = 4$	9	**Lu**
			10	**Li**
ℓ	=	*Study group to route mapping*	11	**Sc**
$\ell(\mathbb{X})$	=	$R\mathbb{X}$ for all $\mathbb{X} \in \mathcal{F}$		

Table 1.2: A solution to study group accomodation problem

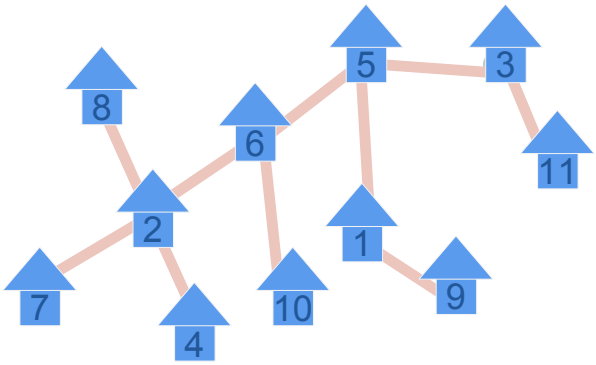

Figure 1.1: *Infinite Loop* street map.

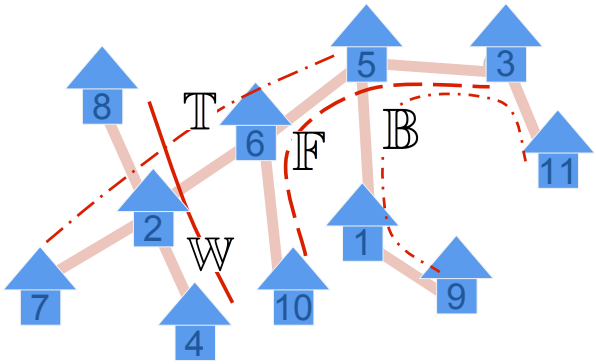

Figure 1.2: *Infinite Loop* street map with study group routes allocated.

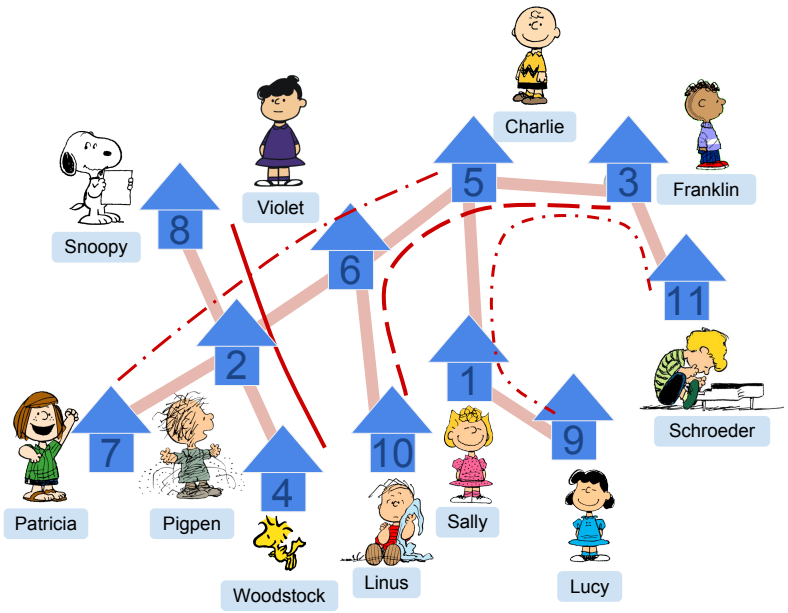

Figure 1.3: Individual allocation of apartments to students in *Infinite Loop* that meets the requirements stated before.

Peanuts images © Charles Schulz

here, are the courses offered (say Course 101 *"Influence of post modernism in Wallace's work"*, Course 102 *"A study on fragmented prose method"* and so on). In this variation of the problem, the collection $\mathcal{F}$ is the set of study groups but the study groups are filled by course IDs (in place of students in the earlier example). For instance, Course 101 is mandatory for all study groups $\mathbb{B}$, $\mathbb{T}$, $\mathbb{W}$, $\mathbb{F}$ and Course 102 is mandatory for only the $\mathbb{B}$ group) and so on. The sequence of class slots for the day (or week or any time period) is analogous to the street map in the accommodation problem. It is quite obvious now why this version of the problem (where the "target graph" is a path and not any tree) is called an interval assignment problem.

The interval assignment problem to a set system is equivalent to the consecutive-ones property (COP) problem in binary matrices[Hsu02, NS09]. The COP problem is to rearrange rows (columns) of a binary matrix in such a way that every column (row) has its **1**s occur consecutively. If this is possible the matrix is said to have the COP. COP is a well researched combinatorial problem and has several positive results on tests for it and computing the COP permutation (i.e. the course schedule in the above illustration) which will be surveyed later in this document. Hence we are interested in extensions of COP, more specifically, the extension of interval assignment problem to tree path assignment problem (which is illustrated by the study group accommodation problem).

1.3 Basic preliminaries

Here we see some basic definitions and conventions necessary for the contents of this thesis.

1.3.1 Matrices

If n is a positive integer, $[n]$ denotes the set $\{1, 2, ...n\}$.

Definition 1.3.1 (Binary matrix). Let M be an $n \times m$ matrix. $m_{i,j}$ denotes its (i, j)th element, i.e. element at ith row and jth column. M is a **binary matrix** if each of its element is 0 or **1**; for all $i \in [n]$ and $j \in [m]$, $m_{i,j} \in \{0, \mathbf{1}\}$

Definition 1.3.2 (Permutation). A **permutation** λ of a set $X = \{x_1, x_2, \ldots x_n\}$ is a bijection $\lambda : X \to X$. λ may be written as a sequence $x_{i_1} x_{i_2} \ldots x_{i_n}$, where $i_j \in [n]$ to mean that $\lambda(x_j) = x_{i_j}$. This document uses both notations as convenient in

context.

The idea of consecutive-ones property of binary matrices was mentioned a few times in earlier sections. Now we will see the formal definition of COP in Definition **??**.

Definition 1.3.3 (Consecutive-ones property (on columns)). Let M be a binary matrix.

1. A **block of 1s (block of 0s)** in a column of M is a maximal set of consecutive **1**-entries (0-entries) in this column.
2. M has the **strong consecutive-ones property (strong COP)** if in every column the **1**s appear consecutively, i. e. if every column contains at most one block of **1**s.
3. M has the **consecutive-ones property** if its rows can be permuted in such a way that the resulting matrix has the strong COP.
4. If an ordering for the rows of M gives the strong COP, it is called a **COP order** or **COP permutation**.

When the roles of rows and columns are exchanged in Definition 1.3.3, it defines COP on rows. Both are equivalent properties (for the purposes of this thesis) and both conventions are seen the literature. Figure 1.4 gives an example of COP on columns.

M_1:

r_1	**1**	0	**1**
r_2	0	**1**	0
r_3	**1**	0	0
r_4	0	**1**	**1**

M_1':

r_3	**1**	0	0
r_1	**1**	0	**1**
r_4	0	**1**	**1**
r_2	0	**1**	0

M_2:

d_1	**1**	0	0
d_2	**1**	**1**	**1**
d_3	0	**1**	0
d_4	0	0	**1**

Figure 1.4: Matrices with and without COP. M_1 has COP because by permuting its rows, r_1-r_4, one can obtain M_1' where the **1**s in each column are consecutive. M_2, however, does not have COP since no permutation of its rows, d_1-d_4, will arrange **1**s in each column consecutively [Dom08].

A less restrictive property of binary matrices that is a generalization of the COP is circular-ones property (CROP). The matrix can be visualized to be wrapped around a horizontal cylinder and demands that after some row permutations, if required, in every column the **1**s appear consecutively on the cylinder – note that this implies the 0s also appear consecutively.

Definition 1.3.4 (Circular-ones property (on columns)). Let M be a binary matrix.

1. M has the **strong circular-ones property (strong CROP)** if in every column the **1**s appear consecutively or the 0s appear consecutively or both.
2. M has the **circular-ones property (CROP)** if its rows can be permuted in such a way that the resulting matrix has the strong CROP.

3. If an ordering for the rows of M gives the strong CROP, then it is called the **CROP ordering** or **CROP permutation**.

1.3.2 Sets

Definition 1.3.5 (Set system). Let U be a universe with $|U| = n$.

1. The set $\mathcal{F} \subseteq (2^U \setminus \emptyset)$ is called a **set system** with universe U.
2. Two sets $S, S' \in \mathcal{F}$ are said to **overlap**, denoted by $S \between S'$, if they have a non-empty intersection and neither is contained in the other.

$$S \between S' \text{ if and only if } S \cap S' \neq \emptyset, S \nsubseteq S', S' \nsubseteq S$$

Definition 1.3.6. Let X be a partially ordered set with $\preccurlyeq$ being the partial order on X.

1. An element $X_m \in X$ is a **maximal upper bound** of X if $\nexists X_q \in X$ such that $X_m \preccurlyeq X_q$. A maximal upper bound on X is denoted by $mub(X)$.
2. A set of sets form a **single inclusion chain** when the Hasse diagram of their subset relation forms a single chain.

1.3.3 Graphs

Definition 1.3.7 (Graph, Tree, Path).

1. An **(undirected) graph** is $G = (V, E)$ is such that V is a finite set and E is a set of ordered pairs, $E \subseteq V \times V$. An element in V is called a **vertex (pl. vertices)** and an element from E is called an **edge**. V and E may be written as $V(G)$ and $E(G)$ respectively. If $u, v \in V$ and $(u, v) \in E$ then v is said to be **adjacent** to u and vice versa. A **subgraph** of G is a graph $G' = (V', E')$ such that $V' \subseteq V$ and $E' \subseteq E$.
2. A **path** is a graph $P = (V, E)$ with vertex set $V = \{v_1, \ldots, v_n\}$ and edge set $E = \{(v_1, v_2), (v_2, v_3), \ldots, (v_{n-2}, v_{n-1}), (v_{n-1}, v_n)\}$. The vertices v_1, v_n are called the **endpoints** of P. The graph P is called a **cycle** if it has an additional edge (v_n, v_1).
3. A **tree** is a graph T that has no subgraph that is a cycle.

Definition 1.3.8 (Intersection graph). Let $\mathcal{F}$ be a set system. Then its **intersection graph** $\mathbb{I}(\mathcal{F})$ is a graph such that its vertex set has a bijection to $\mathcal{F}$ and there exists an edge between two vertices if and only if their corresponding sets in $\mathcal{F}$ have a non-empty intersection.

Definition 1.3.9 (Interval graph, Path graph). Let G be a graph.

1. G is an **interval graph** if there a set of intervals $\mathcal{I}$ such that G is isomorphic to the intersection graph of $\mathcal{I}$.
$$G \cong \mathbb{I}(\mathcal{I})$$
2. G is a **path graph** if there exists a tree T and a set of paths from T, $\mathcal{P}$ such that G is isomorphic to the intersection graph of $\mathcal{P}$.
$$G \cong \mathbb{I}(\mathcal{P})$$
3. G is a **chordal graph** if it has no induced cycle of length greater than 3. Moreover, a **chordal graph** is characterized as an intersection graph of subtrees of a tree.

1.4 Consecutive-ones Property - a Brief Survey

In this section, a brief survey of the consecutive-ones problem and its optimization problems is presented.

1.4.1 Matrices with COP

As seen earlier, the interval assignment problem (illustrated as the course scheduling problem in Section 1.2), is a special case of the problem we address in this thesis, namely the tree path labeling problem (illustrated as the study group accomodation problem). The interval assignment problem and COP problem are equivalent problems. In this section we will see some of the results that exists in the literature today towards solving the COP problem and optimization problems surrounding it.

Recall that a matrix with COP is one whose rows (columns) can be rearranged so that the **1**s in every column (row) are in consecutive rows (columns). COP in binary matrices has several practical applications in diverse fields including scheduling [HL06], information retrieval [Kou77] and computational biology [ABH98]. Further, it is a tool in graph theory [Gol04] for interval graph recognition, characterization of Hamiltonian graphs, planarity testing [BL76] and in integer linear programming [HT02, HL06].

The obvious first questions after being introduced to the consecutive ones property of binary matrices are if COP can be detected efficiently in a binary matrix and if so, can the COP permutation of the matrix also be computed efficiently? Recognition of COP

in a binary matrix is polynomial time solvable and the first such algorithm was given by [FG65]. A landmark result came a few years later when [Tuc72] discovered the families of forbidden submatrices that prevent a matrix from having COP and most, if not all, results that came later were based on this discovery which connected COP in binary matrices to convex bipartite graphs. In fact, the forbidden submatrices came as a corollary to the discovery that convex bipartite graphs are AT-free on at least one of the partitions in [Tuc72]. The first linear time algorithm for COP testing (COT) was invented by [BL76] using a data structure called PQ-trees. Since then several COT algorithms have been invented – some of which involved variations of PQ-trees [MM96, Hsu01, McC04], some involved set theory and ICPIA [Hsu02, NS09], parallel COT algorithms[AS95, BS03, CY91] and certifying algorithms[McC04].

The construction of PQ-trees in [BL76] draws on the close relationship of matrices with COP to interval graphs. A PQ tree of a matrix is one that stores all row (column) permutations of the matrix that give the COP orders (there could be multiple orders of rows or columns) of the matrix. This is constructed using an elaborate linear time procedure and is also a test for planarity. PQR trees is a generalized data structure based on PQ trees [MM96, MPT98]. [TM05] describes an improved algorithm to build PQR trees. [Hsu02] describes the simpler algorithm for COT. Hsu also invented PC trees [Hsu01] which is claimed to be much easier to implement. [NS09] describes a characterization of consecutive-ones property solely based on the cardinality properties of the set representations of the columns (rows); every column (row) is equivalent to a set that has the row (column) indices of the rows (columns) that have one entries in this column (row). This is interesting and relevant, especially to this thesis because it simplifies COT to a great degree.

[McC04] describes a different approach to COT. While all previous COT algorithms gave the COP order if the matrix has the property but exited stating negative if otherwise, this algorithm gives an evidence by way of a certificate of matrix even when it has no COP. This enables a user to verify the algorithm's result even when the answer is negative. This is significant from an implementation perspective because automated program verification is hard and manual verification is more viable. Hence having a certificate reinforces an implementation's credibility. Note that when the matrix *has* COP, the COP order is the certificate. The internal machinery of this algorithm is related to the weighted betweenness problem addressed in [COR98].

1.4.2 Optimization problems in COP

So far we have been concerned about matrices that have the consecutive ones property. However in real life applications, it is rare that data sets represented by binary matrices have COP, primarily due to the noisy nature of data available. At the same time, COP is not arbitrary and is a desirable property in practical data representation [COR98, JKC+04, Kou77]. In this context, there are several interesting problems when a matrix does not have COP but is "close" to having COP or is allowed to be altered to have COP. These are the optimization problems related to a matrix which does not have COP. Some of the significant problems are surveyed in this section.

[Tuc72] showed that a matrix that does not have COP have certain substructures that prevent it from having COP. Tucker classified these forbidden substructures into five classes of submatrices. This result is presented in the context of convex bipartite graphs which [Tuc72] proved to be AT-free in one of the partitions. By definition, convex bipartite graph have half adjacency matrices that have COP on either rows or columns (graph is biconvex if it has COP on both)[Dom08]. A half adjacency matrix is a binary matrix representing a bipartite graph as follows. The set of rows and the set of columns form the two partitions of the graph. Each row node is adjacent to those nodes that represent the columns that have **1**s in the corresponding row. [Tuc72] proves that this bipartite graph has no asteroidal triple in vertex partition corresponding to rows if and only if the matrix has COP on columns and goes on to identify the forbidden substructures for these bipartite graphs. The matrices corresponding to these substructures are the forbidden submatrices.

Once a matrix has been detected to not have COP (using any of the COT algorithms mentioned earlier), it is naturally of interest to find out the smallest forbidden substructure (in terms of number of rows and/or columns and/or number of entries that are **1**s). [Dom08] discusses a couple of algorithms which are efficient if the number of **1**s in a row is small. This is of significance in the case of sparse matrices where this number is much lesser than the number of columns. $(*, \Delta)$-*matrices* are matrices with no restriction on number of **1**s in any column but have at most Δ **1**s in any row. Min COS-R (Min COS-C), Max COS-R (Max COS-C) are similar problems which deals with inducing COP on a matrix. In Min COS-R (Min COS-C) the question is to find the minimum number of rows (columns) that must be deleted to result in a matrix with COP. In the dual problem Max COS-R (Max COS-C) the search is for the maximum number of rows (columns) that induces a submatrix with COP. Given a matrix M with no COP, [Boo75] shows that finding a submatrix M' with all columns check if b75 deals with COP col

or COP row. also is it any submatrix with k less than r rows or submatrix must have all columns? but a maximum cardinality subset of rows such that M' has COP is NP complete. [HG02] corrects an error of the abridged proof of this reduction as given in [GJ79]. [Dom08] discusses all these problems in detail giving an extensive survey of the previously existing results which are almost exhaustively all approximation results and hardness results. Taking this further, [Dom08] presents new results in the area of parameterized algorithms for this problem.

Another problem is to find the minimum number of entries in the matrix that can be toggled to result in a matrix with COP. [Vel85] discusses approximation of COP AUGMENTATION which is the problem of changing of the minimum number of zero entries to **1**s so that the resulting matrix has COP. As mentioned earlier, this problem is known to be NP complete due to [Boo75]. [Vel85] also proves, using a reduction to the longest path problem, or is it a survey of another result? check. that finding a Tucker's forbidden submatrix of at least k rows is NP complete. how is this different from booth's 75 result??

[JKC+04] discusses the use of matrices with almost-COP (instead of one block of consecutive **1**s, they have x blocks, or *runs*, of consecutive **1**s and x is not too large) in the storage of very large databases. The problem is that of reordering of a binary matrix such that the resulting matrix has at most k runs of **1**s. This is proved to be NP hard using a reduction from the Hamiltonian path problem.

1.5 Generalization of COP - the Motivation

Section 1.4.1 introduced a succinct characterization for consecutive-ones property which is solely based on the cardinality properties of the set representations of the matrix's columns [NS09]. This result is very relevant to this thesis because aside from it simplifying COT to a great degree, our generalization problem is motivated by their results.

[NS09] characterizes interval assignments to the sets which can be obtained from a single permutation of the rows. For an assignment to be feasible, the cardinality of the interval assigned to each set in the system must be same as the cardinality of the set, and the intersection cardinality of any two intervals must be same as the intersection cardinality of their corresponding sets. While this is obviously a necessary condition, this result shows this is also sufficient. [NS09] calls this an Intersection Cardinality Preserving Interval Assignment (ICPIA). This paper generalizes the idea from [Hsu02]

of decomposing a given binary matrix into prime matrices for COT and describes an algorithm to test if an ICPIA exists for a given set system.

The equivalence of the problem of testing for the consecutive-ones property to the constraint statisfaction problem of interval assignment [NS09] or interval labeling [KKLV10] is as follows. Every column (row) of the binary matrix can be converted into a set of non-negative integers which are the indices of rows (columns) with **1**s in that column (row). It is apparent that if the matrix has COP in columns (rows), then constructing such sets after applying the COP permutation to the rows (columns) of the matrix will result in sets with consecutive integers. In other words, after application of COP reordering, the sets are intervals. Indeed the problem now becomes finding interval assignments to a given set system such that there exists a permutation of the universe of set of row indices (column indices) which converts each set to its assigned interval.

The problem of interest in this thesis, namely, tree path labeling problem, is a natural generalization of the interval assignment problem or the COT problem. The problem is defined as follows – given a set system $\mathcal{F}$ from a universe U and a target 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 Compute Feasible Tree Path Labeling problem or simply *tree path labeling* problem for an input set system and target tree pair – $(\mathcal{F}, T)$. The special case of the target tree being a path, is the interval assignment problem. We focus on generalizing the notion of an ICPIA [NS09] 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 [1] bijection between U and $V(T)$ if and only if the intersection cardinalities among any three sets (not necessarily distinct) is equal to that of the corresponding paths assigned to them and the input passes a filtering algorithm (described in this paper) successfully. This algorithmic characterization gives a natural data structure that stores all the feasible bijections between U and $V(T)$. This reduces the search space for the solution considerably from the universe of all possible bijections between U and $V(T)$ to only those bijections that maintain the characterization. Further, the filtering algorithm is also an efficient algorithm to test if a tree path labeling [2] is feasible.

1. The notion of *feasibility* is formally defined in Section 3.2.

2. The terms *tree path labeling* and *tree path assignment* are, in informal language, synonyms. Formally, the former refers to the bijection $\ell : \mathcal{F} \to \mathcal{P}$. The latter refers to the set of ordered pairs $\{(S, P) \mid S \in \mathcal{F}, P \in \mathcal{P}\}$. $\mathcal{P}$ is a set of paths on T.

1.6 Summary of New Results in this Thesis

We see in Section 1.5 that pairwise intersection cardinality preservation is necessary and sufficient for an interval assignment to be feasible for a given hypergraph [3] [4] and thus is a characterization for COP [NS09]. In our work we extend this characterization and find that trio-wise intersection cardinality preservation makes a tree path labeling [5] [4] (TPL) feasible, which is a generalization of the COP problem. This problem is defined by Feasible Tree Path Labeling (See Section 3.1 for problem definition).

We give a necessary and sufficient condition by way of *Intersection Cardinality Preservation Path Labeling* (ICPPL) and a filtering algorithm for Feasible Tree Path Labeling to output in affirmative. ICPPL captures the trio-wise cardinality property described earlier [6]. This characterization can be checked in polynomial time. A relevant consequence of this constructive procedure is that 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 of any three hyperedges is of the same cardinality as the intersection of the corresponding paths. Thus this generalizes the well studied question of the feasible interval assignment problem which is the special case when the target tree T is simply a path [Hsu02, NS09].

Aside from checking if a given TPL is feasible, we also solve the problem of computing a feasible TPL for a given hypergraph and target tree, if one exists. This problem is the Compute Feasible Tree Path Labeling problem (See Section 3.1 for problem definition).

We present a polynomial time algorithm for Compute Feasible Tree Path Labeling when the target tree T belongs to a special class of trees called *k-subdivided stars* and when the hyperedges in the hypergraph $\mathcal{F}$ have at most $k+2$ vertices (Compute k-subdivided Star Path Labeling– See Section 3.1 for problem definition). A couple of examples of k-subdivided stars are given in Figure 1.5.

In spite of this being a restricted case, we believe that our results are of significant interest in understanding the nature of Graph Isomorphism which is polynomial time solvable in interval graphs while being hard on path graphs[KKLV10]. k-subdivided stars

3. A *hypergraph* is an alternate representation of a set system and will be used in this thesis.
4. See Section 3.2 for the formal definition.
5. A *tree path labeling* ℓ is a bijection of paths from the target tree T to the hyperedges in given hypergraph $\mathcal{F}$.
6. See Section 3.3 for the definition of ICPPL.

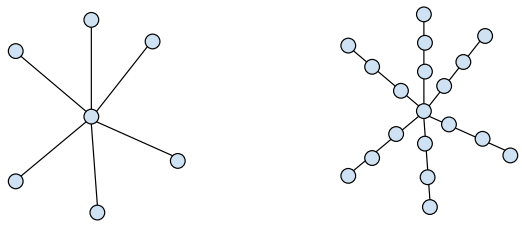

Figure 1.5: Examples of k-subdivided stars. (a) $k = 0$ (b) $k = 2$

are a class of trees which are in many ways very close to intervals or paths. Each ray [7] [4] are independent except for the root [8] [4] and hence can be considered as an independent interval till the root. Our algorithm builds on this fact and uses the interval assignment algorithm[NS09] up until "reaching" the root and then uses the trio-wise intersection cardinality (the extra condition in ICPPL that generalizes ICPIA) check to resolve the ambiguity about which ray the algorithm should "grow" the solution into in the next iteration.

We also have an algorithm for solving Compute Feasible Tree Path Labeling with no restrictions on the target tree or set size which runs in exponential time. This algorithm finds a path labeling from T by decomposing the problem into subproblems of finding path labeling of subsets of $\mathcal{F}$ from subtrees of T. Given the fact that binary matrices naturally represent a set system (see Section 1.5) and that the *overlap* relation between the sets involved is an obvious equivalence relation, $\mathcal{F}$ quite naturally partitions into equivalence classes known as *overlap components*. In the context of COP, overlap components were used in [Hsu02] and [KKLV10]. Moreover, [NS09] discovered that these equivalence classes form a total order. We extend this to TPL and find that when $\mathcal{F}$ is a path hypergraph [9], the classes can be partially ordered as an in-tree in polynomial time. Once $\mathcal{F}$ is "broken" into overlap components, one must identify the subtree of T that it needs to map to and this is the hard part which is currently open to be solved in polynomial time.

7. The path from a leaf to the root, the vertex with highest degree, is called a *ray* of the k-subdivided star.

8. The vertex with maximum degree in a k-subdivided star is called *root*.

9. If there exists an FTPL for a hypergraph $\mathcal{F}$, it is called a path hypergraph.