Hypergraph-based Techniques for Pliable Index Coding

Thesis submitted in partial fulfillment
of the requirements for the degree of

Master of Science
in
Computer Science and Engineering by Research

by

Visvesh Subramanian
20171133
visvesh.subramanian@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
July 2024

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "*Hypergraph-based Techniques for Pliable Index Coding*" by Visvesh Subramanian , has been carried out under my supervision and is not submitted elsewhere for a degree.

_____ _____

Date Adviser: Prof. Dr. Prasad Krishnan

To loved ones and well-wishers.

Acknowledgments

I am grateful to my research advisor Dr. Prasad Krishnan, for his support and understanding. While working under his guidance, I learnt the values of patience and discipline, while we tackled problems together. My parents and brother have shown interest at times during the course of my research work and it has acted as a motivating factor. Their frequent check-ins on my well-being, both academically and otherwise also helped. I am also grateful for the support from my coauthor Mrs Tulasi.S, which enabled us to collaborate and make meaningful contributions. Additionally, I appreciate the friends I have made throughout the years at IIIT Hyderabad. I also thank the few friends from my batch who recommended SPCRC as a place for research. I enjoyed my final weeks and months at the institute (the summer semester), mostly spending that time at the SPCRC lab with friends.

Abstract

The Index Coding problem is a well known problem setting in the field of coding theory. A variant of this, called Pliable Index Coding (PICOD), was introduced in [5]. It is called "pliable" because there are multiple valid decoding choices whereas the index coding problem has only one valid decoding choice. Finding the optimal code length for both the index coding problem and the PICOD problem is computationally NP-hard. Hence various coding schemes have been proposed, with varying performance guarantees. Those coding schemes were designed by representing the PICOD problem in alternate ways. These achievability schemes upper bound the optimal length of the pliable index code. Lower bounds have also been obtained in prior literature by utilizing information-theoretic arguments. Following this trend, by representing the PICOD problem as a hypergraph, we obtain a new upper bound and lower bound. The parameter that obtains the lower bound is a novel parameter of the hypergraph representation of the PICOD problem, and an algorithm to obtain it from a given PICOD problem is also described. The upper bound parameter is the maximum degree of any vertex in the hypergraph representation of the PICOD problem, and it gives simple upper bound for the optimal length of the PICOD problem. These two bounds are shown to be tight for a class of PICOD problems. Another upper bound was also explored viz. Γ which is another parameter related to the hypergraph representation of the PICOD problem. Additionally, when this parameter is small, it is possible to categorize and analyze each class of the PICOD problem exhaustively, which was also done.

Contents

List of Figures

Notations

$[m]$	The set of integers $\{1, 2, 3, \ldots, m\}$
$\beta_q(\mathcal{H})$	The optimal number of transmissions required to solve a PICOD problem represented by $\mathcal{H}$ over alphabet F_q
Δ_j	degree of vertex j in hypergraph
$\mathcal{E}(\mathcal{H})$	edges in hypergraph $\mathcal{H}$
$\mathcal{H}$	A hypergraph
$\mathcal{V}(\mathcal{H})$	Vertices in hypergraph $\mathcal{H}$
F_q	Finite field over alphabet q where q is a prime integer
PICOD	Abbreviation for Pliable Index Coding

Chapter 1

Introduction

Communication systems are systems that can be used to transmit data over any type of channel. They are based on the idea of dividing the data into smaller units and then transmitting these units of data over the channel. These packets may be encoded at the source to reduce the number of transmissions or to introduce redundancy to boost the fault tolerance of the communication system. This information, once received, will have to be decoded at the receiver's end.

Index Coding, introduced by Birk and Kol in 1998 [3], is a well-known problem setting involving a central server and multiple clients. The motivating example given by the authors was to minimize the number of transmissions in satellite communication. Satellite mode of communications is broadcast in nature. Hence when it communicates with multiple base stations on the earth simultaneously, some of those base stations receive messages they don't need. Thus they collect and store the messages which are not required for them, to use them to decode the messages that are arriving later.

It can be formally characterized by the following parameters: a server which contains m messages denoted by $\{b_i : i \in [m]\}$, b_i lies in some finite alphabet set $\mathcal{A}$, and n clients (or receivers) indexed by $[n]$. The receiver r has with it a set of messages, which is called side information $\{b_i : i \in S_r\}$ where $S_r \subseteq [m]$. The set of all the requested message indices by the different receivers is called the decoding choice and is denoted by a multiset $\mathcal{D}$. An index code is a set of transmissions made by the server to satisfy the decoding choices of the clients. A client is considered to be satisfied only if it can decode the particular message as specified by its corresponding index in the decoding choice.

The index coding problem has been tackled in various ways. By representing the index coding problem graphically, various graph theoretic generalizations of vertex colouring have been explored in prior research, for instance, fractional clique covering [4], fractional local partial clique covering [1], fractional local graph colouring [11] ,[2] and interlinked cycle covering [14], further upper bounds and achievable schemes were given for the index coding problem. Linear programming and semidefinite programming were also used to give better bounds for the index coding rate [4]. We now consider an example of an index coding problem.

Example 1. *Consider three receivers r_1, r_2, r_3 having side information sets $S_1 = \{2\}, S_2 = \{3\}, S_3 = \{1\}$ respectively. Their decoding choices are denoted by $\mathcal{D} = \{1, 2, 3\}$.*

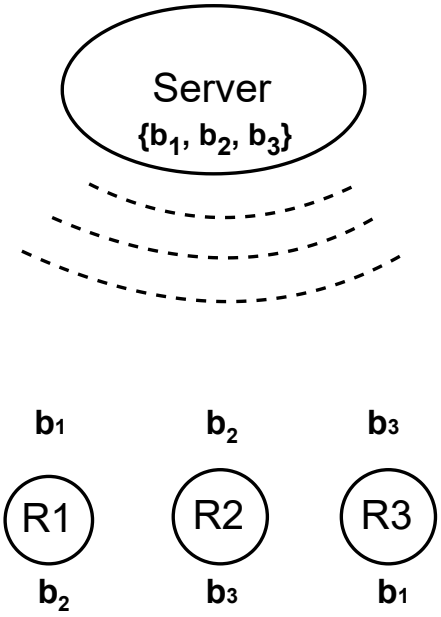

Figure 1.1: The PICOD example described above is shown graphically. The message above the receiver satisfies the receiver (its decoding choice), and the message below is in the side information set.

A solution to the index coding problem will be to transmit the coded messages $(b_1 + b_2)$ *and* $(b_2 + b_3)$*. This is enough to satisfy all three because:*

1. *Receiver* r_1 *subtracts* b_2 *(which is present in its side information set) from* $b_1 + b_2$ *to get* b_1*.*
2. r_2 *subtracts* b_3 *from* $b_2 + b_3$ *to get* b_2*.*
3. r_3 *has* $b_1 + b_2$ *and* $b_2 + b_3$ *and side information* b_1*, hence it is easy to decode* b_3 *from these 3 expressions.*

A variant of the index coding problem called Pliable Index Coding (PICOD) was introduced by Brahma and Fragouli in [5]. The PICOD problem is an adaptation of the index coding problem, initially introduced by [5]. This also involves a server and n clients with side information sets. However, in PICOD, we assume that clients are pliable (flexible in their requirements) and are content with any of the messages that they do not possess. The transmission length for the pliable index coding scenario could be much lower than in the index coding scenario. It was also demonstrated in Section 4 of [5] that finding an optimal solution to a PICOD problem is NP-HARD, even though bounds are significantly better than index coding problems which is also NP-HARD.

Using the same index coding setting as above, when we solve the PICOD variant, we get many more transmission schemes of length 2. One of them is $\{b_1, b_2\}$. Receivers r_1 and r_2 both have b_1 in their

request sets and are satisfied by this coding scheme. The message, b_2 is present in the request set of receiver r_3, hence, this is also satisfied by the coding scheme.

1.1 The Pliable Index Coding Problem

The pliable index coding problem (PICOD) is an adapted version of the index coding problem, aiming to minimize the number of transmissions needed to satisfy all receivers. The NP-hardness of finding an optimal length pliable index code for the PICOD problem has been considered and proven in [5]. A probabilistic argument to support the existence of codes with specific lengths has also been done in the PICOD literature as described in the following paragraph. Algorithms using principles such as greedy and set cover techniques such as the ones described in [5] are introduced as a means to construct pliable index codes.

PICOD and PICOD-type settings find use in many distributed computing scenarios. Similar to the satellite broadcast motivating example given by [3], content distribution networks (CDN's) might have to solve such a problem as the PICOD problem. Distributed machine learning algorithms eg. Batch and Epoch training are done by sampling a small fraction as the test data to prevent overfitting. the PICOD problem randomly assigns a portion of the data, hence it may find use here. For data shuffling-related problems, the above reasoning could be given as well. PICOD-type solutions are favourable when it is a broadcast medium, and when the desired solution is probabilistic/heuristic in nature.

Since it has been shown that the pliable index coding (PICOD) problem is NP-hard, various polynomial time schemes perform differently with different PICOD settings. A probabilistic argument was used in [5] to prove the existence of a code with a length of order $O(\min\{\log m \times (1+\log^+(\frac{n}{\log m})), m, n\})$, where $\log^+(x) = \max\{0, \log(x)\}$. When $m = n^\delta$ for some constant $\delta > 0$, the bound gets simplified and implies that $O(\log^2 n)$ is sufficient. Several algorithms based on greedy and set cover techniques have been introduced and compared for developing pliable index codes. In a related study [12], a polynomial-time algorithm for the general PICOD problem is presented, achieving a length of $O(\log^2 n)$ shown in [12]. Therefore unlike the index coding problem which requires instances of size $\Theta(\sqrt{n})$, an implementation for PICOD instances can typically achieve the desired result with significantly fewer transmissions. Pliable index coding often requires exponentially fewer transmissions in the worst case, with a complexity of $O(\log^2 n)$. This result implies that recognizing the need to solve a pliable index coding problem instead of a conventional index coding problem can lead to exponential efficiency gains in terms of transmission and reception.

We note that solving the index coding for all possible decoding schemes and picking the best is not a viable strategy as the implementation of such an algorithm is not polynomial time. Hence, only polynomial time algorithms are considered interesting. The question then becomes how to accomplish this, given the inherent difficulty of the flexible nature of the pliable index coding problem.

Any PICOD problem can be equivalently represented using a hypergraph $\mathcal{H}$ with the vertices representing the messages, and the request sets as hyperedges, where the request set of the client r is given

by the complement of its side information set, i.e. $I_r = [m] \setminus S_r$. Formally, β_q represents the optimal number of transmissions needed to satisfy a PICOD problem represented with a hypergraph $\mathcal{H}$, using an alphabet of size q (typically, we use the finite field F_q). This value β_q is the minimum over all the possible decoding choices for corresponding index coding problems represented with the hypergraph $\mathcal{H}$ and whose decoding choices are $\mathcal{D}$. This index coding problem is represented by $\mathcal{H}_{\mathcal{D}}$.

$$\beta_q(\mathcal{H}) = \min_{\mathcal{D}} \beta_q(\mathcal{H}_{\mathcal{D}})$$

The results of this thesis essentially present upper and lower bounds on $\beta_q(\mathcal{H})$.

1.2 Contributions and Structure of this thesis

This thesis is an overview of the PICOD problem and a few other similar problem settings. Chapter 2 presents a review of the literature on PICOD. This chapter begins by discussing the characterization of the problem, which is the process of defining the problem and its parameters. We then discuss alternative representations of the PICOD problem, such as hypergraphs, bipartite graphs, and incidence matrices. These representations provide different ways of looking at the problem and can be used to develop new solutions, especially graph theoretic methods that exist in the literature.

Chapter 3 then discusses bounds on the optimal length of the problem. This is the main contribution of the thesis, and it is based on the paper [13] that was published in ITW 2022. The bounds provide an upper limit on the length of a solution to the problem, by describing and justifying an achievable scheme that takes at most $\Delta(\mathcal{H})$ transmissions, where $\Delta(\mathcal{H})$ is the maximum degree of any vertex in the hypergraph $\mathcal{H}$ which represents the PICOD problem. The thesis also discusses the proof of the converse bound using acyclic graphs. We also introduce a novel parameter $\eta(\mathcal{H})$ called the nesting number, and show that $\eta(\mathcal{H})$ is a lower bound for the optimal PICOD length for $\mathcal{H}$. A class of hypergraphs where both the upper and lower bounds meet is described, which shows that the bounds are tight. An algorithm for finding $\eta(\mathcal{H})$ is then presented. This makes the findings regarding the lower bound more practical since the $\eta(\mathcal{H})$ bound is only applicable once we can determine $\eta(\mathcal{H})$.

In the following chapter (Chapter 4), we look at another connectivity parameter, namely Γ. The relation of some connectivity parameters with S_{max} and S_{min}, which are the maximum and minimum sizes of any hyperedge is also discussed. The thesis also discusses PICOD problems with small Γ, where Γ is the edge overlap parameter of a hypergraph. When Γ is small (equal to 0, 1, or 2) it is of particular interest because it can be solved using simple techniques, and analyzing cases exhaustively is possible. The conclusion of the thesis summarizes the main findings and provides its relation to existing PICOD schemes.

Chapter 2

Literature Review

In this chapter, the PICOD problem is defined using mathematical notations. The alternative ways to represent this problem as seen in the existing literature are described. There are multiple ways of coming up with schemes to satisfy all clients, each guaranteeing different bounds, and different average code lengths when simulated. The following study of existing literature will help us to set the context for subsequent chapters of this thesis, where we present our contributions.

We consider the concept of hypergraphs as a representation of the PICOD problem, where messages of the PICOD problem correspond to vertices of the PICOD hypergraph and receivers' request sets define hyperedges of the PICOD hypergraph. Hypergraphs allow for accurate modelling of the relationships between messages and receivers. This enables us to utilize tools from hypergraph theory to construct achievability and converse schemes for PICOD. The parameters of hypergraphs, such as edge overlap parameter, conflict-free chromatic number, and maximum vertex degree, are explored to understand their effects on the minimum number of transmissions required to solve the PICOD problem. The section concludes by presenting various bounds on the number of transmissions based on these hypergraph parameters. This chapter also discusses the representation of the PICOD problem using bipartite graphs and incidence matrices, which facilitate the application of graph algorithms and matrix-oriented techniques.

2.1 Formal Definition of PICOD Problem

A PICOD problem can be characterized by the following parameters: m messages denoted by $\{x_i : i \in [m]\}$ where x_i lies in some finite alphabet set $\mathcal{A}$. The n clients are indexed by $[n]$. The receiver r has a subset of messages with it and is called its side information ($\{x_i : i \in S_r\}$), and S_r is the set of indices of those messages it has. The receiver r demands any message that it does not already have in its side information set. The set of indices of messages that the client r doesn't have is $I_r \triangleq [m] \setminus S_r$, and is called the request set. It comprises the messages $\{x_i : i \in I_r\}$. The set of all request sets is represented by $\mathcal{J} \triangleq \{I_r : r \in [n]\}$. The parameters $(n, m, \mathcal{J})$ characterize a PICOD problem.

To satisfy the receivers, the server transmits encoded symbols to the receivers in a broadcast link. Encoding is done at the server end and is denoted by the following function $f : F_q^m \rightarrow F_q^K$, where K is the total number of transmissions or code length. Let $f(b_1, b_2, ...b_m) = (x_1, x_2, ...x_K)$ be the K transmitted messages. Also, we recall that the server has all the messages b_1 through b_m, and the various clients have different unique subsets of what the server has.

The decoding functions are applied to the receivers and are denoted by the set of functions: $\phi_i : F_q^K \times F_q^{|S_i|} \rightarrow F_q \times [m]$ for $i \in [n]$. The process at the receiver i is as follows: $\phi_i(\{x_k\}_{k\in[K]}, \{b_j\}_{j\in S_i}) = (b_k, k : k \in I_i)$ receiver i receives message j and its index.

A linear encoding (and corresponding decoding) scheme employs linear functions of the input symbols which we describe as follows.

- In the encoding phase, the i^{th} broadcast symbol is the linear combination $\sum_{k=1}^{m} a_{k_i} b_k$ where $a_{k_i} \in F_q$ is the encoding coefficient.
- Succinctly, this can be shown as a product of the message vector $\mathbf{b}$ and $K \times m$ coefficient matrix A as $\mathbf{x} = A\mathbf{b}$.
- In the decoding phase, we first remark that the encoding scheme will be known to all receivers. This can be done by transmitting the encoding matrix A to all receivers. The transmission overhead is negligible in comparison to any coded symbols x_i.
- For a given receiver i, it knows A, x, and $\{b_j | j \in S_i\}$, it will have to solve for b_j for some $j \in I_i$.
- Since this receiver has a side information set, each broadcast message $x_i, i \in [K]$ will be modified to $x_k^{(i)} = x_k - \sum_{j\in S_i} a_{k_j} b_j$, and this is done for all $k \in [K]$.
- This can be represented in vector form as $\mathbf{A_{r_i}} \mathbf{b_{r_i}} = x^{(i)}$, where $\mathbf{A_{r_i}}$ is the submatrix of $\mathbf{A}$ which only comprises those columns of $\mathbf{A}$ whose indices are in the request set of index i.
- Formally $\mathbf{A_{r_i}} = \{\mathbf{A_j}\} \forall j \in I_i$, and $\mathbf{b_{r_i}}$ is the set of messages in the request set of client i.

If from the above equation, we can solve for at least one value in the request set (denoted by the vector $\mathbf{b_{r_i}}$), we can consider the client i to be satisfied. If even one receiver is not satisfied, then K (number of transmissions) has to be increased.

A more generic case of the above-mentioned PICOD problem is the multi-request PICOD problem and is often characterised by the notation PICOD(t). Here a client gets satisfied only if it can decode any t messages that are in its request set. If $t < |I_r|$, It must be able to decode all the messages to be considered satisfied. A trivial achievable scheme for this generic case is to run the algorithm for the single-request PICOD t times. However, researchers were able to come up with schemes that perform much better than this trivial scheme, such as in [12], where the t-request PICOD code length is considerably better than the PICOD code length multiplied by a factor of t.

2.2 Representations of the PICOD problem

The PICOD problem can be represented in many ways, and each representation gives some insight into developing different coding schemes. This helps us in describing our approach to designing our achievability scheme. Tthree representations that are discussed here are the hypergraph, bipartite graph, and incidence matrix. These various representations also allow us to use already obtained results from graph theory and combinatorics to construct PICOD schemes.

2.2.1 Hypergraph Representation of PICOD

Hypergraphs provide a representation of the PICOD problem. Unlike traditional graphs, hypergraphs allow for more flexible and accurate modelling of the relationships between the messages and receivers. In the PICOD problem, the messages correspond to vertices, and the receivers' request sets define the hyperedges. By using hypergraphs, we can explicitly capture the dependencies between subsets of messages and receivers. The hyperedges in the hypergraph precisely indicate which messages are required by each receiver for successful decoding. They provide a clear visual representation that aids in understanding the problem structure, identifying patterns, and detecting dependencies.

Example 2. *Consider a Hypergraph $\mathcal{H}(V, E)$ such that $V = \{1, 2, 3, 4, 5, 6, 7\}$ and $E = \{\{1, 2, 3\}, \{2, 4, 5\}, \{1, 3, 6\}, \{2, 6\}, \{5, 6, 7\}\}$. Since the hyperedges correspond to clients and vertices correspond to messages, there are 7 messages (represented by vertices) and 5 clients (represented by the edges). It's not hard to see that any single linear combination of messages b_1 through b_9 can't satisfy all clients, hence we need at least 2 transmissions. One optimal scheme for the PICOD problem corresponding to the above hypergraph is $\{b_2 + b_7, b_6\}$. The first coded symbol $b_2 + b_7$ can be used to decode b_2 for clients c_1,c_2 and c_4, and decode b_7 for c_5. This transmission is not helpful for c_3, because both b_2 and b_7 are already in its side information set. The next transmission of b_6 will satisfy it. If we consider the PICOD(t) problem where $t = 2$, then one trivial coding scheme is to solve the PICOD problem twice. For the first iteration, we would follow the same steps as above and get the 2 transmissions, the problem would then change to: $E = \{\{1, 3\}, \{4, 5\}, \{1, 3\}, \{6\}, \{5, 6\}\}$, with the vertex set remaining the same. We see that the coded message $b_3 + b_4 + b_6$ is enough to satisfy all remaining clients in one transmission. Hence $\{b_2 + b_7, b_6, b_3 + b_4 + b_6\}$ is one achievable coding scheme for the corresponding PICOD(2) problem for hypergraph $\mathcal{H}$.*

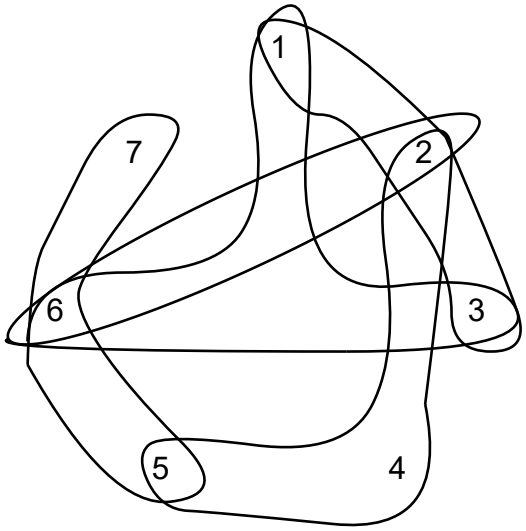

Figure 2.1: The hypergraph shown represents the PICOD problem of Example 2.

The above hypergraph represents the PICOD problem defined in example 2.

Connectivity parameters refer to the properties that define how the vertices and hyperedges are connected or related to each other. These parameters play an important role in characterizing the structure and connectivity of hypergraphs. By analyzing these connectivity parameters, we can identify favourable structural properties in the hypergraph that ensure a lower number of transmissions. These help in designing algorithms, and establishing theoretical bounds for PICOD problems. More specifically we are interested in how these connectivity parameters affect the minimum number of transmissions needed to solve a PICOD problem represented by the hypergraph $\mathcal{H}$. This number is denoted by $\beta_q(\mathcal{H})$. The list of parameters associated with the PICOD hypergraph, which we use in this thesis.

- n is the number of hyperedges.
- m is the number of vertices.
- $\Gamma(\mathcal{H})$ is the edge-overlap parameter which tells us the maximum number of other hyperedges incident on (or overlapping with) any one hyperedge.
- $\Delta(\mathcal{H})$ is the maximum degree of any vertex in hypergraph $\mathcal{H}$, where the degree of a vertex is the number of hyperedges incident on it.

Definition 1 (Connected Component)**.** *A connected component is a substructure of a hypergraph such that every pair of vertices within the substructure can be reached by traversing through the overlapping hyperedges of the hypergraph.*

This connectivity ensures that there is a path of hyperedges that links any two vertices within the component. A minor but important observation when dealing with hypergraphs is that there might not always be a single connected component. When a hypergraph represents a PICOD problem, and we're interested in obtaining the minimum number of transmissions needed to satisfy the PICOD problem, we

can solve each connected component independently. Then the minimum number of transmissions will be equal to the maximum of all values obtained for each connected component as shown in the claim below.

Claim 1. *Suppose a hypergraph has more than one connected component, then the minimum number of transmissions needed to satisfy its corresponding PICOD problem is the maximum of all values obtained when each connected component is solved separately.*

Proof. Suppose cc_1 and cc_2 are two connected components that don't intersect. Let $a = \{a_1, a_2, ...a_{n_1}\}$ be the optimal PICOD code for PICOD problem represented by cc_1 and $b = \{b_1, b_2, ...b_{n_2}\}$ be the optimal PICOD code for PICOD problem represented by cc_2. Then we can see that any linear combination of a_i and b_i will not affect the ability of receivers in cc_1 and cc_2 to decode, because they will have b_i and a_i in their side information sets respectively. This argument can be extended to all pairs of non intersecting connected components in the hypergraph. Each connected component will produce a separate code. Hence the i^{th} decoding message of each code can be linearly combined (if the i^{th} message is not present, then we use 0 in its place) for the coding scheme of the entire PICOD problem. It is easy to see that the length of such a code will be the maximum of the code lengths for each of the connected components. □

2.2.2 Bipartite Graph Representation of PICOD

A PICOD (Pliable Index Coding) problem can be represented as a bipartite graph without using matrix notation. In a bipartite graph representation, we have two sets of vertices: message vertices and client vertices. An edge between a message vertex and a client vertex indicates that the client requests that message. The absence of an edge represents no request. It allows us to apply graph algorithms and techniques to analyze and solve the index coding problem efficiently.

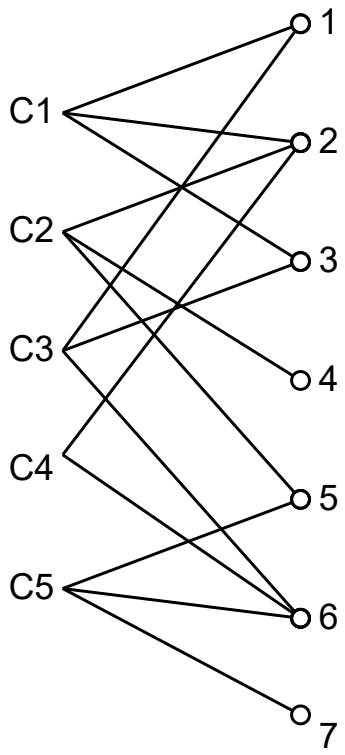

Figure 2.2: The bipartite graph shown represents the PICOD problem of Example 2.
The above bipartite graph represents the Example 2.

2.2.3 Incidence Matrix Representation of PICOD

A PICOD problem can be represented as an incidence matrix, also referred to as a bipartite matrix. The incidence matrix represents the relationships between the messages and the clients in the index coding problem.

The incidence matrix is constructed based on the connections between the messages and the clients. The columns of the matrix correspond to the message vertices, and the rows correspond to the client vertices. The entry in the matrix indicates whether a particular message is needed by a specific client. A non-zero entry represents a connection, indicating that the corresponding client demands the corresponding message. In the incidence matrix representing this PICOD problem, each column corresponds to a message, and each row corresponds to a client. A ”1” in the matrix indicates that the corresponding message is demanded by the corresponding client, while a ”0” indicates no demand. By analyzing the incidence matrix, we can understand the connections between the messages and clients in the PICOD problem. This representation allows us to apply various algorithms and techniques that are matrix oriented, such as the greedy algorithm mentioned below, and others, to solve the pliable index coding problem efficiently.

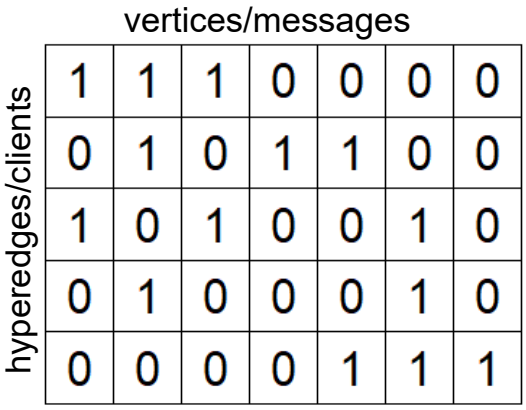

Figure 2.3: The incidence matrix shown represents the PICOD problem of Example 2.

2.3 Prior Achievability Schemes (Known Upper Bounds for Optimal PICOD Length)

Achievability schemes for the PICOD problem have been devised by researchers in the past. It is helpful to first go through them because we can get an idea of what the state-of-the-art encoding schemes can achieve in terms of code length. Chapter 3 will detail an achievability scheme and make comparisons with these existing schemes because their approaches may have some similarities. Achievable schemes are also of interest because they give us upper bounds on the minimum number of transmissions required for a PICOD problem. They also give us ideas on how to approach such graph-centric problems in mathematics, which are NP-Hard. Having seen ways of representing the PICOD problem, we shall now look at the encoding schemes that have utilised each of them.

2.3.1 GrCov and RandCov [5]

The GrCov and RandCov algorithms use the bi-partite graph representation for the PICOD problem while designing the coding scheme. In the original PICOD paper [5], two basic polynomial time heuristic approximation algorithms were described: GrCov and RandCov. In GrCov, short for greedy cover, the encoding phase proceeds iteratively, where in each iteration we sum up a set of selected messages for broadcasting. The process of selection is fairly straightforward. First, initialize an empty set B, and iterate through all messages. For each message, compute the number of satisfied clients before and after the addition of the message to set B. Add the message to set B if by adding, it increases the number of satisfied clients. In other words we choose a maximal set, not the set which has the maximum number of messages. Decoding happens on a transmission by transmission basis, i.e. it doesn't remember previous transmissions and attempts to decode using a combination of transmissions. In the case each client makes t requests, each of the clients maintains a counter, and instead of getting removed when satisfied (like in the single requests scenario), the counter gets decremented, and the client is removed when the

counter reaches 0. The authors of this paper conjectured that the number of transmissions is $O(log(n))$ times the length of the optimal linear code. This algorithm is called greedy cover because we greedily choose the maximal (not maximum) subset of messages to be broadcasted in every iteration. This paper and a few others use the term "effective degree", so we formally define it below.

Definition 2. *Effective degree and effective clients: Given a particular order of the message vertices $\pi = (j_1, j_2, ..., j_m)$, the effective degree of message b_{jl} is defined as the number of b_{jl} 's neighbours who do not connect with message $b_{j'}$, for any $j' = j_1, j_2, ..., j_{l-1}$. The neighbours contributing to b_{jl} 's effective degree are called effective clients of b_{jl} . Let us denote by $N[j]$ the set of neighbors of message b_j and by $N[j_1, j_2, ..., j_{l-1}]$ the set $N[j_1] \bigcup N[j_2] \bigcup ...N[j_{l-1}]$.*

Formally, the effective clients of message b_{jl} are defined as $N^{\dagger}{}_{\pi}[jl] = N[jl] \setminus N[j1, j2, ..., jl-1]$ with respect to the order π. Correspondingly, the effective degree of message b_{jl} is defined as $|N^{\dagger}{}_{\pi}[jl]|$ with respect to π.

In RandCov, short for randomized cover, we split the clients into non-empty groups $C = \{C_1, C_2, ...C_g\}$, such that in each group, the maximum number of messages that any client can be satisfied by is not more than r times the number of messages any other client can be satisfied by, where r is some constant. It is easy to see that due to this constraint, there can only be at maximum $g \leq log_r(n)$ groups. Also, the message set gets split into $\{B_1, B_2...B_g\}$ such that $\forall b \in B_i$, where b is in the request set of client c if $c \in C_i$. If in group C_i, the maximum number of messages that can satisfy any one client is $d_{max,i}$, then we pick messages from B_i with probability $\frac{1}{d_{max,i}}$. This process is repeated in each group until all clients are satisfied. It is called randomized cover due to the randomized attempt at picking an independent set of messages, instead of using a deterministic algorithm. This trades-off efficiency of code length for achieving a lower running time complexity, which was calculated to be $\log(mn \log n)$.

2.3.2 BinGreedy [12]

The BinGreedy encoding algorithm uses the incidence matrix representation while designing the coding scheme. This deterministic algorithm guarantees an upper bound of $O(\log^2 n)$ transmissions. The algorithm broadly runs in three phases: sorting, grouping and transmitting. This splits the PICOD problem into $\leq \log(n)$ groups and satisfies a fraction of all clients with one transmission per group. Hence it takes $O(\log(n))$ transmissions per group to satisfy all clients in that group. The theorem 1 in the work [12] shows us the precise maximum number of transmissions overall is $\frac{2}{\log(1.5)} \log^2 n$. In the t-requests case, a similar approach was used, but unlike GrCov, a weightage function was used for clients to prioritize those clients which needed more messages to be satisfied. This helped in ensuring that no client was getting prioritized over others. A trivial upper bound would be obtained by simply repeating the PICOD problem t times, giving us the bound $t \log^2 n$. The paper showed that the upper bound was much lower than this trivial bound, and it was analytically proven to be $O(t \log(n) + \log^2 n)$.

2.3.3 Algorithm via conflict-free colorings of PICOD Hypergraphs [6]

As the name suggests, the arguments made in the paper [6] are made for hypergraph representations of the PICOD problem. A conflict-free colouring of a hypergraph $\mathcal{H}(\mathcal{V}, \mathcal{E})$ as described in the paper [6] as a colouring of the vertices of $\mathcal{H}$ such that every hyperedge in $\mathcal{H}$ gets at least one colour occurring exactly once. The conflict-free chromatic number $\chi_{CF}(\mathcal{H})$ is the minimum number of colours required for such a colouring. Formally, let $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ be a hypergraph with vertex set $\mathcal{V}$ and hyperedge set $\mathcal{E}$. A conflict-free colouring of $\mathcal{H}$ is a function $C : \mathcal{V} \to [L]$, where $[L]$ denotes the set of colours, such that for every hyperedge $e \in \mathcal{E}$, there exists a vertex v in e such that $C(v)$ is distinct from the colours assigned to the other vertices in e. Different colours represent different broadcast indices, so if vertices have the same colour, their corresponding messages in the PICOD problem are summed up and broadcast together. It is clear that in the process of obtaining this chromatic number, we also solve the associated PICOD problem. A conflict free collection of colourings is a more general case where multiple colourings are used, and a client is satisfied if at least one colouring scheme in the collection satisfies it. This was termed the "conflict free covering number" by the paper and is denoted by α_{CF}. An example was also shown where fewer colours were required in case of multiple colourings with few colours each as opposed to a single colouring with many colours.

$\Gamma(H)$ is the edge-overlap parameter which we also denote by Γ , it represents the maximum number of hyperedges $\mathcal{H}$ that intersect with any one particular hyperedge. It can be thought of as a measure of the density of the hypergraph. This itself is an upper bound on the number of transmissions, $\Gamma(\mathcal{H}) \geq \beta_q$, and it is proven in theorem 3 . Γ is formally defined to be $max_{r\in[n]}|\{r' \in [n] \setminus r : I_r \bigcap I_{r'} \neq \phi\}|$, where n is the number of hyperedges, I_j is the set of vertices in the hyperedge j. Theorem 4 in [6] proves the existence of a pliable index code of length $O(\log^2 \Gamma)$ and shows a probabilistic coding scheme which constructs this code with a probability of $1 - \frac{2}{(m+n)}$, and with run time complexity of $O(m^5 n^3 \log \Gamma/\Gamma) \cdot O(\log^2 \log \Gamma)$.

The other important result in the mentioned paper, relevant to this thesis is Theorem 8, which also proves an existence result. The theorem states that in the t-request scenario of the PICOD problem $\mathcal{H}(\mathcal{V} = [m], \mathcal{I})$, with m messages and edge-overlap parameter Γ, there exists a PIC (Pliable Index Code) that satisfies all clients and has a length $l = max(O(\log \Gamma \log m), O(t \log m))$ over any field size $\geq m$, only if $m \geq 12 \max(t, \log(6(\Gamma + 1))$. There is also a binary PIC satisfying all clients with the same code length if $|I_r| \geq 12 \max(t, \log(6(\Gamma + 1))), \forall I_r \in |\mathcal{I}|$. These PICOD schemes can be constructed with a running time of $O(m^4 n^2 \log^2 m/\Gamma)$.

2.3.4 Other bounds in the literature

The paper [10] provided codes for two cases of the PICOD problem, one in which each client can decode exactly one message and one in which the total number of messages that the effective clients can decode is maximised. The paper [7] introduces the idea of complete-S PICOD(t). $S \subseteq [0 : m - t]$ where m is the number of messages and t is the number of requested messages per client. S is the

unique set of sizes of request sets in the PICOD problem, and the complete-S problem is the scenario in which all possible side information subsets of size S_i are present $\forall S_i \in S$. Therefore there are $n = \Sigma_{s \in S} \binom{m}{s}$ clients in this PICOD problem. It was shown in this paper that an achievable code scheme of length $\sum_{i \in [\mathcal{S}]} \min\{m - \min_{s \in S_i}\{s\}, \max_{s \in S_i}\{s\} + t\}$ where $\mathcal{S}$ is a partition of S. This minimizing over all such partitions we get: $\min_{\mathcal{S}} \sum_{i \in [\mathcal{S}]} \min\{m - \min_{s \in S_i}\{s\}, \max_{s \in S_i}\{s\} + t\}$.

In the paper which introduced the PICOD problem [5], a probabilistic argument was employed to show the existence of a coding scheme with the following bound: $O(min\{\log m(1+log^{+}(\frac{n}{\log m})), m, n\})$ where $\log^{+}(x) = max\{0, log(x)\}$.

2.4 Lower Bounds from the literature on the Optimal Length of PICOD

In the paper [9], a lower bound was shown using the properties of the set of absent receivers. The first step was to consider all possible receive sets (power set of the message set), and remove the subsets that represent the receivers which are present. Formally, the set of all present receivrs $\mathbb{U} = 2^{[1:m]} \setminus \{[1:m]\}$, and the absent receiver set is $\mathbb{U}^{abs} = 2^{[1:m]} \setminus \{[1:m] \cup \mathbb{U}\}$. The lower bound on the hypergraph described by $\mathbb{U}$ is $m - L_{max}$, where L_{max} is the length of the longest nested chain of absent receivers. If $H_i \in \mathbb{U}$, and $H_1 \subseteq H_2 \subseteq ... \subseteq H_{L_{max}}$, then this is a nested chain of absent receivers. and L_{max} is its length.

In the paper [6], lower bounds were obtained when the hypergraph $\mathcal{H} = (V, \mathcal{E})$ meets a specific constraints on its structure. The optimal scalar linear PICOD length for such problems in the t-request case (denoted by $l^{*(t)}(\mathcal{H})$ in the work) are shown to be asymptotically tight, up to a multiplicative factor of $\log t$. More precisely, it was shown that $l^{*(t)}(\mathcal{H}) = \Omega(t \log m / \log t)$ thus describing a lower bound for such graphs, and $l^{*(t)}(\mathcal{H}) = O(t \log m)$ by showing an achievability scheme.

Chapter 3

Bounds on the Optimal Number of Transmissions

$\beta(\mathcal{H})$ is the optimal number of transmissions required to satisfy all the clients in a PICOD problem. In this chapter, we obtain novel upper and lower bounds on the number of transmissions in an optimal PICOD scheme. Upper bounds are obtained by presenting efficient transmission schemes that minimize the number of transmissions required. Lower bounds provide an intuition for evaluating the complexity of the problem and the expected performance of coding schemes.

For the upper bound, we present Algorithm 3.1.1 in this chapter, which constructs a PICOD scheme with a length at most $\Delta(\mathcal{H})$, where $\Delta(\mathcal{H})$ represents the maximum degree of any vertex in the hypergraph that represents the PICOD problem. The algorithm uses a vertex coloring approach and finds maximal independent sets of vertices to construct transmissions. The correctness of the algorithm is discussed, and its time complexity is shown to be polynomial in the system parameters.

Regarding the lower bound, we introduce a new structural parameter called the nesting number of the PICOD hypergraph (denoted by $\eta(\mathcal{H})$). By identifying a nested collection of hyperedges in the hypergraph, we derive a lower bound for the optimal PICOD length based on the nesting length. This lower bound provides insights into the minimum number of transmissions required and highlights the inherent complexity of the PICOD problem. There may be other lower bounds as well, and these lower bounds must be proven information-theoretically. We also note that the lower bound provided here applies to nonlinear coding and decoding schemes as well.

This chapter introduces an upper bound for the number of transmissions in the PICOD problem based on the maximum vertex degree ($\Delta(\mathcal{H})$, or in short Δ) of the hypergraph representation. The theorem states that $\beta_q(\mathcal{H}) \leq \Delta(\mathcal{H})$, meaning that the number of transmissions can be bounded by the maximum degree of the hypergraph. The algorithm for achieving this bound involves selecting vertices of the highest degree and constructing a maximal independent set on them. This reduces the maximum degree of the hypergraph, guaranteeing an upper bound of Δ on the number of transmissions. The algorithm works on a copy of the hypergraph and involves making deletions. In this chapter, we present a lower bound in the context of PICOD hypergraphs. It introduces the notion of a "nested collection," which is a substructure within the hypergraph that resembles a segment tree. By identifying and analyzing this nested collection, a lower bound on the number of transmissions can be derived based on its nesting

length. The section also presents an iterative method that constructs an acyclic subgraph to prove the lower bound, demonstrating that the process can go on for exactly $\log^2(L+1)$ steps, where L represents the length of the nested collection.

3.1 Upper Bound

An important connecting parameter for hypergraphs is the maximum vertex degree, denoted by $\Delta(\mathcal{H})$. It can be guaranteed that the total number of transmissions when following the achievable scheme shown in Algorithm 3.1.1 will be at most $\Delta(\mathcal{H})$. The correctness of the algorithm is ensured by the construction of independent sets of hyperedges that satisfy clients that correspond to those hyperedges. Since this algorithm involves the removal of vertices and hyperedges of the hypergraph, we perform the entire algorithm on a copy of the initial hypergraph $\mathcal{H}'$. The algorithm keeps removing a certain number of hyperedges and terminates when there are no more hyperedges remaining in $\mathcal{H}'$. At this point, it is guaranteed that all clients are satisfied. The number of transmissions in Algorithm 3.1.1 is at most $\Delta(\mathcal{H})$, thus providing an achievable scheme with a length at most $\Delta(\mathcal{H})$.

We now rigorously prove why the Algorithm 3.1.1 terminates in utmost Δ steps, We prove it in Theorem 1 with the help of the following Lemma and Observation.

Lemma 1. *For a hypergraph $\mathcal{H}$ with vertex set $\mathcal{V}$, If I is an independent set of vertices in $\mathcal{H}$, then, $\beta_q(\mathcal{H}) \leq 1 + \beta_q(\mathcal{H}[\mathcal{V} \setminus I])$.*

Proof. For the independent set I, let the server make one broadcast transmission of the symbol x derived as follows,

$$x = \sum_{i \in I} b_i. \tag{3.1}$$

This transmission satisfies all the clients indexed by $j \in [n]$ such that $R_j \cap I \neq \emptyset$, since I is an independent set, which means $|R_j \cap I| = 1$ if $R_j \cap I \neq \emptyset$. Thus the only unsatisfied clients are the ones indexed by $\{j : R_j \cap I = \emptyset\} = \{j \in [n] : R_j \subseteq \mathcal{V} \setminus I\}$. These $\{R_j : \forall j$ such that $R_j \cap I = \emptyset\}$ are precisely the edges of $\mathcal{H}[\mathcal{V} \setminus I]$. This requires $\beta_q(\mathcal{H}[\mathcal{V} \setminus I])$ transmissions. Hence we get the RHS of the lemma. □

Observation 1. *We observe that the degree of the vertices in $\mathcal{V} \setminus I$, (in the above lemma) are adjacent to at least one vertex in the set I and will decrease by at least one in the induced hypergraph $\mathcal{H}[\mathcal{V} \setminus I]$ after we send the broadcast message x.*

We now have Theorem 1 which shows the main achievability scheme and the upper bound of this work.

Theorem 1. $\beta_q((H)) \leq \Delta(\mathcal{H})$

Proof. Induction base case: For $\Delta(\mathcal{H}) = 1$, consider the following achievable scheme for $\Delta(\mathcal{H}) = 1$. If a hypergraph $\mathcal{H}$ has $\Delta(\mathcal{H}) = 1$, it implies that its vertices either have a degree equal to 0 (i.e., not part of request-set of any client) or equal to 1. Without loss of generality, we can assume that only vertices with degree 1 are in $\mathcal{H}$, as this does not affect the calculation of $\beta_q(\mathcal{H})$. The edges $\mathcal{E}(\mathcal{H})$ where $|\mathcal{E}(\mathcal{H})| = n$ in such a hypergraph are such that for any two edges R_i, R_j, $i, j \in [n]$ and $i \neq j$, $R_i \cap R_j = \emptyset$. Thus, we can find a maximal independent set $I = \{v_1, v_2, ...v_n\}$, where $v_i \in R_i$ for $i \in [n]$. Now, from the definition of a maximal independent set, the server can make one broadcast transmission given by $\sum_{i \in I} b_{v_i}$. Each client i can decode the message $b_{v_i} \in R_i$ from the above transmission, as all other messages in the sum are in its side-information set. Thus, $\beta_q(\mathcal{H}) \leq 1$.

Induction assumption: For any integer $\Delta > 1$, assume that for any hypergraph $\mathcal{H}$ with maximum degree $\Delta - 1$, it holds that $\beta(\mathcal{H}) \leq \Delta - 1$. We will now show that the claim holds for any $\mathcal{H}$ with $\Delta(\mathcal{H}) = \Delta$.

Denote $\mathcal{S} = \{v : \Delta_v = \Delta\}$. We take a maximal (need not be maximum) independent set of this set $\mathcal{S}$, and call it I_Δ. If $v_i, v_j \in \mathcal{S}$ and both belong to any hyperedge $v_i, v_j \in e$ where $e \in \mathcal{E}(\mathcal{H})$, then $\mathcal{S}$ is not a maximal independent set. Now, we partition the vertices $\mathcal{V} \setminus I_\Delta$ into two groups, $\mathcal{S} \setminus I_\Delta$ and $\mathcal{V} \setminus \mathcal{S}$. Observe that the following are true in $\mathcal{H}[\mathcal{V} \setminus I_\Delta]$.

- We claim that the degree of any vertex in $\mathcal{S} \setminus I_\Delta$ is at most $\Delta - 1$ in $\mathcal{H}[\mathcal{V} \setminus I_\Delta]$. To see this, we note the following. Firstly, any vertex in $\mathcal{S} \setminus I_\Delta$ is adjacent to at least one vertex in I_Δ (by the maximality of I_Δ). Hence the degree of this vertex will reduce by at least 1, in $\mathcal{H}[\mathcal{V} \setminus I_\Delta]$ from observation 1 . Thus, this claim follows.

- The degree of all vertices in $\mathcal{V} \setminus \mathcal{S}$ is at most $\Delta - 1$, as it was so in $\mathcal{H}$ itself, by choice of $\mathcal{S}$.

Therefore we can say that the maximum degree of the hypergraph $\mathcal{H}[\mathcal{V} \setminus I_\Delta]$ is at most $\Delta - 1$ after this transmission. According to the induction assumption, we have that $\beta_q(\mathcal{H}[\mathcal{V} \setminus I_\Delta]) \leq \Delta - 1$. As I_Δ is by definition an independent set of $\mathcal{H}$, we can apply lemma1 to this. This gives $\beta_q(\mathcal{H}) \leq 1 + \beta_q(\mathcal{H}[\mathcal{V} \setminus I_\Delta]) \leq 1 + (\Delta - 1) = \Delta$. This completes the proof. □

This bound is obtained through the constructive Algorithm 3.1.1 below that establishes an upper bound on the number of transmissions in a hypergraph representation of a problem. By demonstrating the existence of such a transmission scheme and providing explicit steps on how to carry it out, the constructive proof establishes the upper bound on the number of transmissions. It shows that, with this

particular approach, the number of transmissions required to solve the problem can be bounded by the maximum vertex degree $\Delta(\mathcal{H})$ of the hypergraph.

3.1.1 Achievable Scheme for PICOD

Algorithm 1: Algorithm to construct a PICOD code of length at most $\Delta(\mathcal{H})$

Result: The code of length at most $\Delta(\mathcal{H})$

1 initialization $\mathcal{H}' = \mathcal{H}$, $c = 1$
2 **for** $\delta = \Delta(\mathcal{H}) : 1$ **do**
3 *Transmit=FALSE*
4 **for** $v \in \mathcal{V}(\mathcal{H}')$ **do**
5 **if** $\Delta_v = \delta$ **then**
6 *Transmit=TRUE*
7 $\mathcal{C}(v) = color_c$
8 $\mathcal{V}(\mathcal{H}') \leftarrow \mathcal{V}(\mathcal{H}') \backslash v$
9 **for** $R \in \mathcal{E}(H')$ **do**
10 **if** $v \in R$ **then**
11 **for** $j \in R \setminus v$ **do**
12 $\Delta_j = \Delta_j - 1$
13 **end**
14 $\mathcal{E}(H') \leftarrow \mathcal{E}(H') \setminus R$
15 **end**
16 **end**
17 **end**
18 **end**
19 **if** *Transmit=TRUE* **then**
20 server transmits $\sum_{v \in \mathcal{V}(H)\ : C(v) = color_c} b_v$
21 $c = c + 1$.
22 **end**
23 **if** $\mathcal{E}(H') = 0$ **then**
24 break
25 **end**
26 **end**

We illustrate the above algorithm with the example 2 where hypergraph $\mathcal{H}(V, E)$ has $V = \{1, 2, 3, 4, 5, 6, 7\}$ and $E = \{\{1, 2, 3\}, \{2, 4, 5\}, \{1, 3, 6\}, \{2, 6\}, \{5, 6, 7\}\}$. In line 5, $\delta = 3$ and $v = 2$ (because the degree of V_2 is 3). We assign it $color_1$ in line 7. In line 8 we remove it from the hypergraph vertex set. Lines 9 through 16 pick those hyperedges which have been satisfied and remove them from the hyperedge set.

Also, the degrees of other vertices in these selected hyperedges get decremented. We can see that E_1, E_2 and E_4 contain v_2, hence these edges are removed from the hyperedge set and v_2 is removed from the vertex set. Degrees of vertices v_1, v_3, v_4, v_5 and v_6 get decremented by 1. Since after this step, there are no more vertices with degree 3, hence we break out of the for loop on line 4. Line 19 broadcasts the sum of all messages which have $color_1$. The coded message broadcasted is b_2. The hypergraph $\mathcal{H}'$ gets updated to $\mathcal{H}'(V, E)$ where $V = \{1, 3, 4, 5, 6, 7\}$ and $E = \{\{1, 3, 6\}, \{5, 6, 7\}\}$. Line 23 evaluates to $True$, so we go for another iteration of the for loop on line 2. On line 5, we see that only v_6 has degree 2, so we pick this. It gets removed from the vertex set and both the remaining hyperedges are removed from the hyperedge set. On line 19, we transmit b_6. This time line 23 evaluates to false and we break from the outer for loop on line 2. We have solved the PICOD problem with the coding scheme $\{b_2, b_6\}$. The algorithm uses a vertex-colouring approach and since it terminates in at most $\Delta(\mathcal{H})$ iterations, it shows the existence of an achievable scheme. We also discuss the time complexity of this algorithm, which turns out to be polynomial in the system parameters. Further, we compare our algorithm with existing deterministic algorithms available in the literature. In the following subsection, we give justifications for the steps in the algorithm and verify its correctness.

3.1.2 Correctness of the algorithm

Algorithm 3.1.1 makes a copy of the original hypergraph $\mathcal{H}$ and works on a copied hypergraph $\mathcal{H}'$, initialized with the given input PICOD hypergraph $\mathcal{H}$ (line 1). The algorithm considers (in line 1) a set of $\Delta(\mathcal{H})$ colours denoted by $\{\text{color-}1, \ldots, \text{color-}\Delta(\mathcal{H})\}$. In line 1, the colour's index (denoted by the variable c) is initialized to 1. The loop from lines $4-18$ of Algorithm 3.1.1 finds a maximal independent set of vertices greedily, and those vertices must have degree δ, where δ starts at Δ and is decremented by 1 after each iteration. The existence of such an independent set is kept track of by the variable $Empty$, which is set as $TRUE$ initially (in line 3) and updated to $FALSE$ (in line 6) if at least one vertex with degree δ is found (and thus included in the independent set) in $\mathcal{H}'$. In line 7, the colour c is assigned to vertices. The color of vertex v is shown by a function $C(v)$. The graph $\mathcal{H}'$ is updated (in lines $9-16$) in the following way after adding any vertex v to the independent set.

- Pick vertex v and remove it from the vertex set $\mathcal{V}(\mathcal{H}')$ of $\mathcal{H}'$ (line 8).
- Degrees of the vertices present in any edge containing v are then decreased by one (lines $10-14$).
- All the edges of $\mathcal{H}'$ that include vertex v are removed from $\mathcal{E}(\mathcal{H}')$ (lines 14) (as these will be satisfied by the transmission in line 19 as argued below).

When we choose another vertex (line 4) to be given colour-c, it cannot be adjacent to any vertex v with the same colour (as all edges containing v would have been removed). Thus, these steps construct an independent set of vertices with degree δ in $\mathcal{H}'$. This ensures that the vertices form an independent set, and since we iterate over all vertices in $\mathcal{V}(\mathcal{H})$, the independent set is maximal. We then sum up the messages corresponding to the vertices in the independent set which has colour c, and transmit it (Line

19). Note that this transmission satisfies any client who requests any of these messages that are present in the maximal set. Since each of these clients is missing precisely one message in the independent set constructed, they all get satisfied. Finally, the colour index c is updated (line 21) to assign colours to the vertices with the next δ value.

The entire loop is iterated, with the updated graph $\mathcal{H}'$, until $\mathcal{E}(\mathcal{H}')$ becomes empty (emptiness condition checked in lines 23-25), which must eventually happen, as $\mathcal{H}$ has finitely number of edges. This also means that each edge $R \in \mathcal{E}(\mathcal{H})$ is removed at some point (Line 14), which means some vertex in R is picked as part of some independent set constructed by the algorithm. By the transmission in line 19 corresponding to this independent set, the client R is thus satisfied. Thus, every client is satisfied by Algorithm 3.1.1. Finally, the algorithm stops when $\mathcal{E}(\mathcal{H}')$ is empty (line 23). Observe that the final value of c is the number of transmissions. Since $c \leq \Delta(\mathcal{H})$, Algorithm 3.1.1 provides an achievable scheme of length at most $\Delta(\mathcal{H})$, thus giving a constructive proof for Theorem 3.1.1.

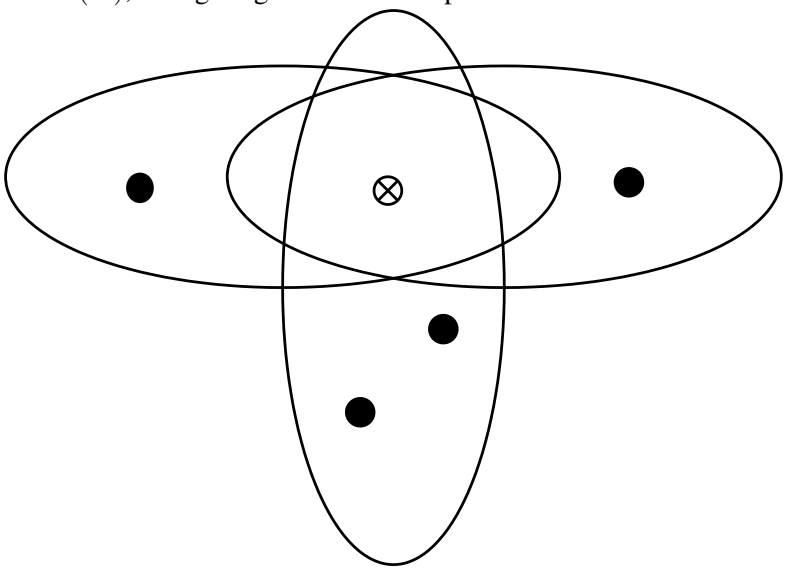

Figure 3.1: The figure helps us understand lines 8 through 16 in the Algorithm 3.1.1. This gives us a better idea of how the hypergraph gets updated when a single vertex (the one with an X in the middle) is picked.

The vertex with an 'X' in the centre corresponds to those vertices v that have been chosen in the 8^{th} step of the algorithm. The hyperedges that contain this vertex are removed from the edge set in the 14^{th} step, because the client it represents has been satisfied by the message represented by v. Also the other vertices in this figure have their degrees decremented by 1 in the 12^{th} step, for every hyperedge removed.

3.1.3 Complexity of the algorithm

For a hypergraph $\mathcal{H}$ with m vertices and n edges, it can be observed from the way the loops are structured in of our algorithm:

- There are at most $\Delta(\mathcal{H})$ values δ can take, so $\mathcal{O}(\Delta(\mathcal{H}))$ iterations are needed for the outermost for loop.
- Similarly, the second for loop iterates for $\mathcal{O}(m)$ rounds.
- The third for loop iterates for at most $\mathcal{O}(n)$ rounds.
- And for the innermost for loop the size of each edge can at most be m, so we need $\mathcal{O}(m)$ rounds.

Multiplying these gives us the runtime complexity to be $\mathcal{O}(\Delta(\mathcal{H})m^2n)$.

3.1.4 Comparison with Existing Algorithms

The algorithm provided employs a vertex-coloring approach to achieve a transmission scheme in the PICOD problem. Similar techniques have been used previously to solve the PICOD problem. Hence it's worthwhile to look at how this achievable scheme performs in comparison with the previously obtained schemes. Hence comparisons with existing deterministic algorithms are also discussed, highlighting the algorithm's potential effectiveness and efficiency.

1. Conflict Free Coloring of Hypergraph [6]

   The algorithm described bears a resemblance to the concept of the conflict free colouring of hypergraphs. Conflict-free colouring aims to assign colours to the vertices of a hypergraph in such a way that no hyperedge contains two vertices of the same colour. This ensures that each hyperedge has at least one vertex with a unique colour, allowing for conflict-free decoding.

   In the provided algorithm, the process of assigning colours to the vertices can be seen as a form of conflict-free colouring. The algorithm iteratively selects vertices with the highest degree (line 3) and assigns them a color (line 6). This ensures that vertices within the same hyperedge do not share the same color, thereby avoiding conflicts during decoding.

   Additionally, the algorithm updates the hypergraph by removing selected vertices and adjusting the degrees of remaining vertices (lines 8-15). This step is akin to the process of modifying the hypergraph based on the colouring in conflict-free colouring, where hyperedges are adjusted to avoid conflicts. While the algorithm does not explicitly use the term "conflict-free colouring," it shares similarities in terms of the goal of assigning colours to vertices in a way that enables conflict-free decoding, albeit in the context of achieving an optimal transmission scheme.

2. Greedy Covering Algorithm (GRCOV) [5]

   The provided algorithm shares similarities with the greedy cover approach used in hypergraph theory. Greedy cover algorithms aim to find a minimal set of hyperedges that covers all the vertices in a hypergraph. The algorithm prioritizes selecting hyperedges that cover the largest number of uncovered vertices at each step.

In the Algorithm 3.1.1 the process of selecting vertices with the highest degree and assigning them a colour can be seen as a form of greedy selection. By focusing on vertices with the highest degree, the algorithm aims to cover a significant portion of the hypergraph's vertices in each iteration.

Similarly, the algorithm updates the hypergraph by removing selected vertices and adjusting the degrees of remaining vertices. This step is akin to the process of removing the covered vertices and updating the degrees of the remaining vertices in greedy cover algorithms.

While the specific terminology of greedy cover is not explicitly used in the algorithm, its underlying principles align with the idea of selecting and covering vertices in a way that minimizes the number of remaining uncovered vertices, leading to an efficient transmission scheme.

3. Edge covering of Hypergraphs [5]

   The provided algorithm bears similarities to the concept of edge covering number in hypergraph theory. The edge covering number represents the minimum number of hyperedges needed to cover all the vertices in a hypergraph. In the described algorithm, the process of selecting vertices with the highest degree and assigning them a color is reminiscent of finding a set of hyperedges that covers the vertices. By focusing on vertices with the highest degree, the algorithm aims to cover as many vertices as possible with each selection. Additionally, the algorithm updates the hypergraph by removing selected vertices and adjusting the degrees of the remaining vertices. This step can be viewed as removing the covered vertices and updating the degrees of the remaining vertices in the context of edge covering.

   In the original BinGreedy algorithm, messages are divided into logarithmic groups based on their effective degrees. This logarithmic grouping ensures a balanced distribution of messages across different levels of effectiveness. On the other hand, this algorithm divides messages into $\Delta(\mathcal{H})$ groups, with each group representing a specific effective degree, as shown by the loop that iterates from 1 through $\Delta(\mathcal{H})$. Both algorithms aim to maximize the number of satisfied clients by assigning coding sub-vectors strategically during the transmission phase. However, the specific grouping strategy in this algorithm emphasizes the fine-grained categorization of messages based on their individual effectiveness, and the coding vectors are now being replaced by scalars 0 and 1. While the original BinGreedy algorithm achieves a performance guarantee of $O(\log^2(n))$ in terms of code length, this modified algorithm offers a different performance as we are able to satisfy all clients in a single transmission instead of a fraction of them like in the previous case. Simulations were run, and a similar performance was seen especially for low delta values. Overall, this slightly different approach to grouping messages based on their effective degrees guarantees a different bound O(delta). we increase the number of groups in order to decrease the number of transmissions needed per group

4. Relationship to BinGreedy algorithm [12]

In the original BinGreedy algorithm, messages are divided into logarithmic groups based on their effective degrees. This logarithmic grouping ensures a balanced distribution of messages across different levels of effectiveness. On the other hand, this algorithm divides messages into $\Delta(\mathcal{H})$ groups, with each group representing a specific effective degree (effective degree as defined in [12]), as shown in line 2 of Algorithm 3.1.1 which iterates from $\Delta(\mathcal{H})$ through 1. Both algorithms aim to maximize the number of satisfied clients by assigning coding sub-vectors strategically during the transmission phase. However, the specific grouping strategy in this algorithm emphasizes the fine-grained categorization of messages based on their individual effectiveness, and the coding vectors are now being replaced by scalars 0 and 1. While the BinGreedy algorithm achieves a performance guarantee of $O(\log^2(n))$ in terms of code length, this modified algorithm offers a different performance as we can satisfy all clients in a single transmission per group, instead of a fraction of them per group like in the case of BinGreedy. Simulations were run, and a similar performance was seen especially for low delta values. Overall, this slightly different approach to grouping messages based on their effective degrees guarantees a different bound $O(\Delta)$. One way to see this different performance guarantee is that we increase the number of groups to decrease the number of transmissions needed per group: $\log(n) \times \log(n)$ now becomes $\Delta \times 1$.

3.2 Lower Bound

In the context of hypergraph-based systems, a converse, or lower bound, refers to the fundamental lower limits on the number of transmissions needed to solve a problem. The concept of a "nested collection" plays a crucial role in obtaining such lower bounds. By identifying a substructure within the hypergraph known as a nested collection of hyperedges, we can derive a lower bound for the system based on its nesting length. This lower bound provides insights into the minimum number of transmissions required, helping us understand the inherent complexity of the problem

3.2.1 Nested Collection

A nested collection is a subset/collection of hyperedges of a hypergraph that has a specific property that constrains the hyperedges to look like ranges in a segment tree (a segment tree is a data structure). Hyperedges can be denoted as lines (ranges on a number line as shown in figure 3.2) and vertices are points (integers on the number line).

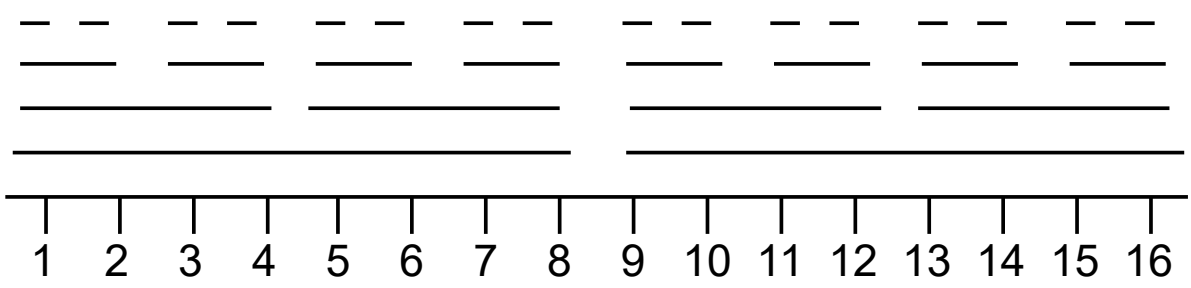

Figure 3.2: Hyperedges of a nested collection: where a line denotes a hyperedge and the points on the x-axis that lie within the range of the line are vertices that are inside the hyperedge.

Definition: For a hypergraph $\mathcal{H}(\mathcal{V}, \mathcal{E})$, any subset $\mathcal{L} \subseteq \mathcal{E}$ that satisfies the following two properties is considered a nested collection of hyperedges of hypergraph $\mathcal{H}$:

1. The number of hyperedges in the subset must be one less than a power of 2, i.e $|\mathcal{L}| = 2^i - 1$ where $i \in \mathcal{Z}^+$.
2. If these $|\mathcal{L}|$ hyperedges can be arranged in a balanced binary tree structure where the children nodes ($\mathcal{L}_{left}$ and $\mathcal{L}_{right}$) of any node $\mathcal{L}_{parent}$ satisfy:
   (a) $\mathcal{L}_{left} \bigcup \mathcal{L}_{right} \subseteq \mathcal{L}_{parent}$.
   (b) $\mathcal{L}_{left} \bigcap \mathcal{L}_{right} = \phi$.
3. Every hyperedge in $\mathcal{L}$ is used in the above construction.

The *line diagram* above satisfies these constraints because:

1. Number of lines = $15 = 2^4 - 1$.
2. Consider the tree where the root is at the bottom and the leaves are at the top, then:
   (a) Union of adjacent lines ¡= parent line(bottom).
   (b) Adjacent lines don't intersect.
3. Since every edge belongs to $\mathcal{L}$, this condition is also satisfied.

3.2.2 Converse Bound for $\beta_q(\mathcal{H})$

Lemma 3 in [8] shows a way of obtaining a lower bound on a pliable index coding problem. It depends on Lemmas 1 and 2 from the same paper. Lemma 1 states $\beta_q(\mathcal{P}_{m,\mathbb{U}}) = \min_D \beta_q(\mathcal{P}_{m,\mathbb{U},D})$, where $\mathcal{P}_{m,\mathbb{U}}$ is a pliable index problem and $\mathcal{P}_{m,\mathbb{U},D}$ is the index coding problem with a decoding choice D. Also m is the number of messages and $\mathbb{U}$ is the set of receivers. This holds because an index code

that solves the index coding problem $\mathcal{P}_{m,\mathbb{U},D}$ will also be a valid pliable index code for $\mathcal{P}_{m,\mathbb{U}}$. Iterating over all decoding choices and picking the minimum value of $\beta_q(\mathcal{P}_{m,\mathbb{U},D})$ is same as solving the PICOD problem. Lemma 2 considers the bipartite representations of an index coding denoted by G. An acyclic subgraph G' is obtained from G by performing a set of pruning operations on G as described in the paper. The paper claims $\beta_q(\mathcal{H}) \geq m(G')$. Hence Lemma 1 relates the PICOD problem with the index coding problem and Lemma 2 obtains the lower bound of code length of the index coding problem. Lemma 3 now combines these two Lemmas and obtains a relation for the lower bound of the code length of a PICOD problem. Combining the two lemmas, we get:

$$\beta_q(\mathcal{P}_{m,\mathbb{U}}) \geq \min_D m(G'_D)$$

Now we shall use the same reasoning on the PICOD problem described by figure 3.2.

Given an index coding problem, we first construct a bipartite graph as follows:

1. The two groups of vertices are arranged in two columns, the left column represents receivers R_i for $i \in [n]$ and the right. Columns have the messages b_{d_i} for $i \in [m]$.

2. A solid arrow from the message vertex(right) to the receiver vertex(left) only if it is demanded by it.

3. A dotted arrow from a receiver vertex if it has a particular message vertex in its side information.

We now show that for a nested collection, any decoding choice will result in an acyclic graph of $\log_2(|\mathcal{L}| + 1)$. We show this by constructing the acyclic subgraph one level per step and show that this process can go on for exactly $\log_2(|\mathcal{L}| + 1)$ steps. Since the nested collection has a recursive structure, an iterative method is a suitable way to prove this bound.

Base step ($i = 1$): pick the hyperedge with the largest number of vertices (R_1) and its demanded message (b_{d_1}), and put a solid arrow from right to left. repeated step ($i = 2$ onwards) : pick the hyperedge with the largest number of vertices (R_i) such that $R_i \subset R_{i-1}$ and $b_{d_i} \notin R_i$ and its demanded message (b_{d_1}), and put a solid arrow from right to left. Also, put dotted arrows from R_i to all $b_{d_{[1:i-1]}}$ (messages above this level).

The nature of the nested collection guarantees that we'll be able to find such messages and receivers (hyperedges) so that we get a total of $\log_2(|\mathcal{L}| + 1)$ levels and the bipartite graph obtained is acyclic. Hence by using lemma 3 from [9], we obtain a minimum of $\log_2(|\mathcal{L}| + 1)$ transmissions.

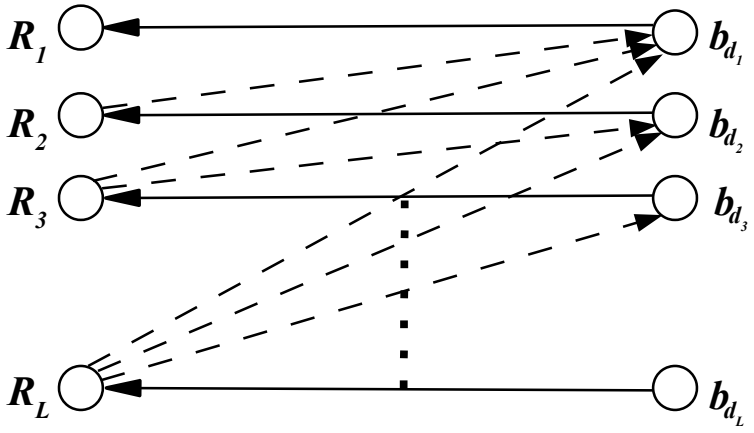

Figure 3.3: Acyclic induced graph of the bipartite graph representation of the hypergraph.

To understand the tightness of this lower bound, we note that in certain classes of problems, $\eta(\mathcal{H}) = \Delta(\mathcal{H})$. In such hypergraphs, $\beta_q(\mathcal{H}) = \Delta(\mathcal{H})$ for every prime power q, and hence $\beta_q(\mathcal{H}) = \eta(\mathcal{H})$. thus proving that our converse bound is tight. and a class of hypergraphs for which this is true is when $\eta(\mathcal{H}) = \Delta(\mathcal{H})$. For example, the hypergraph $\mathcal{H}(V, E)$ falls within this class, where $V = \{1, 2, 3\}$ and $E = \{\{1\}, \{2, 3\}, \{1, 2, 3\}\}$ In this case, $\eta(\mathcal{H}) = \Delta(\mathcal{H}) = 2$.

3.3 Algorithm for Finding $\eta(\mathcal{H})$

In this section, we give an algorithm for calculating the value of $\eta(\mathcal{H})$, given the hypergraph $\mathcal{H}$. Now, if we know a collection of a hypergraph with the largest length, the nesting number $\eta(\mathcal{H})$ is simply its length. Hence, we focus on constructing a nested collection.Further, we need the length of this nested collection to be the largest possible. The following algorithm helps us find such a collection.

Algorithm 2: Algorithm for finding $\eta(\mathcal{H})$

initialization;
$\mathcal{E}' = \mathcal{E}$; $S = \mathcal{E}'$
while $\mathcal{E}' \neq \phi$ **do**
    for $e \in \mathcal{E}'$ **do**
        if $e_1 \bigcup e_2 \subseteq e$ *and* $e_1 \bigcap e_2 = \phi$ *where* $e_1, e_2 \in \mathcal{E}'$ **then**
            $S \leftarrow S \setminus e$
        end
    end
    $\mathcal{E}' = \mathcal{E}' \setminus S$
end

We briefly describe the above algorithm. At every iteration, we remove hyperedges that cannot be a parent node to any pair of non-intersecting hyperedges in the corresponding binary tree. we keep iterating until the edge set is empty.

Theorem 2. *The number of iterations needed to complete Algorithm 3.3 is equal to the nesting number $\eta(\mathcal{H})$ of the hypergraph $\mathcal{H}$. Hence we use this algorithm to find the nesting number value.*

Proof. Just for this proof, let's consider the true value of the nesting number is $\eta'(\mathcal{H})$. If the algorithm takes $\eta(\mathcal{H})$ steps, then there exists a nested collection with a nesting number of at least $\eta(\mathcal{H})$. We go through the iterations of the above algorithm in the reverse order, hence groups of hypergraphs appear after each iteration. In this new ordering, the first iteration will generate a random set of hyperedges, we mark any one of these hyperedges arbitrarily. For the following steps, we mark any arbitrary pair of non-intersecting hyperedges that are subsets of a node marked in the previous step. We repeat this process until we reach the first step in the previous ordering. The marked hyperedges constitute the nested collection. since we used every iteration to create at most one new level, the maximum possible nesting number $\eta'(\mathcal{H}) \leq \eta(\mathcal{H})$.

If there exists a nested collection of $\eta'(\mathcal{H})$, the algorithm will take at least $\eta'(\mathcal{H})$ rounds. This is because in the hyperedge set $\mathcal{E} = \mathcal{L}$, where $\mathcal{L}$ is the nested collection of hyperedges of length $\eta'(\mathcal{H})$, it is easy to see that it would take at least $\eta'(\mathcal{H})$ rounds. Addition of more hyperedges to $\mathcal{E}$ (thus making $\mathcal{L} \subseteq \mathcal{E}$), cannot decrease the number of steps the algorithm takes to terminate. hence $\eta'(\mathcal{H}) \geq \eta(\mathcal{H})$. Since both inequalities are true, this implies $\eta'(\mathcal{H}) = \eta(\mathcal{H})$. □

Chapter 4

Bounds using Other Connectivity Parameters

We have seen a few upper bounds so far, the main one being the $\Delta(\mathcal{H})$ bound. We shall now look at another upper bound that is of interest. The edge overlap parameter Γ is a measure of the density of the hypergraph. It is the maximum number of hyperedges intersecting with any one hyperedge. Intuitively, we can see that a low Γ indicates that the hypergraph is sparse and the PICOD problem described by it can be solved with a shorter code. It is also worth noting that the Γ parameter is linked to the Δ parameter because a high Δ value implies that many hyperedges are intersecting at a single vertex, and thus also with each other. This means that the Γ parameter will also have to be high. We shall now look at why Γ is a bound for β_q.

Theorem 3. *Let Γ represent the edge overlap parameter of a given PICOD hypergraph $\mathcal{H}$ i.e. $\Gamma \triangleq \max_{\forall e} \Gamma_e$, where Γ_e is the edge overlap parameter for the hyperedge e. $\Gamma_e = \sum_{\forall i \in E, i \neq e} f(e,i)$. Here $f(e,i)$ is the function that indicates if edge e and i intersect. Formally, $f(e,i) = \begin{cases} 1 & \text{if } e \bigcap i \neq \phi \\ 0 & \text{otherwise} \end{cases}$. The optimal number of transmissions for a PICOD problem is upper bounded by Γ, i.e., $\beta_q(\mathcal{H}) \leq \Gamma$, for any prime power q.*

Proof. We can achieve this by constructing a dual graph of our PICOD problem's hypergraph. In this dual graph, vertices correspond to hyperedges and an edge between two vertices exists if the corresponding hyperedges intersect. It is easy to see that the maximum vertex degree of our dual graph is equal to the edge overlap parameter of the hypergraph and is Γ. We can partition the vertex set of the dual graph into utmost Γ subsets such that no subset has a pair of adjacent vertices. The hypergraph induced by these subsets of the dual graph have no intersecting hyperedges and hence have $\Gamma = 0$. Thus, we have found a way to split the original hypergraph into utmost Γ sub-hypergraphs with each sub-hypergraph having $\Gamma = 0$. In the following subsection, we shall see that PICOD problems with $\Gamma = 0$ hypergraphs can be satisfied in 1 transmission. Hence splitting the hypergraph in this way allows us to solve each

partition in 1 transmission, and since we have Γ partitions, we can satisfy our PICOD problem in at most Γ transmissions. $\square$

Suppose we prefer a matrix representation instead of a hypergraph representation while dealing with the Γ parameter. In that case, we need a scheme for populating this matrix that we shall call an indicator matrix. Let Γ_{ij} denote an indicator that indicates whether hyperedges e_i and e_j intersect with each other or not. Let Γ_i represent the number of other edges intersecting an edge e_i, can be defined as : $\Gamma_i = \sum_{j \in [n] \setminus \{i\}} \Gamma_{ij}$. Then *maximum overlap parameter* Γ is defined as, $\Gamma = \max_{i \in [n]} \Gamma_i$. Now, let $\bar{\Gamma}$ denote the edge overlap indicator matrix. It is a symmetric matrix with principal diagonal elements being all zeros. In the figure 4.1.

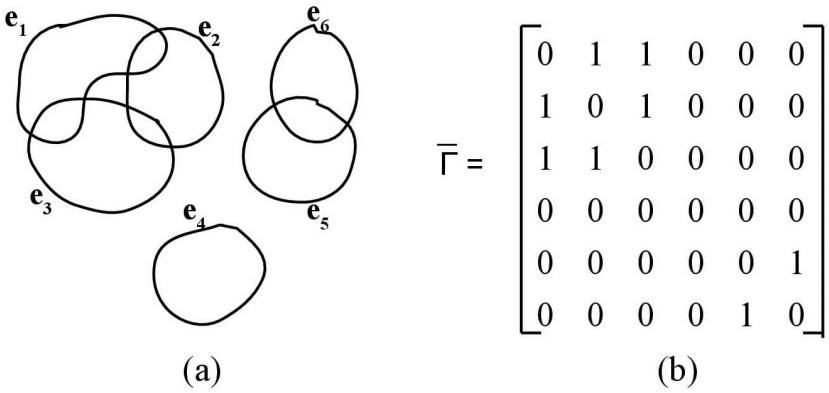

Figure 4.1: (a) A hypergraph with 6 edges (b) Corresponding edge overlap indicator matrix $\bar{\Gamma}$.

4.1 Complete characterisation for small Γ hypergraphs

Γ, the edge overlap parameter, is the maximum number of hyperedges intersecting any particular hyperedge. This is an indication of how "dense" the hypergraph is. As mentioned in the above theorem, it also bounds the optimal number of transmissions needed to satisfy a PICOD problem. Another observation that can be made is that if a hypergraph has a maximum degree Δ, then its Γ parameter must be at least $\Delta - 1$, because if we consider a hyperedge incident on a vertex with degree Δ, that hyperedge will be intersecting with $\Delta - 1$ other hyperedges which are also incident on this vertex. Dealing with small values of Γ allows for a comprehensive analysis of the various classes of hypergraphs that can be constructed, which shall be done in this section. We'll also see an algorithm that can be used in the $\Gamma = 2$ scenario.

4.1.1 PICOD for hypergraphs with $\Gamma = 0, 1$

We now consider the case of $\Gamma = 0$. $\bar{\Gamma}$ is a *null matrix* and the number of transmissions required to satisfy all the clients is just *one* which takes up the form $\sum_{i \in [n]} b_i$. When $\Gamma = 1$, every row (or every column) of $\bar{\Gamma}$ matrix satisfy the following condition: $\sum_{i \in [n]} \Gamma_{ij} \leq 1$. This can be equivalently written as,

$\sum_{j\geq i}\Gamma_{ij}+\sum_{j<i}\Gamma_{ij}\leq 1$. Further, since $\bar{\Gamma}$ matrix is symmetric ($\Gamma_{ij}=\Gamma_{ji}\forall(i,j)$) the above condition can be written only in terms of elements in upper triangular matrix as follows: $\sum_{j\geq i}\Gamma_{ij}+\sum_{j<i}\Gamma_{ji}\leq 1$. The algorithm for this case is straightforward. We recall the idea of connected components from 1. In case of many connected components, we find transmission schemes for each of them independently. The number of transmissions to satisfy any collection of connected components is the same as the maximum number of transmissions needed to satisfy all hyperedges of any one of those connected components. Hence if $\Gamma=1$, then the only connected component possible is 2 hyperedges intersecting (like in a Venn diagram with 2 circles). We just pick any arbitrary message in this intersection b where $b\in c_i\bigcap c_j$. In order to solve for the entire collection of connected components, We just sum up the coded messages obtained at each connected component. For example if connected components $C=\{C_i,C_2\}$ produce codes $\{\{m_{11},m_{12}\},\{m_{21}\}\}$. then we sum them up as $\{\{m_{11}+m_{21},m_{12}\}\}$.

4.1.2 PICOD for hypergraphs with $\Gamma=2$

In this case, every row (or every column) of $\bar{\Gamma}$ matrix satisfies the following condition: $\sum_{i\in[n]}\Gamma_{ij}\leq 2$. This can be equivalently written as, $\sum_{j\geq i}\Gamma_{ij}+\sum_{j<i}\Gamma_{ij}\leq 2$. Further, since $\bar{\Gamma}$ matrix is symmetric the above condition can be written only in terms of elements in the upper triangular matrix as follows: $\sum_{j\geq i}\Gamma_{ij}+\sum_{j<i}\Gamma_{ji}\leq 2$. The same idea of connected components applies here as well. since $\Gamma=2$, we can explore all possibilities. They're listed below:

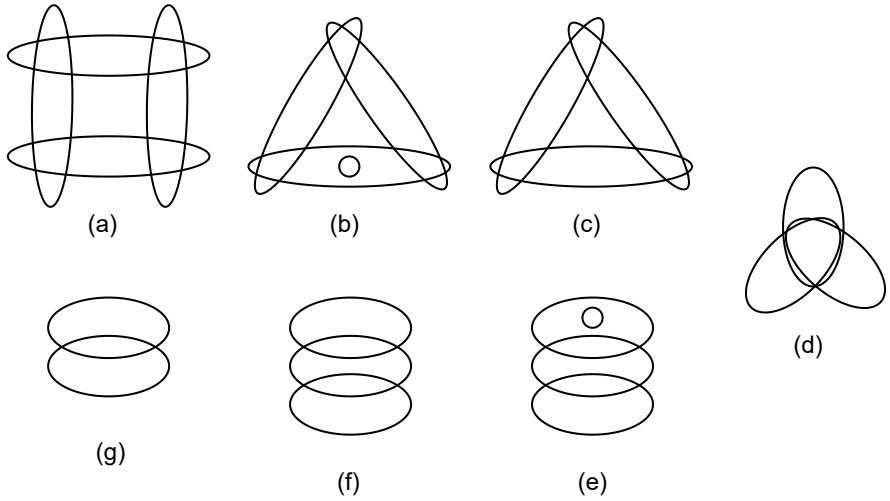

Figure 4.2: The exhaustive set of classes of $\Gamma=2$ hypergraphs.

(a) Even cycle: When an even number of hyperedges intersect in this manner,

(b) Odd cycle with at least one exclusive vertex: When an odd number of hyperedges intersect in this manner, and at least one of these has a vertex that isn't present in any other hyperedge. This is what is referred to as an "exclusive vertex".

(c) Odd cycle without any exclusive vertex: Similar to (b) but without any such "exclusive vertex".

(d) Star: Exactly 3 hyperedges involved in this class. There exists atleast 1 message common to all of them.

(e) Odd chain with at least one exclusive vertex on an odd indexed hyperedge: The following 3 classes are going to be chain structures unlike (a), (b) and (c) which are ring structures. An odd number of hyperedges are present in a configuration, and an exclusive vertex present in an odd indexed hyperedge when the indexing starts at 1 from a hyperedge with $\Gamma = 1$.

(f) Odd chain without any exclusive vertex on an odd indexed hyperedge: Similar to (e), but without any such exclusive vertex.

(g) Even chain: Similar to (a), but the first hyperedge doesn't connect to the last.

It is easy to find schemes for classes (a), (b), (d), (e), (g), which require only 1 transmission. Classes (c) and (f) require a minimum of 2 transmissions.

The algorithm that takes exactly 2 transmissions to satisfy hypergraphs with $\Gamma = 2$ is described below:

Algorithm 3: Achievable scheme for $\Gamma = 2$

Input : Upper triangular matrix of $\bar{\Gamma}$
Output: color1, color2

Initialize: $SAT[1:n]$ = FALSE;
for $i = 1$ *to* $n-1$ **do**
  if $SAT[i] =$ *FALSE* **then**
    for $j = i+1$ *to* n **do**
      if $\Gamma_{ij} = 1$ **then**
        $color_1 = color_1 + b_k$, where $k \in e_i \cap e_j$;
        $SAT[i]$ = TRUE;
        $SAT[j]$ = TRUE;
        break;
      end
    end
  end
  for *i=1:n-1* **do**
    if $SAT[i] =$ *FALSE* **then**
      $color_2 = color_2 + b_k$ where $i \in e_i$
    end
  end
end

4.2 Connectivity Parameters and Hyperedge Sizes

The quantities S_{max} and S_{min} represent the maximum and minimum sizes (i.e. number of vertices) of any hyperedge in hypergraph $\mathcal{H}$. We now derive a few relations between the hyperedge's size and the connectivity parameters. We do this because there may be cases when special restrictions are placed on the sizes of the hyperedges.

Taking any one hyperedge of size S_i into consideration, the maximum number of distinct non-overlapping edges is the size of the power set of $V(H) \setminus V(\mathcal{E}) = 2^{m-S_i} - 1$. If this hyperedge intersects with Γ_i other hyperedges, then $n - 1 - \Gamma_i \leq 2^{m-S_i} - 1$. simplifying, we get $\log_2(n - \Gamma_i) \leq m - S_i$. Since this is true for all S_i, we can replace this with S_{max}. Also, $L.H.S$ is maximized when Γ_i is minimum. hence

$$S_{max} \leq m - \log(n - \Gamma_{min})$$

It was shown in chapter 2 that the PICOD problem can be represented in terms of a bipartite matrix. By using different strategies to count the number of 1's, we can find relations between different connectivity parameters. The maximum number of 1's per column is $\Gamma + 1$. hence an upper bound on the total number of 1's is $m \times (\Gamma + 1)$, and a lower bound is $n \times S_{min}$. Thus, $n \cdot S_{min} \leq \sum_{i=1}^{n} S_i \leq m \cdot (\Gamma + 1)$.

$$\implies S_{min} \leq \frac{m \cdot (\Gamma + 1)}{n}$$

Now we consider only the columns that correspond to messages in the hyperedge with the smallest number of vertices. $\sum_{i=0}^{S_{min}}$ number of 1's in $column_i \geq \Gamma_{min} + S_{min}$. Since the maximum number of 1's in any column is Δ, $R.H.S$ can be replaced with $\Delta \cdot S_{min} \geq \Gamma_{min} + S_{min}$, rearranging terms, we get $\Delta \geq \frac{\Gamma_{min}}{S_{min}} + 1$. hence,

$$\Delta > \frac{\Gamma_{min}}{S_{min}}$$

Thus we have obtained a few relations between the connectivity bounds Γ, Δ and the size of the hyperedges.

Chapter 5

Conclusions

In this work, we consider the pliable index coding (PICOD) problem, represented via a hypergraph $\mathcal{H}$. As our essential contributions, we presented novel upper and lower bounds on $\beta_q(\mathcal{H})$, which is the optimal length of the pliable index code (designed over finite field with q elements) represented by a hypergraph $\mathcal{H}$.

The results of this thesis were presented in Chapter 3 and Chapter 4. In Chapter 3, we presented a novel upper bound for the optimal PICOD length using the parameter $\Delta(\mathcal{H})$ which represents the degree of the hypergraph $\mathcal{H}$. We do this by showing an explicit achievable scheme, via an algorithm. We compared our algorithm with previously proposed algorithms and presented the important distinctions. We also presented a lower bound using a novel parameter of the hypergraph, denoted by $\eta(\mathcal{H})$, which we call the nesting number of the graph. We presented an efficient algorithm, which calculates the value of the nesting number. The complexity of obtaining the coding scheme was also shown to be of polynomial time complexity. Therefore, we have presented both an upper and lower bound for β_q. The tightness of these bounds was also proved for some special classes of hypergraphs.

Chapter 4 deals with another bound, that may be of interest, namely Γ. The small Γ PICOD problems were completely characterized and an algorithm was provided for these small Γ PICOD problems. Results regarding the maximum and minimum request set size of any client (S_{max} and S_{min}) were also explored.

We now have an upper bound for Δ, using Algorithm 3.1.1. This can be extended to the t-requests PICOD scenario also. It is easy to see that a naive way of solving the PICOD-t problem is by repetition of Algorithm 3.1.1 t times, thus giving an upper bound of $O(t\Delta)$. A scope for optimization of Algorithm 3.1.1 can be explored, where we do not group the vertices based on their effective degrees. This does not improve the Δ bound guarantee but could satisfy a larger number of clients per transmission, thus potentially offering a marginal improvement. It is also easy to see that by not grouping (but still sorting by effective degree), we do not cause any inconsistency that affects the correctness of the algorithm.

Simulations can be run for various PICOD and PICOD-t coding schemes as a way of testing performance. For the purpose of simulations, hypergraphs should be constructed randomly, and plots should be made to compare the algorithms like BinGreedy [12], GrCov [5], Algorithm 3.1.1, and a modified

version of it as proposed in the previous paragraph. The code length can be plotted against various parameters like Δ, p (probability a message is requested by a client) and n (number of clients). Finally, we can also explore the effect of sorting by effective degree during each step, while running the GrCov [5] algorithm, since this could give us an improvement.

Publications Related to This Thesis

- Bounding the Optimal Length of Pliable Index Coding via a Hypergraph-based Approach [13].
- Bounding the Optimal Length of Pliable Index Coding via a Hypergraph-based Approach [In preparation for submission to journal].

Bibliography

[1] A. Agarwal and A. Mazumdar. Local partial clique covers for index coding. *arXiv:1603.02366*, pages 1152–1156, 2016.

[2] F. Arbabjolfaei and Y.-H. Kim. Local time sharing for index coding. *IEEE ISIT'14*, pages 1152–1156, 2014.

[3] Z. Bar-Yossef, Y. Birk, T. S. Jayram, and T. Kol. Index coding with side information. *IEEE Transactions on Information Theory*, pages 1479–1494, 2011.

[4] A. Blasiak, R. Kleinberg, and E. Lubetzky. Index coding via linear programming. *ArXiv-CoRR*, page 48, 2011.

[5] S. Brahma, , and C. Fragouli. Pliable index coding. *IEEE transactions*, 61(11):12, 2015.

[6] P. Krishnan, R. Mathew, and S. Kalyanasundaram. Pliable index coding via conflict-free colorings of hypergraphs. *arXiv*, page 39, 2022.

[7] T. Liu and D. Tuninetti. Tight information theoretic converse results for some pliable index coding problems. *IEEE Transactions on Information Theory*, page 16, 2019.

[8] L. Ong, B. N. Vellambi, and J. Kliewer. Optimal-rate characterisation for pliable index coding using absent receivers. *arXiv:cs.IT*, page 6, 2019.

[9] L. Ong, B. N. Vellambi, J. Kliewer, and P. Sadeghi. Improved lower bounds for pliable index coding using absent receivers. *arxiv*, page 7, 2019.

[10] S. Sasi and B. S. Rajan. Code construction for pliable index coding. *ISIT*, page 5, 2019.

[11] K. Shanmugam, A. G. Dimakis, and M. Langberg. Local graph coloring and index coding. *Proceedings of IEEE International Symposium on Information Theory (ISIT)*, pages 1152–1156, 2013.

[12] L. Song, , and C. Fragouli. A polynomial-time algorithm for pliable index coding. *arxiv*, page 48, 2017.

[13] V. Subramanian, P. Krishnan, et al. Bounding the optimal length of pliable index coding via a hypergraph-based approach. In *2022 IEEE Information Theory Workshop (ITW)*, pages 696–701. IEEE, 2022.

[14] C. Thapa, L. Ong, and S. J. Johnson. Interlinked cycles for index coding:generalizing cycles and cliques. *arXiv:1603.00092*, pages 1152–1156, 2016.