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 & ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

NOVEMBER 2011

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 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"*, *"The String* $\mathbb{T}$*heory"*, *"[*$\mathbb{W}$*]Rhetoric and the Math Melodrama"* and *"*$\mathbb{F}$*ate, Time, and Language: An Essay on Free Will"*. 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. Also given there is the study group allocation to students.

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

Table 1: Students and study groups in *Wallace Studies Institute*

The campus has a residential area *Infinite Loop* that has n single occupancy apart-

ments reserved for the study groups' accommodation. All these apartments are located such that the streets connecting them do *not* form loops. Fig 1 shows the street map for *Infinite Loop*. It may be noted that as a graph, it classifies as a tree.

			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**
ℓ	=	*Study group to route mapping*	10	**Li**
$\ell(\mathbb{X})$	=	$R\mathbb{X}$ for all $\mathbb{X} \in \mathcal{F}$	11	**Sc**

Table 2: A solution to study group accommodation problem

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 enforcing 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 Fig 1 and the students' allocation of apartments that obeys this route allocation is also shown. Table 2 shows the same solution set theoretically. How this is algorithmically computed is the focus of this thesis.

As a special case, 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

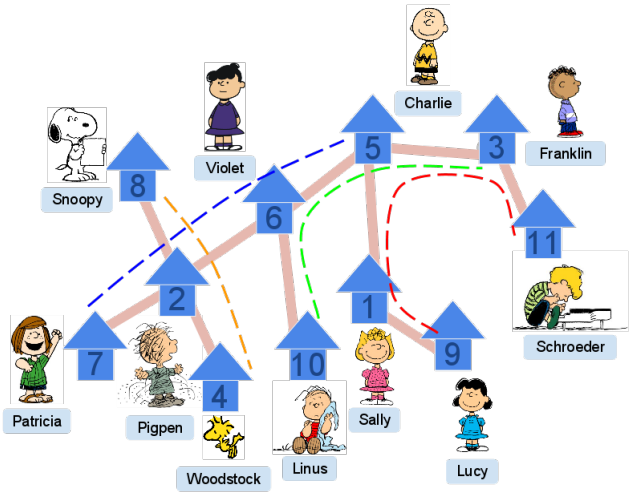


Figure 1: Individual allocation of apartments to students in *Infinite Loop* that meets the requirements stated before. The routes are color coded as follows: red for $\mathbb{B}$ group, blue for $\mathbb{T}$ group, orange for $\mathbb{W}$ group, green for $\mathbb{F}$ group. *Peanuts images © Charles Schulz*

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. It is debatable if this will not hamper the attention span and memory retention rate of the students but that is, regrettably, out of the scope of this thesis. The parallels between this class allocation problem and the accommodation problem can be seen as follows. The set U 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).

Section 2 gives a brief survey of COP and optimization problems related to it followed by motivation for the thesis in Section 3. Section 4 presents a summary of our results on the extention of COP namely, the tree path labeling problem.

2 Consecutive Ones Property Testing - a Survey

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

2.1 Matrices with COP

As seen earlier, the interval assignment problem (illustrated as the course scheduling problem in Section 1.1), is a special case of the problem we address in this thesis, namely the tree path labeling problem (illustrated as the study group accommodation 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). Figure 2 shows examples of this property. COP in binary matrices has several practical applications in diverse fields including scheduling [HL06], information retrieval [Kou77] and com-

M_1:

c_1	c_2	c_3	c_4
1	0	**1**	0
0	**1**	0	**1**
1	0	0	**1**

M_1':

c_3	c_1	c_4	c_2
1	**1**	0	0
0	0	**1**	**1**
0	**1**	**1**	0

M_2:

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

Figure 2: Matrices with and without COP. M_1 has COP because by permuting its columns, c_1-c_4, one can obtain M_1' where the **1**s in each row are consecutive. M_2, however, does not have COP since no permutation of its columns, d_1-d_4, will arrange **1**s in each row consecutively [Dom08].

putational 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 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].

2.2 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. 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 if and only if the matrix has COP 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 sparce 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 has 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 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, that finding a Tucker's forbidden submatrix of at least k rows is NP

complete.

[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 proven to be NP hard using a reduction from the hamiltonian path problem.

3 Generalization of COP - The Motivation

As seen in Section 2.1, [NS09] describes a characterization of consecutive ones property based on the cardinality properties of the set representations of the columns. 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.

The result in [NS09] 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 obviously a necessary condition, this result shows this is also sufficient. [NS09] calls such an interval assignment 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 tree path labeling problem is a natural generalization of the interval assignment problem or the COP problem. 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 [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 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 gives a natural data structure that stores all the relevant 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 to the set system is feasible.

4 Summary of results

As we saw earlier by the result in [NS09], pairwise intersection cardinality preservation is necessary and sufficient for an interval assignment to be feasible for a given hypergraph[1] and thus is a characterization for COP. In our work we extend this characterization and find that trio-wise intersection cardinality preservation makes a tree path labeling [2] (TPL) feasible, which is a generalization of the COP problem. This problem is defined as follows.

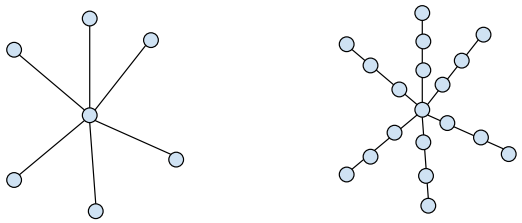
FEASBILE TREE PATH LABELING

`Input`	A hypergraph $\mathcal{F}$ with vertex set U, a tree T, a set of paths $\mathcal{P}$ from T and a bijection $\ell : \mathcal{F} \to \mathcal{P}$.
`Question`	Does there exist a bijection $\phi : U \to V(T)$ such that ϕ when applied on any hyperedge in $\mathcal{F}$ will give the path mapped to it by the given tree path labeling ℓ. i.e., $\ell(S) = \{\phi(x) \mid x \in S\}$, for every hyperedge $S \in \mathcal{F}$.

We give a necessary and sufficient condition (called ICPPL) for FEASBILE TREE PATH LABELING to output in affirmative. This characterization can be checked in polynomial time. The most interesting consequence is that in our constructive procedure, it is suffi-

[1] A *hypergraph* is an alternate representation of a set system and will be used through out this section.

[2] A *tree path labeling* ℓ is a bijection of paths from the target tree T to the hyperedges in given hypergraph $\mathcal{F}$.



(a) (b)

Figure 3: Examples of k-subdivided stars. (a) $k = 0$ (b) $k = 2$

cient 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, Compute Feasible Path Labeling, is defined as follows.

Compute Feasible Path Labeling

`Input`	A hypergraph $\mathcal{F}$ with vertex set U and a tree T.
`Question`	Does there exist a set of paths $\mathcal{P}$ from T and a bijection $\ell : \mathcal{F} \to \mathcal{P}$, such that Feasbile Tree Path Labeling returns **true** on $(\mathcal{F}, T, \ell)$.

We present a polynomial time algorithm for Compute Feasible 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. A couple of examples of k-subdivided stars are given in Figure 3.

COMPUTE k-SUBDIVIDED STAR PATH LABELING

`Input`	A hypergraph $\mathcal{F}$ with vertex set U such that every hyperedge $S \in \mathcal{F}$ is of cardinality atmost $k+2$ and a k-subdivided star T.
`Question`	Does there exist a set of paths $\mathcal{P}$ from T and a bijection $\ell : \mathcal{F} \to \mathcal{P}$, such that FEASBILE TREE PATH LABELING returns **true** on $(\mathcal{F}, T, \ell)$.

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 are a class of trees which are in many ways very close to intervals or paths. Each ray[3] are independent except for the root[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 after 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 PATH LABELING with no restrictions which runs in exponential time. This algorithm finds a path labeling from T by decomposing the problem into smaller 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 and that the overlap relation between the sets involved is an obvious equivalence relation, $\mathcal{F}$ naturally partitions into equivalence classes known as overlap components. In the context of COP, *overlap components* were used in [Hsu02] and [KKLV10]. Moreoever, [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[5], 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.

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

[4]The vertex with maximum degree in a k-subdivided star is called *root*

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

5 Conclusion

We give a characterization for feasible tree path labeling of path hypergraphs. The proof for this is constructive and computes a feasiblility bijection mapping vertices of the hypergraph to vertices of the given target tree. This thesis also discovered an exponential algorithm that computes a feasible TPL to a given hypergraph if it is a path hypergraph.

Our results have close connections to recognition of path graphs and to path graph isomorphism. Graphs which can be represented as the intersection graph of paths in a tree are called *path graphs*[Gol04]. Thus, a hypergraph $\mathcal{F}$ can be interpreted as paths in a tree, if and only if the intersection graph of $\mathcal{F}$ is a *path graph*. Path graphs are a subclass of chordal graphs since chordal graphs are characterized as the intersection graphs of subtrees of a tree[Gol04]. Path graphs are well studied in the literature [Ren70]–[Gol04]. Path graph recognition can be done in polynomial time[Gav78, Sch93]. Clearly, this is a necessary condition in terms of the intersection graph of the input hypergraph $\mathcal{F}$ in Feasbile Tree Path Labeling. However, one can easily obtain a counterexample to show the insufficiency of this condition. Path graph isomorphism is known be isomorphism-complete[KKLV10]. Therefore, it is unlikely that we can solve the problem of finding 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. To this effect we consider TPL on k-subdivided star and give a polynomial time solution for the same.

Whether TPL on general trees is solvable in P remains open. So do optimization opportunities in TPL (possible extentions to optimization for COP in the Section 2.2.).

REFERENCES

[ABH98] J. E. Atkins, E. G. Boman, and B. Hendrickson. A spectral algorithm for seriation and the consecutive ones problem. *SICOMP: SIAM Journal on Computing*, 28, 1998.

[AS95] Annexstein and Swaminathan. On testing consecutive-ones property in parallel. In *SPAA: Annual ACM Symposium on Parallel Algorithms and Architectures*, 1995.

[BL76] Kellogg S. Booth and George S. Lueker. 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, December 1976.

[Boo75] Kellogg S. Booth. *PQ-tree algorithms*. PhD thesis, Univ. California, Berkeley, 1975.

[BS03] David A. Bader and Sukanya Sreshta. A new parallel algorithm for planarity testing, April 11 2003.

[COR98] Thomas Christof, Marcus Oswald, and Gerhard Reinelt. Consecutive ones and a betweenness problem in computational biology. *Lecture Notes in Computer Science*, 1412, 1998.

[CY91] Lin Chen and Yaacov Yesha. Parallel recognition of the consecutive ones property with applications. *J. Algorithms*, 12(3):375–392, 1991.

[Dom08] Michael Dom. *Recognition, Generation, and Application of Binary Matrices with the Consecutive-Ones Property*. PhD thesis, Institut für Informatik, Friedrich-Schiller-Universität Jena, Germany, 2008. Published by Cuvillier, 2009.

[FG65] D. R. Fulkerson and O. A. Gross. Incidence matrices and interval graphs. *Pac. J. Math.*, 15:835–855, 1965.

[Gav78] Fanica Gavril. A recognition algorithm for the intersection graphs of paths in trees. *Discrete Mathematics*, 23(3):211 – 227, 1978.

[GH64] P. C. Gilmore and A. J. Hoffman. A characterization of comparability graphs and of interval graphs. *Can. J. Math.*, 16:539–548, 1964.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability*. Freeman, San Francisco, 1979.

[Gol04] Martin Charles Golumbic. *Algorithmic graph theory and perfect graphs*, volume 57 of *Annals of Discrete Mathematics*. Elsevier Science B.V., 2004. Second Edition.

[HG02] Hajiaghayi and Ganjali. A note on the consecutive ones submatrix problem. *IPL: Information Processing Letters*, 83, 2002.

[HL06] Dorit S. Hochbaum and Asaf Levin. Cyclical scheduling and multi-shift scheduling: Complexity and approximation algorithms. *Discrete Optimization*, 3(4):327–340, 2006.

[Hsu01] Wen-Lian Hsu. PC-trees vs. PQ-trees. *Lecture Notes in Computer Science*, 2108:207–217, 2001.

[Hsu02] Wen-Lian Hsu. A simple test for the consecutive ones property. *J. Algorithms*, 43(1):1–16, 2002.

[HT02] Hochbaum and Tucker. Minimax problems with bitonic matrices. *NETWORKS: Networks: An International Journal*, 40, 2002.

[JKC+04] Johnson, Krishnan, Chhugani, Kumar, and Venkatasubramanian. Compressing large boolean matrices using reordering techniques. In *VLDB: International Conference on Very Large Data Bases*. Morgan Kaufmann Publishers, 2004.

[KKLV10] Johannes Köbler, Sebastian Kuhnert, Bastian Laubner, and Oleg Verbitsky. Interval graphs: Canonical representation in logspace. *Electronic Colloquium on Computational Complexity (ECCC)*, 17:43, 2010.

[Kou77] Lawrence T. Kou. Polynomial complete consecutive information retrieval problems. *SIAM Journal on Computing*, 6(1):67–75, March 1977.

[Lau09] Bastian Laubner. Capturing polynomial time on interval graphs. *CoRR*, abs/0911.3799, 2009. informal publication.

[Lin92] Steven Lindell. A logspace algorithm for tree canonization (extended abstract). In *STOC*, pages 400–404. ACM, 1992.

[McC04] Ross M. McConnell. 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.

[MM96] J. Meidanis and Erasmo G. Munuera. A theory for the consecutive ones property. In *Proceedings of WSP'96 - Third South American Workshop on String Processing*, pages 194–202, 1996.

[MPT98] Meidanis, Porto, and Telles. On the consecutive ones property. *DAMATH: Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, 88, 1998.

[NS09] N. S. Narayanaswamy and R. Subashini. A new characterization of matrices with the consecutive ones property. *Discrete Applied Mathematics*, 157(18):3721–3727, 2009.

[Rei08] Omer Reingold. Undirected connectivity in log-space. *J. ACM*, 55(4), 2008.

[Ren70] Peter L. Renz. Intersection representations of graphs by arcs. *Pacific J. Math.*, 34(2):501–510, 1970.

[Sch93] Alejandro A. Schaffer. A faster algorithm to recognize undirected path graphs. *Discrete Applied Mathematics*, 43:261–295, 1993.

[TM05] Guilherme P. Telles and João Meidanis. Building PQR trees in almost-linear time. *Electronic Notes in Discrete Mathematics*, 19:33–39, 2005.

[Tuc72] Alan Tucker. A structure theorem for the consecutive 1's property. *J. Comb. Theory Series B*, 12:153–162, 1972.

[Vel85] Marinus Veldhorst. Approximation of the consecutive ones matrix augmentation problem. *SIAM Journal on Computing*, 14(3):709–729, August 1985.

6 Proposed Contents of the Thesis

The outline of the thesis is as follows:

Chapter 1 - Introduction
- Section 1.1 Consecutive Ones Testing
- Section 1.2 Matrix modification to attain COP or "almost" COP
- Section 1.3 Graph Isomorphism
- Section 1.4 Logspace complexity of graph canonization using COP
- Section 1.5 Motivation, Objective and Scope of Thesis
- Section 1.6 Summary of Results
- Section 1.7 Organization of document

Chapter 2 - Consecutive ones property
- Section 2.1 Characterization of COP
- Section 2.2 Recognition of COP
- Section 2.3 Alteration to matrices to get COP
- Section 2.4 Complexity of certain COP variations

Chapter 3 - Other problems related to COP

Chapter 4 Research
- Section 4.1 Tree path labeling of path hypergraphs
- Section 4.2 Extension of the structural characterization of matrices in [NS09]

Chapter 5 - Conclusion

Chapter 6 - Bibliography