On Multi-source Multi-Sink Hyperedge Networks: Enumeration, Rate Region Computation, and Hierarchy

A Thesis

Submitted to the Faculty

of

Drexel University

by

Congduan Li

in partial fulfillment of the

requirements for the degree

of

Doctor of Philosophy

May 2015

Acknowledgments

I would like to express my deepest appreciation to my advisors and committee members, Dr. John MacLaren Walsh, Dr. Steven Weber, Dr. Naga Kandasamy, Dr. Chao Tian and Dr. Babak Hassibi, for their advice, encouragement, enduring patience and constant support. It is an honor for me to work with them. My appreciation also goes to my colleagues in the Adaptive Signal Processing and Information Theory Research Group (ASPITRG) and Model and Analysis of Networks Laboratory (MANL) for their support and collaboration.

I also thank all the other faculty, staffs and students in the Department of Electrical and Computer Engineering at Drexel University who give me a lot of help during my study time at Drexel University.

Last but not least, I would like to thank my wife, my whole family, and my friends for their support and everlasting love.

Contents

List of Tables

List of Figures

Abstract

On Multi-source Multi-Sink Hyperedge Networks: Enumeration, Rate Region Computation, and Hierarchy

CONGDUAN LI

This dissertation utilizes computation tools to solve an important open problem in information theory and network coding, that is, to calculate the rate regions of multi-source multi-sink networks. For enumeration purpose, notions of minima networks and network equivalence are defined. Then efficient enumeration algorithms, based on list of Sperner families and Leiterspiel algorithm, are developed, where the algorithm based on Leiterspiel outer performs the former one. With the enumeration tools, millions of non-isomorphic networks are enumerated and most of them are solved by our computation software. To analyze the huge repository of rate regions, operators that relate networks of different sizes are defined so that more large networks can be solved and forbidden network minors regarding some classes of linear codes can be obtained. This dissertation contains two major chapters, where §2 focuses on multi-level diversity coding systems (MDCS), a special class of multi-source networks, and §3 extends the discussion to the general multi-source multi-sink hyperedge networks with more powerful hierarchy structure developed.

In §2, the rate regions of multilevel diversity coding systems (MDCS), a sub-class of the broader family of multi-source multi-sink networks with special structure, are investigated. After showing how to enumerate all non-isomorphic MDCS instances of a given size, the Shannon outer bound and several achievable inner bounds based on linear codes are given for the rate region of each non-isomorphic instance. For thousands of MDCS instances, the bounds match, and hence exact rate regions are proven. Results gained from these computations are summarized in key statistics involving aspects such as the sufficiency of scalar binary codes, the necessary size of vector binary codes, etc. Also, it is shown how to generate computer aided human readable converse proofs, as well as how to construct the codes for an achievability proof. Based on this large repository of rate regions, a series of results about general MDCS cases that they inspired are introduced and proved. In particular, a series of embedding operations that preserve the property of sufficiency of scalar or vector codes are presented. The utility of these operations is demonstrated by boiling the thousands of MDCS instances for which binary scalar codes are insufficient down to 12 forbidden smallest embedded MDCS instances.

With the exciting rapid computations on MDCS problem, §3 investigates further to the enumeration, rate region computation, and hierarchy of general multi-source multi-sink hyperedge networks under network coding, which includes multiple network models, such as independent distributed storage systems and index coding problems as special cases. A notion of minimal networks and a notion of network equivalence under group action are defined. By harnessing the Leiterspiel algorithm, an efficient enumeration algorithm is presented. Using this algorithm, millions of non-isomorphic networks are obtained. Then by applying computation tools, exact rate regions of most of the obtained networks are calculated, leaving the rest with outer bounds and simple code achieving inner bounds. In order to better understand and analyze the huge repository of rate regions, several embedding (as extensions of embedding operations defined in §2) and combination operations are defined in a manner that rate region of the network after operation can be derived from the rate region of networks involved in the operation. The embedding operations enable us to obtain a list of forbidden network minors for the sufficiency of a class of linear codes. The combination operations enable us to solve large networks that can be obtained by combining some solvable networks. The integration of both embedding and combination operations is able to generate many more network capacity regions than can be obtained through combination operations alone.

Chapter 1: Introduction

This dissertation investigates the use of computer aided tools to calculate rate regions of networks. Millions of non-isomorphic networks are enumerated and solved by our computation software. Operators that relate networks of different sizes are defined so that more large networks can be solved and forbidden network minors regarding some classes of linear codes can be obtained. This dissertation contains two major chapters, where §2 focuses on multi-level diversity coding systems (MDCS) and §3 discusses the general multi-source multi-sink hyperedge networks. We begin with motivation and background.

Many problems involve the calculation of fundamental limits on source rates, and/or channel capacities in network coding. For instance, the satellite communications [3], the distributed storage systems [4, 5], live streaming [6–8], etc.

Yan, Yeung and Zhang gave an implicit characterization for the rate region of network coding problems in terms of region of entropic vectors, which essentially contains valid entropy vectors and will be discussed in §2.3.3. This formulation gives a way to develop computation tools by which a computer can be of help in calculating the rate regions. However, as shown in §2.3.3, the fully characterization of the region of entropic vectors is unknown for more than 4 variables. Thus, for most of the networks under consideration, the exact rate region is still an open question. An alternate solution is to replace the region of entropic vectors with its polyhedral outer and inner bound so that one can develop polyhedral computation tools.

The outer bound we typically use is the Shannon outer bound, which is characterized by fundamental inequalities on Shannon entropy. Non-Shannon type inequalities [9, 10] could be used if necessary. The inner bounds we used are derived from representable matroids [11], as the representations of matroids can be used as linear codes. The outer bound calculation may serve as a converse proof and the inner bound calculation serves as an achievability proof since it gives achieving codes. If the outer and inner bound match, in which case we have both converse and achievability proofs, we obtain the exact rate region. Since the computer aided calculations involve high dimensional polyhedral computation and projection, though we are able to get the results, it is not easy for a

human to determine how we get it. Therefore, we let the computer to automatically generate the human readable proofs by selecting proper inequalities and coefficients used in the calculation, as shown in §2.7.

Having the computation tool, we first tried on a small class of networks, Multi-level Diversity Coding Systems (MDCS) [1, 12]. This model is interesting because it is one of the earliest model inspired the distributed storage system that many researchers are currently working in. In a MDCS instance, there are multiple prioritized sources available to multiple decoders, and the decoders are classified as different levels where a level k decoder requests first k sources. As will be shown in §2, we enumerated all 7050 non-isomorphic MDCS instances with size up to $(3, 4)$ and calculated various bounds on their rate regions, where two networks are isomorphism if one can be obtained by permuting the variables in the other. By comparing the outer and inner bounds, we obtained exact rate region for most of them. For the other ones, we have outer bounds, and simple code, say binary, achievable inner bounds.

Having a repository of rate region for thousands of network, it makes sense to develop some tools to analyze them. Inspired by Robertson and Seymour's graph minor project [13, 14], we also would like to see if for networks, there exist forbidden network minors for the sufficiency of a class of linear codes. We first defined some embedding operations on networks such that if a class of codes does not suffice for a smaller network, we know it will not suffice either for the larger network that has the smaller one embedded in. With these operators, we can obtain the forbidden network minor list regarding the sufficiency of a class of linear codes. This work also inspired our later following work on general networks. As shown in §2.6, for the more than 7000 MDCS instances considered, there are only 12 forbidden network minors for sufficiency of scalar binary codes, and 28 forbidden minors for sufficiency of superposition (source-separate) coding.

The rapid computation of rate regions for MDCS instances motivates us to move on to general networks. We slightly generalize the model in [15] so that the edges in the network can be hyperedges, which happens frequently in real networks. As will be shown in §3.2, the network model presented contains several different types of problems, such as the independent distributed source coding (IDSC) problems and the index coding problems.

Since the network model is not as simple as MDCS, we first define a notion of *minimal* network that does not contain redundancies and discussed their impacts on rate region if we do the reduction, as shown in §3.3. For the enumeration of general hyperedge networks, we have to consider the topologies and more minimality constraints than MDCS. We extended our enumeration based on list of Sperner families [16] for MDCS, which will be presented in §B. However, due to the isomorphism

involved in the enumeration, though partial of isomorphisms can be removed at the beginning, the process becomes difficult for even moderate size networks. We have to develop a new algorithm to enumerate non-isomorphic networks directly instead of including isomorphisms and then removing them.

After defining the network equivalence or isomorphism in terms of group action, the enumeration of non-isomorphic networks becomes obtaining transversals of orbits of all networks with a given size, where each orbit is an equivalence class and its transversal is the representative one. The groups are direct product of symmetry group of sources and symmetry group of edges, since it is not allowed to permute sources with edges. Based on the Leiterspiel algorithm [17], which calculates orbits of subsets incrementally in the size of subsets, we developed a more efficient algorithm (§3.4) to enumerate the non-isomorphic networks directly. Using this new technique, we are able to enumerate millions of non-isomorphic network instances.

With the list of all non-isomorphic minimal networks in hand, we can then utilize our computation tools to calculate their rate regions, as what we did for MDCS, and give the converse and achievability proofs. As shown in §3.5, for the obtained 725556 non-isomorphic small network, we solved most of them by giving exact rate region and gave bounds on the rest. For the unsolved instances, we can further utilize tighter inner bounds, or outer bound, to obtain their exact rate regions.

Again, with such a huge database of network rate regions, we would like to analyze them. We first augment the embedding operators on MDCS to general networks and then obtain forbidden network minors regarding the sufficiency of a class of linear codes. As shown in §3.8, we have obtained a list of forbidden minors for the sufficiency of scalar binary codes as a demonstration.

We also would like to see if the obtained rate regions of small network can help us in solving larger networks which cannot be handled by our computation tools. That is, if there exist some transformations or reductions for the large networks so that we can utilize the obtained results. Similar spirit of transforming networks can be found in [18, 19], where [18] replaces channels with their noiseless upper and lower bounding models to solve the original problem, and [19] transforms a general network into an index coding problem by coding the topology relations of the network into sources and demands in the associated index coding and shows their equivalence.

We define some combination operation on networks so that the combined network can be solved if the smaller networks involved in the combination are solved. Though these combinations are able to generate more solvable networks, we notice that integration of combination and embedding operations will become a quite powerful way to generate much more new networks that cannot be reached by combinations only. The combination operations can be applied infinite times and

arbitrarily large networks can be obtained. To generate a handleable list of networks, a notion of worst case partial closure of networks is defined where the (predicted) networks involved in the combination and embedding operations cannot exceed a certain size. As demonstrated in §3.8, even with very few tiny networks as a seed list, one can generate thousands of new networks that can be solved. These operations build a hierarchy among networks with different sizes and bring in many more interesting problems to further investigate in.

Chapter 2: Multi-level Diversity Coding Systems

2.1 Introduction

For more than a decade, network coding problems have drawn a substantial amount of attention because of the ability of a network to support more traffic when network coding is utilized instead of routing [20]. For a single-source multicast network, the rate region can be characterized by the max-flow min-cut of the network graph. However, the capacity regions of more general multi-source multicast networks and multiple unicast networks under network coding are still open. In [15], an implicit characterization of the achievable rate region for acylic multi-source multicast networks was given in terms of Γ_N^*, the fundamental region of entropic vectors [2].

While it is known that $\bar{\Gamma}_N^*$ is a convex cone, the exact characterization of it is still an open problem for $N \geq 4$. In fact, it has been shown that the problem of determining the capacity regions of all networks under network coding is equivalent to that of determining $\bar{\Gamma}_N^*$ [15, 21]. However, one can use outer or inner bounds on $\bar{\Gamma}_N^*$ to replace it in the rate region expression in [15] in order to obtain outer bounds and inner bounds for the network coding capacity region. When these bounds match for the network in question, the capacity region has been determined exactly. For example, in our previous work [22, 23], we presented algorithms combining inner bounds obtained from representable matroids, together with the Shannon outer bound, to determine coding rate regions for multi-terminal coded networks. A merit of the method of multiple source multicast rate region computation presented there is that the representable matroid inner bound utilized naturally corresponds to the use of linear codes over a specified field size, and these codes can be reconstructed from the rate region [22].

Recently, many researchers in the network coding community have shown interest in distributed storage systems, where network codes are used to store data on disks on different network computers in a manner which enables them to recover the system after some disk or computer failures [24]. After a failure, the recovered disk can either contain exactly the same contents as before, in which case the problem is referred to as an exact repair problem [5, 25], or a possibly different encoding of the data that still enables the computer array to have the same storage and recovery capabilities

as before, which is known as the functional repair problem [4]. Work on these problems has shown that the fundamental design tradeoffs between storage and repair bandwidth are consequences of network coding rate regions [4, 5].

Some of the earliest models that inspired network coding and distributed storage were multi-level diversity coding systems (MDCS) [1, 12]. Despite being some of the oldest models for distributed storage of interest, and some of the simplest, for unsymmetric cases [26, 27], they remain largely unsolved [1]. Similarly, network coding and distributed storage solutions remain highly dependent on symmetries [4, 25, 28] which will in many instances not be present in real life. Since MDCS cases are some of the simplest asymetric networks of interest, it makes sense to study their rate regions first as one moves towards the direction of more general, and more difficult network coding and distributed storage rate region problems.

The concept of MDCS was introduced by Yeung [29], inspired by diversity coding systems [12]. The multiple sources in a MDCS have ordered importance with the most important sources having priority for decoding. The sources are accessible and encoded by multiple encoders. Every decoder has access to a different subset of the encoders and wishes to reproduce the k highest priority sources, where k depends on the decoder. As in most network coding and distributed storage problems, the sources are assumed to be independent. Note that this definition of MDCS is slightly different from that in the later paper [30], where the source variables represent some possibly correlated random variables, there is one decoder for every subset of encoders, and every decoder wishes to reproduce one source. However, the model in this paper can be taken as a special case of [30].

Some small MDCS instances were studied in [1, 31] where the number of encoders and number of sources were up to 3. Additionally, in [1,31], the sufficiency of superposition codes was studied, where superposition coding means encoders can encode sources separately and then simply concatenate the coded messages. It is shown in [1, 31] that for most 2-level and 3-level 3-encoder MDCS instances, superposition coding is sufficient to obtain the same rate region as all possible codes. For the instances where superposition coding does not suffice, linear codes are constructed that achieve the parts of the fundamental rate region that superposition coding cannot achieve.

One special type of MDCS, known as symmetrical MDCS (SMDCS), was studied in [26, 27]. In a SMDCS problem, there are K sources, K encoders and $2^K - 1$ decoders where every decoder has access to one unique non-empty subset of the encoders. All decoders who have access to the same number of encoders will reproduce exactly the same prioritized sources. For instance, all $\binom{|\mathcal{E}|}{l}$ decoders which have access to the size-l subsets of encoders respectively, must be able to reproduce

first l sources, $\mathbf{X}_{1:l}$. Due to the special structure of SMDCS, their rate regions are characterized exactly in [27]. In addition, it is shown that superposition coding suffices for all SMDCS instances.

Another special type of MDCS, known as asymmetrical MDCS (AMDCS), was studied in [32], and is a different special case of the MDCS definition in [30] than the one considered here. In an AMDCS problem in [32], $2^K - 1$ sources with an ordered importance are encoded into K messages. The $2^K - 1$ decoders, where each has access to a non-empty subset of the K messages, are mapped to levels from 1 to $2^K - 1$, i.e., are reproducing $X_1, \mathbf{X}_{1:2}, \ldots, \mathbf{X}_{1:(2^K-1)}$, respectively. It is shown in [32] that superposition (source separation coding) does not suffice for AMDCS when $K = 3$ and coding between sources is necessary to achieve the entire rate region.

Though for special MDCS problems described above, their exact rate regions have been derived with analytical expressions, the exact rate regions of MDCS instances in general are still open. Furthermore, in the general case, it is not clear if the rate regions obtained from the Shannon outer bound are achievable, and, if so, by simple linear codes. This work calculates bounds on rate regions of thousands of MDCS instances, and investigates the sufficiency of simple codes as well.

Contributions: The primary contributions of this chapter include: *i.*) an algorithm to enumerate MDCS instances is given; *ii.*) an analytical expression for the rate region of MDCS is obtained by adapting [15]; *iii.*) an indirect way to calculate MDCS rate regions by bounding them using Shannon outer bound and representable matroid inner bound on region of entropic vectors is proposed; *iv.*) the construction of simple linear codes for achievable rate regions is given; *v.*) rate regions and their achievability by various classes of codes for thousands of MDCS instances are proven and discussed; *vi.*) several embedding operations are defined for which it is proved that non-sufficiency of $\mathbb{F}_q$ linear codes is inherited by a larger MDCS instance from its smaller embedded MDCS instance; *vii.*) using these operations, the thousands of MDCS instances for which scalar binary codes do not suffice are boiled down to 12 forbidden embedded instances; *viii.*) an algorithm to generate converse proofs for rate regions automatically with the help of computer is given.

Notation: A capital letter is used to represent a random variable. For example, X_i is a random variable with index i. A vector is represented by a solid bolded capital letter. For instance, $\mathbf{X}_{1:k} = (X_1, \ldots, X_k)$. Another commonly used notation for index set is $[[K]] = 1 : K$, where $1 : K = \{1, 2, \ldots, K\}$. A set is usually denoted by calligraphic letters like $\mathcal{D}, \mathcal{E}$, etc. We use hollow bolded capital letters, e.g $\mathbb{A}$, to denote matrices with the exception that we still use $\mathbb{F}, \mathbb{R}$ to represent a field and real numbers respectively.

Organization: The rest of this chapter is organized as follows. §2.2 states the problem model, then introduces an algorithm for enumerating MDCS instances. After that, inspired by the notion

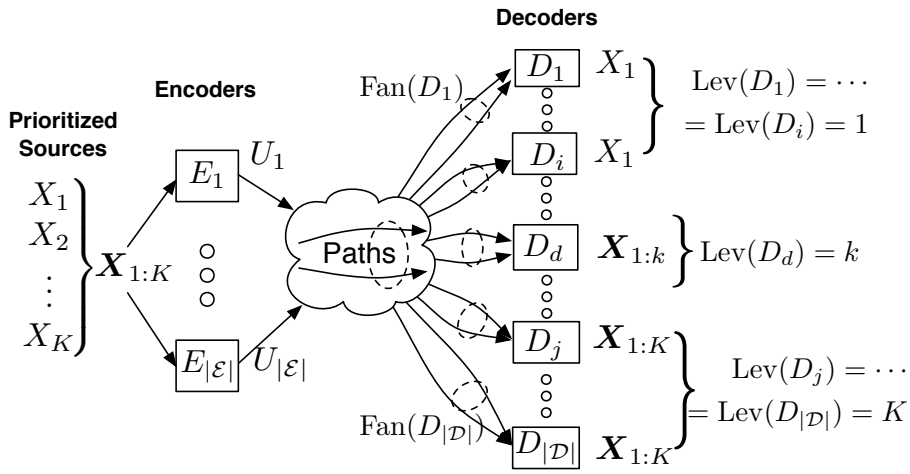


Figure 2.1: Diagram of a general MDCS instance A. *Independent sources* $\mathbf{X}_{1:K}$ *are available to every encoder in* $\mathcal{E}$. *A decoder* D_d *has access to encoders in* $\mathrm{Fan}(D_d)$ *and is able to recover the first* $\mathrm{Lev}(D_d)$ *sources,* $\mathbf{X}_{1:\mathrm{Lev}(D_d)}$.

of forbidden minors from matroid theory [11], some operations which define a notion of embedding between different MDCS instances are defined that preserve insufficiency of a class of codes. The analytical form of the rate regions of MDCS and the way to calculate it by utilizing bounds on region of entropic vectors are presented in §2.3. A construction method for simple codes to achieve the inner bounds is provided in §2.4. In §3.6, we investigate the sufficiency of certain class of simple linear codes, e.g, binary and ternary codes, superposition coding. Rate regions of thousands of bigger MDCS instances that were not studied before are presented in §2.6. Based on the operations defined in §3.6, we find the smallest MDCS instances which, when embedded in a larger MDCS instance, imply the insufficiency of $\mathbb{F}_q$ linear codes and superposition coding. In §2.7 we describe how to generate a converse proof for a MDCS instance with a computer. §3.9 concludes the chapter and states the directions for the future work.

2.2 MDCS and Background

In a MDCS instance, as shown in Fig. 2.1 and denoted as A, there are K *independent* sources $\boldsymbol{X}_{1:K} \equiv (X_1, \ldots, X_K)$ where source k has support $\mathcal{X}_k$, and the sources are prioritized into K levels with X_1 (X_K) the highest (lowest) priority source, respectively. As is standard in source coding, each source X_k is in fact an i.i.d. sequence of random variables $\{X_k^t, t = 1, 2, \ldots\}$ in t, and X_k is a representative random variable with this distribution.

All sources are made available to each of a collection of encoders indexed by the finite set $\mathcal{E}$. The output $\mathrm{Out}(E_e)$ of an encoder E_e is description/message variable $U_e, e \in \mathcal{E}$. The message

variables are mapped to a collection of decoders indexed by the set $\mathcal{D}$ that are classified into K levels, where a level k decoder must losslessly (in the typical Shannon sense) recover source variables $\mathbf{X}_{1:k} = (X_1, \ldots, X_k)$, for each $k \in \{1, \ldots, K\}$. The mapping of encoders to decoders dictates the description variables that are available for each decoder in this recovery. The collection of mappings is denoted as a set $\mathcal{G}$, $\mathcal{G} \subseteq \mathcal{E} \times \mathcal{D}$ of edges where $(E_e, D_d) \in \mathcal{G}$ if E_e is accessible by D_d. The set of encoders mapped to a particular decoder D_d is called the *fan* of D_d, and is denoted as $\text{Fan}(D_d) = \{E_e | (E_e, D_d) \in \mathcal{G}\}$. Similarly, the set of decoders connected to a particular encoder E_e is called the *fan* of E_e, and is denoted by $\text{Fan}(E_e) = \{D_d | (E_e, D_d) \in \mathcal{G}\}$. A decoder D_d is said to be a level k decoder, denoted by $\text{Lev}(D_d) = k$, if it wishes to recover exclusively the first k sources $\mathbf{X}_{1:k}$. Different level k decoders must recover the same subset of source variables using distinct subsets of description variables (encoders), among which one must not be subset of another. If we denote the input and output of a level k decoder D_d as $\text{In}(D_d)$ and $\text{Out}(D_d)$, respectively, we have $\text{In}(D_d) = \{U_e | E_e \in \text{Fan}(D_d), \forall e \in \mathcal{E}\}$ and $\text{Out}(D_d) = \mathbf{X}_{1:k}$.

We say that a MDCS instance is valid if it obeys the following constraints:

(**C1**.) If $\text{Lev}(D_i) = \text{Lev}(D_j)$, then $\text{Fan}(D_i) \not\subseteq \text{Fan}(D_j)$ and $\text{Fan}(D_j) \not\subseteq \text{Fan}(D_i)$;

(**C2**.) If $\text{Lev}(D_i) > \text{Lev}(D_j)$, then $\text{Fan}(D_i) \not\subseteq \text{Fan}(D_j)$;

(**C3**.) $\bigcup_{i \in \mathcal{D}} \text{Fan}(D_i) = \mathcal{E}$;

(**C4**.) There $\nexists\, k, l \in \mathcal{E}$ such that $\text{Fan}(E_k) = \text{Fan}(E_l)$.

(**C5**.) $\forall k \in [[K]],\ \exists\, d \in \mathcal{D}$ such that $\text{Lev}(D_d) = k$.

The first condition (**C1**) indicates that the fan of a decoder cannot be a subset of the fan of another decoder in the same level, for otherwise the decoder with access to more encoders would be redundant. The condition (**C2**) says that the fan of a higher level decoder cannot be a subset of the fan of a lower level decoder, for otherwise there exists a contradiction in their decoding capabilities. The condition (**C3**) requires that every encoder must be contained in the fan of at least one decoder. The condition (**C4**) requires that no two encoders have exactly the same fan, for otherwise the two encoders can be combined together. The condition (**C5**) ensures that there exists at least one decoder for every level.

2.2.1 Representation of MDCS instances

A MDCS instance with K sources and $|\mathcal{E}|$ encoders, shorted as a $(K, |\mathcal{E}|)$ MDCS instance, mainly specifies the relationships between encoders and decoders, since we assume that each encoder has

access to all sources. One representation of a MDCS instance is to list the fan of each decoder. Since $\text{Fan}(D_d) \subseteq \mathcal{E}, \forall d \in \mathcal{D}$, one can represent $\text{Fan}(D_d)$ using a $|\mathcal{E}|$-bit vector or a corresponding integer value, where the entries of the vector from left to right are mapped to $E_{|\mathcal{E}|}, \ldots, E_1$. With this encoding, a MDCS instance can be easily represented by a matrix where the row indices represent the level of decoders and entries are integers representing the fan of each decoder. When some of the levels have fewer decoders than other levels, the row vector includes zeros to make them the same length. For example, the configuration matrix for the MDCS instance shown in Fig. 2.3 is

$$\begin{bmatrix} 0 & 1 \\ 0 & 3 \\ 5 & 6 \end{bmatrix}.$$

This encoding refers to a $(3, 3)$ MDCS instance where there is one level-1 decoder which has access to E_1 ($(001)_2 = 1$), one level-2 decoder which has access to $\{E_1, E_2\}$ ($(011)_2 = 3$), and two level-3 decoders which have access to $\{E_1, E_3\}$ ($(101)_2 = 5$) and $\{E_2, E_3\}$ ($(110)_2 = 6$) respectively.

Another notation for a MDCS instance that we will use extensively in this chapter is the tuple $(\mathbf{X}_{[[K]]}, \mathcal{E}, \mathcal{D}, \mathbf{L}, \mathcal{G})$ where $\mathbf{X}_{[[K]]} = (X_1, \ldots, X_K)$ represents the K sources, $\mathcal{E}$ is the encoder set, $\mathcal{D}$ is the decoder set with corresponding levels in the vector $\mathbf{L}$, and $\mathcal{G}$ is the set of edges between encoders and decoders which indicates the accesses of each decoder: if D_d has access to E_e, the edge $(E_e, D_d) \in \mathcal{G}$. For instance, the MDCS in Fig. 2.3 can be represented using the tuple $(\mathbf{X}_{[[K]]}, \mathcal{E}, \mathcal{D}, \mathbf{L}, \mathcal{G})$ with $K = 3$, $\mathcal{E} = \{E_1, E_2, E_3\}$, $\mathcal{D} = \{D_1, D_2, D_3, D_4\}$, $\mathbf{L} = [1\ 2\ 3\ 3]$ and $\mathcal{G} = \{(E_1, D_1), (E_1, D_2), (E_1, D_3), (E_2, D_2), (E_2, D_3), (E_2, D_4), (E_3, D_4)\}$.

Note that not all pairs of integers $(K, |\mathcal{E}|)$ correspond to a possible MDCS instance. To see why, observe that if a MDCS instance has $|\mathcal{E}|$ encoders, there are at most $2^{|\mathcal{E}|} - 1$ possible decoders since the fan of each decoder is a distinct subset of encoders, and there are at most $2^{|\mathcal{E}|} - 1$ non-empty such subsets. Furthermore, since it is required that there exists at least one decoder for every level as shown in (**C5**), the number of sources or levels K is a lower bound on the number of decoders. Thus, we have

$$K \leq 2^{|\mathcal{E}|} - 1. \tag{2.1}$$

Theoretically speaking, there exist MDCS instances for any $(K, |\mathcal{E}|)$ pair satisfying inequality (2.1). Hence, the next natural question, which will be discussed in next subsection, is how many instances there are for a valid $(K, |\mathcal{E}|)$ pair.

2.2.2 Enumeration of non-isomorphic MDCS instances

When counting the number of possible MDCS instances, we may not wish to distinguish two instances that are symmetric to one another. In particular, though all sources are prioritized, one can permute the encoder variables and associated fan of decoders in a valid MDCS instance to get another symmetric valid MDCS instance. Two such instances are said to be isomorphic to one another.

Definition 1 (Isomorphic MDCS instances)**:** Suppose there are two $(K, |\mathcal{E}|)$ MDCS instances denoted as A, A' with $\mathsf{A} = (\{X_1, \ldots, X_K\}, \mathcal{E}, \mathcal{D}, \mathbf{L}, \mathcal{G})$ and $\mathsf{A}' = (\{X_1, \ldots, X_K\}, \mathcal{E}', \mathcal{D}', \mathbf{L}', \mathcal{G}')$ respectively. A and A' are *isomorphic*, denoted as $\mathsf{A} \cong \mathsf{A}'$, if and only if there exist a permutation of encoders $\mathcal{E}$, $\pi : \mathcal{E} \to \mathcal{E}'$ such that $\mathcal{E}' = \pi(\mathcal{E})$, $\mathcal{D}' = \mathcal{D}$, $\mathbf{L}' = \mathbf{L}$, $\mathcal{G}' = \{(\pi(E_e), D_d) | (E_e, D_d) \in \mathcal{G}\}$. Equivalently, A' can be obtained by permuting encoders of A in the configuration.

Since the isomorphism merely permutes the encoders, to study all possible MDCS instances, it suffices to consider one representative in each isomorphism class, i.e., only consider non-isomorphic MDCS instances.

The easiest way to obtain the list of non-isomorphic MDCS instances is to remove isomorphism from the list of all MDCS instances. In order to obtain all MDCS instances, we observe that the fans of decoders at the same level is a *Sperner family* [16] of the encoders set $\mathcal{E}$ without consideration of empty set, as required by condition (**C1**). A Sperner family of $\mathcal{E}$, sometimes also called an *independent system* or *clutter*, is a collection of subsets of $\mathcal{E}$ such that no element is contained in another.

After including consideration for the conditions (**C2**)–(**C5**), an algorithm to enumerate isomorphic and non-isomorphic MDCS instances is given in Algorithm 8, where the algorithm to augment a MDCS instance for level l with a collection of Sperner families is shown in Algorithm 2. The enumeration process works as follows.

1. List all Sperner families, $\mathrm{Sper}(\mathcal{E})$ of the encoders set $\mathcal{E}$ (required by (**C1**));

2. All Sperner families are possible configurations for level 1, except the empty set and the whole set (when $K > 1$, from (**C5**));

3. For every level l, $l > 1$ (required by (**C5**)), consider current configurations up to level $l - 1$ one by one. For each configuration, remove the Sperner families from the list of all Sperner families containing at least one element that is subset of an element that is already selected in the current configuration (required by (**C2**)). The remaining Sperner families are possible

Input: Encoder index set $\mathcal{E}$, Number of sources K
Output: All non-isomorphic MDCS instances $\mathcal{M}$, all MDCS instances with isomorphism $\mathcal{M}'$

Initialization: List all Sperner families $\text{Sper}(\mathcal{E})$ of $\mathcal{E}$, $pool = \text{Sper}(\mathcal{E})$, $\mathsf{A} = (\mathbf{X}_{[[K]]}, \mathcal{E}, \emptyset, \emptyset, \emptyset)$, $\mathcal{M}' = \mathsf{A}$;

```
for i = 1 : K do
    M'' = M', M' = ∅;
    for every A ∈ M'' do
        pool = Sper(E) \ {I ∈
        Sper(E) | Aug(A, I, i) is invalid MDCS instance from conditions (C1)–(C5)},
        A = Aug(A, pool, i), M' = M' ∪ A;
    end
end
index = 1;
while M' ≠ ∅ do
    M(index) = M'(1); M' = M' \ {I ∈ M' | I ≅ M'(1)};
    index = index + 1;
end
```

Algorithm 1: Enumerate isomorphic and non-isomorphic $(K, |\mathcal{E}|)$ MDCS instances

Input: MDCS instance $\mathsf{A} = (\mathbf{X}_{[[K]]}, \mathcal{E}, \mathcal{D}, \mathcal{G})$, Sperner families $pool$, Level l
Output: All MDCS instances with elements in $pool$ as level l configuration of A:
$\mathcal{A} = Aug(\mathsf{A}, pool, l)$

```
A = ∅;
for every I ∈ pool do
    D' = D ∪ {|D| + 1, ..., |D| + |I|}, L' = [L, l, ..., l] and Length(L')=|L| + |I|;
    for i = 1 : |I| do
        G' = G ∪ {(E_e, D_i) | E_e ∈ I(i)};
    end
    A = A ∪ (X_[[K]], E, D', L', G');
end
```

Algorithm 2: Augment a MDCS instance with Sperner families

Table 2.1: List of numbers of MDCS configurations. $|\mathrm{Sper}(\mathcal{E})|$ *represents the number of all Sperner families of* $\mathcal{E}$*,* $|\mathcal{M}'|$ *is the number of all isomorphic configurations, and* $|\mathcal{M}|$ *is the number of all non-isomorphic configurations. *: Empty set is not considered in the Sperner family; **: [1] counted 3 cases in (2,2) as valid (2,3) MDCS instances and listed 5 instances not satisfying (**C4**); ***: [1] counted the case in (3,2) as a valid (3,3) MDCS instance and missed two. It also listed 2 instances not satisfying (**C4**)*

$(K, \lvert\mathcal{E}\rvert)$	2			3			4		
	$\lvert\mathrm{Sper}(\mathcal{E})\rvert^*$	$\lvert\mathcal{M}'\rvert$	$\lvert\mathcal{M}\rvert$	$\lvert\mathrm{Sper}(\mathcal{E})\rvert^*$	$\lvert\mathcal{M}'\rvert$	$\lvert\mathcal{M}\rvert$	$\lvert\mathrm{Sper}(\mathcal{E})\rvert^*$	$\lvert\mathcal{M}'\rvert$	$\lvert\mathcal{M}\rvert$
1	4	1	1	18	5	3	166	76	13
2	4	5	3	18	96	23^{**}	166	6145	445
3	4	2	1	18	325	68^{***}	166	128388	6803

valid configurations for this current configuration at level l;

4. When $l = K$, if all encoders have been assigned access to at least one decoder (required by (**C3**)), and no two encoders have the same fan (required by (**C4**)), a valid MDCS instance has been obtained;

5. After step 4), all MDCS instances are obtained with isomorphism. Then we remove isomorphism by keeping one instance in every isomorphism class, where the MDCS instances can be obtained by permuting encoders from one another, and remove the others in the isomorphism class from the list of all MDCS instances.

This algorithm lists both all isomorphic and all non-isomorphic K-level $|\mathcal{E}|$-encoder MDCS instances. Note that because the list of Sperner families is fixed, one can use the indices of this list to speed up the enumeration process. Also, two lookup tables for every Sperner family may help in the enumeration regarding the running time. One table records Sperner families who are permutations of a particular Sperner family and the other records those Sperner families which contain a subset of one of its elements. This preprocessing aids rapid isomorphism removal.

The numbers of MDCS instances for some $(K, |\mathcal{E}|)$ pairs are listed in Table 2.1. Hau [1] enumerated 31 distinct MDCS instances for the case of $K = 2$ levels and $|\mathcal{E}| = 3$ encoders, and 69 distinct instances for the case of $K = 3$ levels and $|\mathcal{E}| = 3$ encoders, after symmetries are removed. However, we found that he had three redundant $(K, |\mathcal{E}|) = (2, 3)$ MDCS instances because these three actually should belong to $(K, |\mathcal{E}|) = (2, 2)$ MDCS instances. In addition, he missed two $(K, |\mathcal{E}|) = (3, 3)$ instances and counted one $(3, 2)$ instance as a valid $(3, 3)$ MDCS instance. Furthermore, since we are requiring MDCS instances to satisfy (**C4**), 5 (2) instances in $(2, 3)$ ($(3, 3)$) that violate this condition are found. Therefore, there are 23 and 68 non-isomorphic instances for $(K, |\mathcal{E}|) = (2, 3)$ and $(K, |\mathcal{E}|) = (3, 3)$ respectively in our lists.

2.3 Analytical form of the rate region

In this section, we present an analytical form of the coding rate (capacity) region of MDCS instances in terms of Γ_N^*, which has similar formulation of the rate region of general multi-source multi-sink acyclic networks [15]. In addition, we review some useful bounds on Γ_N^* that can be used to compute the rate region.

2.3.1 Expression of rate region

As shown in Fig. 2.1, we suppose that the source variable generated at source k is X_k with support alphabet $\mathcal{X}_k$ and normalized entropy $H(X_k) = \sum_{x\in\mathcal{X}_k} -p_x \log_q(p_x)$, for every $k \in [[K]]$. Let $\mathbf{R}_\mathcal{E} = (R_1, \ldots, R_{|\mathcal{E}|}) \in \mathbb{R}_+^{|\mathcal{E}|}$ be the rate vector for the encoders, measured in number of $\mathbb{F}_q$ digits. An $(n, \mathbf{R})$ block code in $\mathbb{F}_q$ is defined as follows. For each blocklength $n \in \mathbb{N}$ we consider a collection of $|\mathcal{E}|$ block encoders, where encoder E_e maps each block of n variables from each of the K sources to one of q^{nR_e} different descriptions,

$$f_e^{(n)} : \prod_{k=1}^{K} \mathcal{X}_k^n \to \{1, \ldots, q^{nR_e}\},\ e \in \mathcal{E}. \tag{2.2}$$

The encoder outputs are indicated by $U_e = f_e^{(n)}(\mathbf{X}_{1:K}^{1:n})$ for $e \in \mathcal{E}$. The $|\mathcal{D}|$ decoders are characterized by their priority level and their available descriptions. Specifically, decoder D_d has an associated priority level $k \in [[K]]$ and an available subset of the descriptions, i.e., input of fan of decoder D_d. In other words, decoder D_d has input from its fan $\mathbf{U}_{\mathrm{In}(D_d)} = (U_e, E_e \in \mathrm{Fan}(D_d))$ and must asymptotically losslessly recover source variables $\mathbf{X}_{1:k}$,

$$g_d^{(n)} : \prod_{e:E_e\in\mathrm{Fan}(D_d)} \{1, \ldots, q^{nR_e}\} \to \prod_{i=1}^{k} \mathcal{X}_i^n,\ d \in \mathcal{D}. \tag{2.3}$$

The rate region $\mathcal{R}_\mathcal{E}$ for the configuration specified by K, $\mathcal{E}$, and the encoder to decoder mappings $(\mathrm{Fan}(D_d),\ d \in \mathcal{D})$ consists of all rate vectors $\mathbf{R}_\mathcal{E} = (R_1, \ldots, R_{|\mathcal{E}|})$ such that there exist sequences of encoders $f^n = (f_e^n, e \in \mathcal{E})$ and decoders $g^n = (g_d^n, d \in \mathcal{D})$ for which the asymptotic probability of error goes to zero in n at all decoders. Specifically, define the probability of error for each level $l_d = \mathrm{Lev}(D_d)$ decoder D_d as

$$p_{d,l_d}^{n,\mathrm{err}}(\mathbf{R}) = \mathbb{P}(g_d(f_e(\mathbf{X}_{1:K}^{1:n}), E_e \in \mathrm{Fan}(D_d)) \neq \mathbf{X}_{1:l_d}^{1:n}), \tag{2.4}$$

and the maximum over these as

$$p^{n,\mathrm{err}}(\mathbf{R}) = \max_{k\in[[K]]} \max_{d:l_d=k} p_{d,k}^{n,err}. \tag{2.5}$$

A rate vector is in the rate region, $\mathbf{R}_{\mathcal{E}} \in \mathcal{R}_{\mathcal{E}}$, provided there exists $\{f_e^n\}$ and $\{g_d^n\}$ such that $p^{n,\mathrm{err}}(\mathbf{R}) \to 0$ as $n \to \infty$.

The rate region $\mathcal{R}_{\mathcal{E}}$ can be expressed in terms of the region of entropic vectors, Γ_N^* [15]. For the MDCS problem, collect all of the involved random variables into the set $\mathcal{N} = \{Y_k, k \in [[K]]\} \cup \{U_e, e \in \mathcal{E}\}$ and define $N = |\mathcal{N}|$, where $Y_k, k \in [[K]]$ is the auxiliary random variable associated with $X_k, k \in [[K]]$, and U_e is the random variable associated with encoder $e \in \mathcal{E}$. The rate region $\mathcal{R}_{\mathcal{E}}$ is the set of rate vectors $\mathbf{R}_{\mathcal{E}}$ such that there exists $\mathbf{h} \in \Gamma_N^*$ satisfying the following (see [15])

$$\begin{aligned}
h_{\mathbf{Y}_{1:K}} &= \sum_{k=1}^{K} h_{Y_k} && (2.6)\\
h_{U_e|\mathbf{Y}_{1:K}} &= 0, e \in \mathcal{E} && (2.7)\\
h_{Y_k} &\geq H(X_k), k \in [[K]] && (2.8)\\
h_{\mathbf{Y}_{1:\mathrm{Lev}(D_d)}|\mathbf{U}_{\mathrm{In}(D_d)}} &= 0, \forall d \in \mathcal{D} && (2.9)\\
R_e &\geq h_{U_e}, e \in \mathcal{E} && (2.10)
\end{aligned}$$

where the conditional entropies $h_{A|B}$ are naturally equivalent to $h_{AB} - h_B$. These constraints can be interpreted as follows: (3.5) represents that sources are independent; (3.6) represents that each description is a function of all sources available; (3.7) represents that (3.8) represents that the source rate constraints; recovered source messages at a decoder are a function of the input descriptions available to it; and (3.9) represents the coding rate constraints.

Define the sets

$$\begin{aligned}
\mathcal{L}_1 &= \{\mathbf{h} \in \mathbb{R}_+^{2^N-1} : h_{\mathbf{Y}_{1:K}} = \sum_{i\in[[K]]} h_{Y_i}\} && (2.11)\\
\mathcal{L}_2 &= \{\mathbf{h} \in \mathbb{R}_+^{2^N-1} : h_{U_e|\mathbf{Y}_{1:K}} = 0, e \in \mathcal{E}\} && (2.12)\\
\mathcal{L}_4 &= \{\mathbf{h} \in \mathbb{R}_+^{2^N-1} : h_{Y_k} \geq H(X_k), k \in [[K]]\} && (2.13)\\
\mathcal{L}_5 &= \{\mathbf{h} \in \mathbb{R}_+^{2^N-1} : h_{\mathbf{Y}_{1:k}|\mathbf{U}_{\mathrm{In}(D_d)}} = 0, d \in \mathcal{D}\} && (2.14)
\end{aligned}$$

Note that the sets $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_4, \mathcal{L}_5$ are corresponding to the constraints (3.5), (3.6), (3.7), (3.8), re-

spectively. Define

$$\mathcal{R}'_{\mathcal{E}} = \mathrm{Ex}(\mathrm{Proj}_{h_{U_e, e\in\mathcal{E}}}(\overline{\mathrm{con}(\Gamma^*_{\mathrm{N}} \cap \mathcal{L}_{12})} \cap \mathcal{L}_4 \cap \mathcal{L}_5)) \tag{2.15}$$

where $\mathcal{L}_{12} = \mathcal{L}_1 \cap \mathcal{L}_2$, $\mathrm{Proj}_{h_{U_e, e\in\mathcal{E}}}(\mathcal{B})$ is the projection of the set $\mathcal{B}$ on the coordinates $(h_{U_e}, e \in \mathcal{E})$, and $\mathrm{Ex}(\mathcal{B}) = \{\mathbf{h} \in \mathbb{R}_+^{|\mathcal{E}|} : \mathbf{h} \geq \mathbf{h}' \text{ for some } \mathbf{h}' \in \mathcal{B}\}$, for $\mathcal{B} \subset \mathbb{R}_+^{|\mathcal{E}|}$. Then, $\mathcal{R}'_{\mathcal{E}}$ is equivalent to the rate region $\mathcal{R}_{\mathcal{E}}$.

Theorem 1:

$$\mathcal{R}_{\mathcal{E}} = \mathcal{R}'_{\mathcal{E}}. \tag{2.16}$$

In pursuing the proof of Theorem 10, a brief review of [15] is presented here. In [15], an implicit characterization of the achievable rate region for a general acyclic multi-source multi-sink network coding problem is obtained in terms of Γ^*_N, the fundamental region of entropic vectors. The achievable rate region $\mathcal{R}_{\mathcal{S}}$ consists of all feasible source rate vectors $\mathbf{R}_{\mathcal{S}}$, whose part is played by the source entropies $H(X_k), k \in [[K]]$ in our formulation, for given network link capacities $R_e, e \in \mathcal{E}$, whose part is played by the encoder rates in our formulation. In particular, the network coding capacity region expression in [15] assumes a series of fixed network link capacities $R_e, e \in \mathcal{E}$, and calculates from this a series of possible source rates. Here, we will no longer view the link capacities as fixed, and instead aim to obtain a series of inequalities, forming a convex cone, which link the rates $R_e, e \in \mathcal{E}$ and the source entropies $H(X_k), k \in [[K]]$ to describe the rate region for all possible rates and all possible source entropies. To do this, we will first show how to modify the approach from [15] to get the region of possible link capacities/ encoder rates $R_e, e \in \mathcal{E}$ for a fixed series of source entropies, then show how to further modify the expressions to get a series of inequalities which treat both $R_e, e \in \mathcal{E}$ and $H(X_k), k \in [[K]]$ as variables.

It is not difficult to find that MDCS is a special case of the general model in [15]. The degenerated points include:

1. No intermediate nodes are considered in the MDCS model, while intermediate nodes are included in the model in [15].

2. In [15], the output of a decoder is an arbitrary collection of sources. However, in MDCS, the output of a decoder is specified in a particular manner such that a level l decoder requires $X_1, \ldots, X_l$, the first l sources.

Due to the peculiarities of MDCS, if we delete $\mathcal{L}_3$ in [15], which corresponds to the encoding function for intermediate nodes, and change their decoding constraints and functions to (2.14) and

(3.2) respectively, we can define a source rate region $\mathcal{R}'_{\mathcal{S}}$ of MDCS for channel capacities (encoding rate capacities) $R_e, e \in \mathcal{E}$ as

$$\mathcal{R}'_{\mathcal{S}} = \Lambda(\mathrm{Proj}_{h_{Y_k}, k\in[[K]]}(\overline{\mathrm{con}(\Gamma^*_{\mathrm{N}} \cap \mathcal{L}_{12})} \cap \mathcal{L}'_4 \cap \mathcal{L}_5)), \tag{2.17}$$

where $\Lambda(\mathcal{A}') = \{\mathbf{h} \in \mathbb{R}^K_+ : \mathbf{h} \leq \mathbf{h}' \text{ for some } \mathbf{h}' \in \mathcal{A}'\}$ and $\mathcal{L}'_4 = \{\mathbf{h} \in \mathbb{R}^{2^N-1}_+ : h_{U_e} \leq R_e, e \in \mathcal{E}\}$ corresponding to the rate constraints in (3.9).

Then we can derive the following Corollary from Theorem 1 in [15].

Corollary 1 ([15] Theorem 1 applied to MDCS)**:**

$$\mathcal{R}_{\mathcal{S}} = \mathcal{R}'_{\mathcal{S}}. \tag{2.18}$$

Proof. The only difference between this corollary and Theorem 1 in [15] is that we do not have the constraint $\mathcal{L}_3$ which corresponds to the functions of intermediate nodes. □

Based on Corollary 1, we provide the proof of Theorem 10.

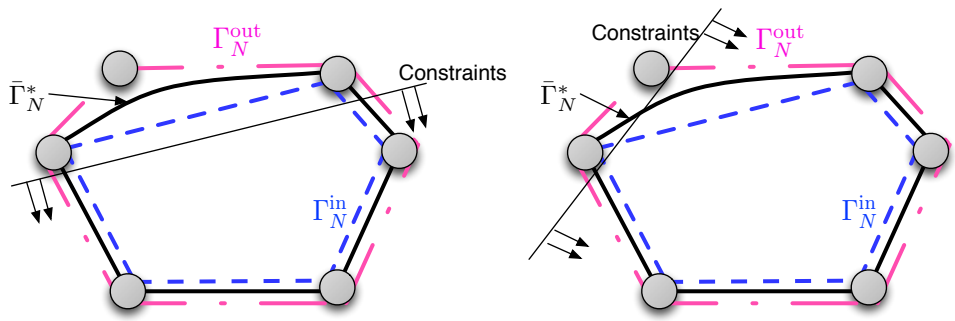
Proof of Theorem 10:

Converse: We need to show that for any point $\mathbf{R}_{\mathcal{E}} = (R_1, \ldots, R_{|\mathcal{E}|})$ in the rate region $\mathcal{R}_{\mathcal{E}}$ respect to source rates $H(X_1), \ldots, H(X_K)$, we have $\mathbf{R}_{\mathcal{E}} \in \mathcal{R}'_{\mathcal{E}}$, i.e., $\mathcal{R}_{\mathcal{E}} \subseteq \mathcal{R}'_{\mathcal{E}}$.

Since $\mathbf{R}_{\mathcal{E}}$ is achievable, there exist encoding and decoding functions at encoders and decoders such that all decoding requirements are satisfied with source rates at $H(X_1), \ldots, H(X_K)$ and coding rates $H(U_e) \leq R_e, e \in \mathcal{E}$. In other words, if $\mathbf{R}_{\mathcal{E}}$ is given as capacities of the encoders in question, the same encoding and decoding functions make the source rate tuple $\mathbf{R}_{\mathcal{S}} = (H(X_1), \ldots, H(X_K))$ achievable, i.e., $\mathbf{R}_{\mathcal{S}} \in \mathcal{R}_{\mathcal{S}}$. From Corollary 1 we see that $\mathbf{R}_{\mathcal{S}} \in \mathcal{R}'_{\mathcal{S}}$. Then there exists a vector $\mathbf{h} \in \overline{\mathrm{con}(\Gamma^*_{\mathrm{N}} \cap \mathcal{L}_{12})} \cap \mathcal{L}'_4 \cap \mathcal{L}_5$ such that $\mathbf{R}_{\mathcal{S}} = \mathrm{Proj}_{Y_k, k\in[[K]]}(\mathbf{h})$. We see that $\mathbf{h} \in \mathcal{L}_4$ since $h_{Y_k} = H(X_k), k \in [[K]]$. Therefore, we have $\mathbf{h} \in \overline{\mathrm{con}(\Gamma^*_{\mathrm{N}} \cap \mathcal{L}_{12})} \cap \mathcal{L}_4 \cap \mathcal{L}_5$. Also we know that $\mathbf{h} \in \mathcal{L}'_4$, $h_{U_e} \leq R_e, e \in \mathcal{E}$. Then we can get $\mathbf{R}_{\mathcal{E}} = (R_e, e \in \mathcal{E}) \in \mathrm{Ex}(\mathrm{Proj}_{h_{U_e}, e\in\mathcal{E}}(\overline{\mathrm{con}(\Gamma^*_{\mathrm{N}} \cap \mathcal{L}_{12})} \cap \mathcal{L}_4 \cap \mathcal{L}_5))$. Therefore, $\mathbf{R}_{\mathcal{E}} \in \mathcal{R}'_{\mathcal{E}}$ and further $\mathcal{R}_{\mathcal{E}} \subseteq \mathcal{R}'_{\mathcal{E}}$.

Achievability: We need to show that for any point $\mathbf{R}'_{\mathcal{E}} \in \mathcal{R}'_{\mathcal{E}}$, there exists a code to achieve it. Formally, $\forall \mathbf{R}'_{\mathcal{E}} \in \mathcal{R}'_{\mathcal{E}}$, we have $\mathbf{R}'_{\mathcal{E}} \in \mathcal{R}_{\mathcal{E}}$.

Suppose $\mathbf{R}'_{\mathcal{E}} \in \mathcal{R}'_{\mathcal{E}}$. Then there exists a vector $\mathbf{h}' \in \overline{\mathrm{con}(\Gamma^*_{\mathrm{N}} \cap \mathcal{L}_{12})} \cap \mathcal{L}_4 \cap \mathcal{L}_5$ such that $\mathbf{R}'_{\mathcal{E}} = \mathrm{Proj}_{U_e, e\in\mathcal{E}}(\mathbf{h}')$. We see that $\mathbf{h}' \in \mathcal{L}'_4$ since $h_{U_e} = H(U_e), e \in \mathcal{E}$. Therefore, we have $\mathbf{h}' \in \overline{\mathrm{con}(\Gamma^*_{\mathrm{N}} \cap \mathcal{L}_{12})} \cap \mathcal{L}'_4 \cap \mathcal{L}_5$. According to Corollary 1, we get $\mathrm{Proj}_{H(X_k), k\in[[K]]}\mathbf{h}' \in \mathcal{R}_{\mathcal{S}}$. By the



(a) When constraints cut off gap (after projection) between outer and inner bound, we have $\mathcal{R} = \mathcal{R}_{\text{out}} = \mathcal{R}_{\text{in}}$

(b) When constraints do NOT cut off gap (after projection) between outer and inner bound, we have $\mathcal{R}_{\text{in}} \subseteq \mathcal{R} \subseteq \mathcal{R}_{\text{out}}$

Figure 2.2: Illustration of computing rate region via bounding the region of entropic vectors

proof of Corollary 1 in [15], $\mathbf{h}'$ is achievable by some codes with capacities of $\mathbf{R}'_{\mathcal{E}}$. Also we know that $\mathbf{h}' \in \mathcal{L}_4$, $h_{Y_k} \geq H(X_k), k \in [[K]]$. Then we can get $\mathbf{R}_{\mathcal{S}} = (H(X_k), k \in [[K]]) \leq \text{Proj}_{h_{Y_k}, k \in [[K]]}(\mathbf{h}')$ is also achievable with capacities of $R_e = h'_{U_e}, e \in \mathcal{E}$. Equivalently, if we set source rates as $H(X_k), k \in [[K]]$, there exists a code to achieve the coding rates $R_e, e \in \mathcal{E}$ using the same code. Therefore, $\mathbf{R}'_{\mathcal{E}}$ is achievable and further $\mathcal{R}'_{\mathcal{E}} \subseteq \mathcal{R}_{\mathcal{E}}$. ■

2.3.2 Computation of rate region

While the analytical formation gives a possible way in principle to calculate the rate region of any MDCS instance, we still have some problems. We know that Γ_N^* is unknown and even not polyhedral for $N \geq 4$. Thus, the direct calculation of rate regions from (2.15) for a MDCS instance with more than 4 variables is infeasible. However, replacing Γ_N^* with polyhedral inner and outer bounds allows (2.15) to become a polyhedral computation, which involves applying some constraints onto a polyhedra and then projecting down onto some coordinates. This inspires us to substitute Γ_N^* with its closed polyhedral outer Γ_N^{out} and inner bounds Γ_N^{in} respectively, and get an outer

$$\mathcal{R}_{\text{out}} = \text{Ex}(\text{Proj}_{h_{U_e}, e \in \mathcal{E}}(\Gamma_N^{\text{out}} \cap \mathcal{L}_{1245})) \tag{2.19}$$

and inner bound

$$\mathcal{R}_{\text{in}} = \text{Ex}(\text{Proj}_{h_{U_e}, e \in \mathcal{E}}(\Gamma_N^{\text{in}} \cap \mathcal{L}_{1245})) \tag{2.20}$$

on the rate region. If $\mathcal{R}_{\text{out}} = \mathcal{R}_{\text{in}}$, we know $\mathcal{R} = \mathcal{R}_{\text{out}} = \mathcal{R}_{\text{in}}$. Otherwise, tighter bounds are necessary. Fig. 2.2 illustrates these two situations.

As described previously, it is desirable to specify a rate region as a series of inequalities linking

sources entropies and encoder rates. However, the equations (2.15), (3.15) and (3.16) are functions of the given source entropies $H(X_k), k \in [[K]]$, as can be seen from the constraints $\mathcal{L}_4$ in (2.13). With a slight abuse of notation, if we define

$$\mathcal{L}_4'' = \{(\mathbf{h}^T, \mathbf{R}^T)^T \in \mathbb{R}_+^{2^N-1+|\mathcal{E}|} : R_e \geq h_{U_e}, e \in \mathcal{E}\}, \tag{2.21}$$

the rate regions in (2.15), (3.15) and (3.16), which are expressed exclusively in terms of the variables $R_e, H(X_k), e \in \mathcal{E}, k \in [[K]]$, will be

$$\mathcal{R}_{\mathcal{E}} = \text{Proj}_{R_e, e\in\mathcal{E}, H(X_k), k\in[[K]]}(\overline{\text{con}(\Gamma_{\text{N}}^* \cap \mathcal{L}_{12})} \cap \mathcal{L}_5 \cap \mathcal{L}_4'') \tag{2.22}$$

$$\mathcal{R}_{\text{out}} = \text{Proj}_{R_e, e\in\mathcal{E}, H(X_k), k\in[[K]]}(\Gamma_N^{\text{out}} \cap \mathcal{L}_{125} \cap \mathcal{L}_4'') \tag{2.23}$$

$$\mathcal{R}_{\text{in}} = \text{Proj}_{R_e, e\in\mathcal{E}, H(X_k), k\in[[K]]}(\Gamma_N^{\text{in}} \cap \mathcal{L}_{125} \cap \mathcal{L}_4''). \tag{2.24}$$

Here, the projection on $H(X_k)$ is in fact projection on $\mathbf{h}_{Y_k}$ since we let $H(X_k) = \mathbf{h}_{Y_k}, k \in [[K]]$. Note that, $\mathcal{L}_4''$ introduces new variables and is a cone in $2^N - 1 + |\mathcal{E}|$ dimensional space. In addition, since $\bar{\Gamma}_N^*, \mathcal{L}_{1,2,5}$ do not have dimensions on $R_e, e \in \mathcal{E}$, we add these free dimensions in the intersection.

In this work, we will follow (3.15) and (3.16) to calculate the rate region. Typically, the Shannon outer bound Γ_N and some inner bounds obtained from matroids, especially representable matroids, are used. We will introduce these bounds in the next subsection. For details on the polyhedral computation methods used to obtain these bounds, interested readers are referred to [22, 23, 33].

2.3.3 Construction of bounds on rate region

We pass now to discussing the bounds on the region of entropic vectors utilized in our work. We would like to first review the region of entropic vectors.

2.3.3.1 Region of entropic vectors Γ_N^*

Consider an arbitrary collection $\mathbf{X} = (X_1, \ldots, X_N)$ of N discrete random variables with joint probability mass function p_X. To each of the $2^N - 1$ non-empty subsets of the collection of random variables, $X_{\mathcal{A}} := (X_i \,|i \in \mathcal{A})$ with $\mathcal{A} \subseteq \{1, \ldots, N\} \equiv [[N]]$, there is associated a joint Shannon entropy $H(X_{\mathcal{A}})$. Stacking these subset entropies for different subsets into a $2^N - 1$ dimensional vector we form an entropy vector

$$\mathbf{h} = [H(X_{\mathcal{A}}) \,|\mathcal{A} \subseteq [[N]], \mathcal{A} \neq \emptyset] \tag{2.25}$$

By virtue of having been created in this manner, the vector $\mathbf{h}$ must live in some subset of $\mathbb{R}_+^{2^N-1}$, and is said to be *entropic* due to the existence of p_X. However, not every point in $\mathbb{R}_+^{2^N-1}$ is entropic since for some many points, there does not exist an associated valid distribution p_X. The set of all entropic vectors form a region, denoted as Γ_N^*. It is known that the closure of the region of entropic vectors $\bar{\Gamma}_N^*$ is a convex cone [15].

2.3.3.2 Shannon outer bound Γ_N

Next observe that elementary properties of Shannon entropies indicates that $H(X_{\mathcal{A}})$ is a non-decreasing submodular function, so that $\forall \mathcal{A} \subseteq \mathcal{B} \subseteq [[N]], \forall \mathcal{C}, \mathcal{D} \subseteq [[N]]$

$$H(X_{\mathcal{A}}) \leq H(X_{\mathcal{B}}) \tag{2.26}$$

$$H(X_{\mathcal{C}\cup\mathcal{D}}) + H(X_{\mathcal{C}\cap\mathcal{D}}) \leq H(X_{\mathcal{C}}) + H(X_{\mathcal{D}}). \tag{2.27}$$

Since they are true for any collection of subset entropies, these linear inequalities (3.18), (3.19) can be viewed as supporting halfspaces for Γ_N^*.

Thus, the intersection of all such inequalities form a polyhedral outer bound Γ_N for Γ_N^* and $\bar{\Gamma}_N^*$, where

$$\Gamma_N := \left\{ \mathbf{h} \in \mathbb{R}_{\geq 0}^{2^N-1} \;\middle|\; \begin{array}{c} h_{\mathcal{A}} \leq h_{\mathcal{B}} \quad \forall \mathcal{A} \subseteq \mathcal{B} \\ h_{\mathcal{C}\cup\mathcal{D}} + h_{\mathcal{C}\cap\mathcal{D}} \leq h_{\mathcal{C}} + h_{\mathcal{D}} \quad \forall \mathcal{C}, \mathcal{D} \end{array} \right\}.$$

This outer bound Γ_N is known as the *Shannon outer bound*, as it can be thought of as the set of all inequalities resulting from the positivity of Shannon's information measures among the random variables. Fujishige observed in 1978 [34] that the entropy function for a collection of random variables $(X_i, i \in [N])$ viewed as a set function is a polymatroid rank function, where a set function $\rho : 2^{\mathcal{S}} \to \mathbb{R}_+$ is a rank function of a polymatroid if it obeys the following axioms:

1. Normalization: $\rho(\emptyset) = 0$;
2. Monotonicity: if $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{S}$ then $\rho(\mathcal{A}) \leq \rho(\mathcal{B})$;
3. Submodularity: if $\mathcal{A}, \mathcal{B} \subseteq \mathcal{S}$ then $\rho(\mathcal{A} \cup \mathcal{B}) + \rho(\mathcal{A} \cap \mathcal{B}) \leq \rho(\mathcal{A}) + \rho(\mathcal{B})$.

While $\Gamma_2 = \Gamma_2^*$ and $\Gamma_3 = \bar{\Gamma}_3^*$, $\bar{\Gamma}_N^* \subsetneq \Gamma_N$ for all $N \geq 4$ [15], and indeed it is known [35] that $\bar{\Gamma}_N^*$ is not even polyhedral for $N \geq 4$.

2.3.3.3 Matroid basics

Matroid theory [11] is an abstract generalization of the independence in the context of linear algebra to the more general setting of set systems, i.e., collections of subsets of a ground set obeying certain axioms. The ground set of size N is without loss of generality $\mathcal{S} = [[N]]$, and in our context each element of the ground set will correspond to a random variable. There are numerous equivalent definitions of matroids; we first present one commonly used in terms of independent sets.

Definition 2: [11] A matroid M is an ordered pair $(\mathcal{S}, \mathcal{I})$ consisting of a finite set $\mathcal{S}$ (the ground set) and a collection $\mathcal{I}$ (called independent sets) of subsets of S obeying:

1. Normalization: $\emptyset \in \mathcal{I}$;
2. Heredity: If $I \in \mathcal{I}$ and $I' \subseteq I$, then $I' \in \mathcal{I}$;
3. Independence augmentation: If $I_1 \in \mathcal{I}$ and $I_2 \in \mathcal{I}$ and $|I_1| < |I_2|$, then there is an element $e \in I_2 - I_1$ such that $I_1 \cup \{e\} \in \mathcal{I}$.

Another common (and equivalent) definition of matroids utilizes *rank functions*. For a matroid $M = (\mathcal{S}, \mathcal{I})$ with $|\mathcal{S}| = N$ the rank function $r : 2^{\mathcal{S}} \to \{0, \ldots, N\}$ is defined as the size of the largest independent set contained in each subset of $\mathcal{S}$, i.e., $r_M(\mathcal{A}) = \max_{\mathcal{B} \subseteq \mathcal{A}}\{|\mathcal{B}| : \mathcal{B} \in \mathcal{I}\}$. The rank of a matroid, r_M, is the rank of the ground set, $r_M = r_M(\mathcal{S})$. The rank function of a matroid can be shown to obey the following properties. In fact these properties may instead be viewed as an alternate definition of a matroid in that any set function obeying these axioms is the rank function of a matroid.

Definition 3: A set function $r : 2^{\mathcal{S}} \to \{0, \ldots, N\}$ is a rank function of a matroid if it obeys the following axioms:

1. Cardinality: $r_M(\mathcal{A}) \leq |\mathcal{A}|$;
2. Monotonicity: if $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{S}$ then $r_M(\mathcal{A}) \leq r_M(\mathcal{B})$;
3. Submodularity: if $\mathcal{A}, \mathcal{B} \subseteq \mathcal{S}$ then $r_M(\mathcal{A} \cup \mathcal{B}) + r_M(\mathcal{A} \cap \mathcal{B}) \leq r_M(\mathcal{A}) + r_M(\mathcal{B})$.

There are many operations on matroids, such as *contraction* and *deletion*. Details about these operations can be found in [11]. Next we give the definition of the important concept of a matroid *minor* based on these two operations.

Definition 4: If M is a matroid on $\mathcal{S}$ and $\mathcal{T} \subseteq \mathcal{S}$, a matroid M' on $\mathcal{T}$ is called a *minor* of M if M' is obtained by any combination of deletion ($\backslash$) and contraction ($/$) of M.

The operations of deletion and contraction mentioned in the definition yield new matroids with new rank functions for the minors. Specifically, let $M/\mathcal{T}$ denote the matroid obtained by contraction of M on $\mathcal{T} \subset \mathcal{S}$, and let $M \setminus \mathcal{T}$ denote the matroid obtained by deletion from M of $\mathcal{T} \subset \mathcal{S}$. Then, by [11] (3.1.5,7), $\forall \mathcal{X} \subseteq \mathcal{S} \setminus \mathcal{T}$

$$\begin{aligned} r_{M/\mathcal{T}}(\mathcal{X}) &= r_M(\mathcal{X} \cup \mathcal{T}) - r_M(\mathcal{T}) \\ r_{M\setminus\mathcal{T}}(\mathcal{X}) &= r_M(\mathcal{X}) \end{aligned} \tag{2.28}$$

To each set function $r_N : 2^{[[N]]} \to \mathbb{R}_{\geq 0}$ we will associate a vector $\mathbf{r}_N \in \mathbb{R}_{\geq 0}^{2^N-1}$ formed by stacking the various values of the function r_N into a vector, e.g.., in a manner associated with a binary counter, such as

$$\mathbf{r} = \left[\; r(\{1\}) \; r(\{2\}) \; r(\{1,2\}) \; r(\{3\}) \; r(\{1,3\}) \; r(\{2,3\}) \; r(\{1,2,3\}) \; r(\{4\}) \; \cdots \; \right]^T . \tag{2.29}$$

For any such function, r_N, and hence for any such vector $\mathbf{r}_N$, for any pair of sets $\mathcal{A}, \mathcal{B} \subseteq [[N]]$ we will define the *minor* associated with deleting $[[N]] \setminus (\mathcal{A} \cup \mathcal{B})$ and contracting on $\mathcal{A}$ as

$$r_{\mathcal{B}|\mathcal{A}}(\mathcal{C}) := r(\mathcal{C} \cup \mathcal{A}) - r(\mathcal{A}) \quad \forall \mathcal{C} \subseteq \mathcal{B} \tag{2.30}$$

and $\mathbf{r}_{\mathcal{B}|\mathcal{A}}$ will denote the associated vector, ordered again by a binary counter whose bit positions are created by enumerating again the elements of $\mathcal{B}$ (i.e. keeping the same order). If r is the rank function of a matroid, this definition is consistent with the definition of the matroid operations of taking a minor, deleting and contracting, although we will apply them to any real valued set function here.

Though there are many classes of matroids, we are especially interested in one of them, *representable matroids*, because they can be related to linear codes to solve network coding problems, as discussed in [22, 23].

2.3.3.4 Representable matroids

Representable matroids are an important class of matroids which connect the independent sets to the conventional notion of independence in a vector space.

Definition 5: A matroid M with ground set S of size $|S| = N$ and rank $r_M = r$ is representable over a field $\mathbb{F}$ if there exists a matrix $\mathbb{A} \in \mathbb{F}^{r \times N}$ such that for each independent set $I \in \mathcal{I}$ the corresponding columns in $\mathbb{A}$, viewed as vectors in $\mathbb{F}^r$, are linearly independent.

There has been significant effort towards characterizing the set of matroids that are representable over various field sizes, with a complete answer only available for fields of sizes two, three, and four. For example, the characterization of binary representable matroids due to Tutte is: A matroid M is binary representable (representable over a binary field) iff it does not have the matroid $U_{2,4}$ as a minor. Here, $U_{k,N}$ is the *uniform* matroid on the ground set $S = [N]$ with independent sets $\mathcal{I}$ equal to all subsets of $[N]$ of size at most k. For example, $U_{2,4}$ has as its independent sets

$$\mathcal{I} = \{\emptyset, 1, 2, 3, 4, \{1,2\}, \{1,3\}, \{1,4\}, \{2,3\}, \{2,4\}, \{3,4\}\}. \tag{2.31}$$

Another important observation is that the first non-representable matroid is *Vámos* matroid, a well known matroid on ground set of size 8. That is to say, all matroids are representable, at least in some field, for $N \leq 7$.

2.3.3.5 Inner bounds from representable matroids

Suppose a matroid M with ground set $\mathcal{S}$ of size $|\mathcal{S}| = N$ and rank $r_M(\mathcal{S}) = k$ is representable over the finite field $\mathbb{F}_q$ of size q and the representing matrix is $\mathbb{A} \in \mathbb{F}_q^{k \times N}$ such that $\forall \mathcal{B} \subseteq \mathcal{S}$ $r_M(\mathcal{B}) = \text{rank}(\mathbb{A}_{:,\mathcal{B}})$, the matrix rank of the columns of $\mathbb{A}$ indexed by B. Let Γ_N^q be the conic hull of all rank functions of matroid with N elements and representable in $\mathbb{F}_q$. This provides an inner bound $\Gamma_N^q \subseteq \bar{\Gamma}_N^*$, because any extremal rank function r of Γ_N^q is by definition representable and hence is associated with a matrix representation $\mathbb{A} \in \mathbb{F}_q^{k \times N}$, from which we can create the random variables

$$(X_1, \ldots, X_N) = \mathbf{u}\mathbb{A}, \quad \mathbf{u} \sim \mathcal{U}(\mathbb{F}_q^k). \tag{2.32}$$

whose elements have entropy for each subset $\mathcal{A}$ of $h_{\mathcal{A}} = r_M(\mathcal{A}) \log_2 q$, for $\mathcal{A} \subseteq \mathcal{S}$. Hence, all extreme rays of Γ_N^q are entropic, and $\Gamma_N^q \subseteq \bar{\Gamma}_N^*$. Further, as will be discussed in §2.4, if a vector in the rate region of a network is (projection of) a $\mathbb{F}_q$-representable matroid rank, the representation $\mathbb{A}$ can be used as a linear code to achieve that rate vector and this code is denoted as a *scalar* $\mathbb{F}_q$ *code*.

One can further generalize the relationship between representable matroids and entropic vectors established by (3.21) by partitioning $\mathcal{S} = \{1, \ldots, N'\}$ up into N disjoint sets, $\mathcal{S}_1, \ldots, \mathcal{S}_N$ and defining for $n \in \{1, \ldots, N\}$ the new vector-valued, random variables $\mathbf{X}'_n = [X_n | n \in \mathcal{S}_{n'}]$. The associated

entropic vector will have entropies $h_A = r_M(\cup_{n\in A}\mathcal{S}_n)\log_2 q$, and is thus proportional to a *projection* of the original rank vector $\mathbf{r}$ keeping only those elements corresponding to all elements in a set in the partition appearing together. Thus, such a projection of $\Gamma^q_{N'}$ forms an inner bound to $\bar{\Gamma}^*_N$, which we will refer to as a *vector representable matroid inner bound* $\Gamma^q_{N,N'}$. As $N' \to \infty$, $\Gamma^q_{N,\infty}$ is the conic hull of all ranks of subspaces on $\mathbb{F}_q$. The union over all field sizes for $\Gamma^q_{N,\infty}$ is the conic hull of the set of ranks of subspaces. Similarly, if a vector in the rate region of a network is (projection of) a vector $\mathbb{F}_q$-representable matroid rank, the representation $\mathbb{A}$ can be used as a linear code to achieve that rate vector and this code is denoted as a *vector* $\mathbb{F}_q$ *code*, as we will explain in next section.

Each of the bounds discussed in this section could be used in equation (3.15) and (3.16) to calculate bounds on the rate region for an MDCS instance A. If we substitute the Shannon outer bound Γ_N into (3.15), we get

$$\mathcal{R}_{\text{out}}(\mathsf{A}) = \text{Proj}_{R_e, e\in\mathcal{E}, H(X_k), k\in[[K]]}(\Gamma_N \cap \mathcal{L}_{125} \cap \mathcal{L}''_4). \tag{2.33}$$

Similarly, when representable matroid inner bound Γ^q_N and the vector representable matroid inner bound $\Gamma^q_{N,\infty}$ are substituted into (3.16), we get

$$\mathcal{R}_{s,q}(\mathsf{A}) = \text{Proj}_{R_e, e\in\mathcal{E}, H(X_k), k\in[[K]]}(\Gamma^q_N \cap \mathcal{L}_{125} \cap \mathcal{L}''_4), \tag{2.34}$$

$$\mathcal{R}_{q}(\mathsf{A}) = \text{Proj}_{R_e, e\in\mathcal{E}, H(X_k), k\in[[K]]}(\Gamma^q_{N,\infty} \cap \mathcal{L}_{125} \cap \mathcal{L}''_4). \tag{2.35}$$

If $\Gamma^q_{N,N'}$ is substituted in (3.16), the inner bound obtained on the rate region is denoted as $\mathcal{R}^{N,N'}_q(\mathsf{A})$.

2.3.3.6 Superposition Coding Rate Region

Another important inner bound for the rate region is the superposition coding rate region. In superposition coding, in each encoder, sources are coded independently from one another, and the output of each encoder is the concatenation of the separately coded messages across the different sources. Since sources are coded separately, the coding rate of each encoder is simply the sum of its separated coding rates for each source. The minimum separated coding rate for each source is determined by the max-flow min-cut bound [2], since this is equivalent to single-source multicast network coding. In particular, suppose a decoder D_d has input $\{U_e | e \in \text{Fan}(D_d)\}$ and recovers

$X_1, \ldots, X_k$, then we have

$$R_e = \sum_{i=1}^{K} R_e^{X_i}, e \in \mathrm{Fan}(D_d), \tag{2.36}$$

$$\sum_{e \in \mathrm{Fan}(D_d)} R_e^{X_i} \geq H(X_i), i = 1, \ldots, k. \tag{2.37}$$

Let $\mathbf{r} = (R_e, H(X_k), R_e^{X_k} | e \in \mathcal{E}, k \in [[K]])$ be the stacking of all of the involved variables in these inequalities. After considering all decoders in $\mathcal{D}$, we project the polyhedron defined by (2.36), (2.37) onto the dimensions of $R_e, H(X_k), e \in \mathcal{E}, k \in [[K]]$ to get the superposition coding rate region. That is, the superposition coding rate region $\mathcal{R}_{sp}(\mathsf{A})$ for a MDCS instance A is

$$\mathcal{R}_{sp}(\mathsf{A}) := \mathrm{Proj}_{R_e, H(X_k), \forall e, k} \left\{ \mathbf{r} \in \mathbb{R}_{\geq 0}^{|\mathcal{E}|+K+K|\mathcal{E}|} \;\middle|\; \begin{array}{c} R_e = \sum_{i=1}^{K} R_e^{X_i}, e \in \mathcal{E}, \\ \sum_{e \in \mathrm{Fan}(D_d)} R_e^{X_i} \geq H(X_i), i = 1, \ldots, \mathrm{Lev}(D_d), \forall d \in \mathcal{D}. \end{array} \right\}. \tag{2.38}$$

While equation (2.38) is the form of the superposition rate region most amenable to computation, we will also give a different but equivalent form for the ease of proving Theorem 7 and Theorem 8 in §3.6. First, we modify the decoding constraints $\mathcal{L}_5$ in equation (2.14) to be

$$\mathcal{L}_5' = \left\{ (\mathbf{h}, R_e^{X_k} | e \in \mathcal{E}, k \in [[K]]) \in \mathbb{R}_{\geq 0}^{|2^N-1+K|\mathcal{E}|} \;\middle|\; \begin{array}{c} \mathbf{h}_{U_e} = \sum_{i=1}^{K} R_e^{X_i}, e \in \mathcal{E}, \\ \sum_{e \in \mathrm{Fan}(D_d)} R_e^{X_i} \geq H(X_i), i = 1, \ldots, \mathrm{Lev}(D_d), \forall d \in \mathcal{D}. \end{array} \right\}. \tag{2.39}$$

Then, an alternate form of the superposition coding rate region is

$$\mathcal{R}_{sp}(\mathsf{A}) = \mathrm{Proj}_{R_e, e \in \mathcal{E}, H(X_k), k \in [[K]]} (\overline{\mathrm{con}(\Gamma_{\mathrm{N}}^* \cap \mathcal{L}_{12})} \cap \mathcal{L}_5' \cap \mathcal{L}_4''). \tag{2.40}$$

If the outer bound matches with any of the inner bounds presented above, the exact rate region $\mathcal{R}(\mathsf{A})$ has been obtained. We will compute the rate regions and some bounds on more than 7000 non-isomorphic MDCS instances in §2.6. But first, in the next section, we will show how to construct the codes achieving the points in the inner bounds presented in this section.

2.4 Constructing Linear Codes to Achieve the Inner Bounds

This section gives explicit constructions for codes achieving all the points in the inner bounds presented in the previous section. We will explain how a point in the rate region obtained from the representable matroid inner bound Γ_N^q can be achieved by linear codes which are constructed from the matrix representations of representable matroids. We will then explain how to extend this to vector codes associated with the vector inner bounds $\Gamma_{N,N'}^q$. Some examples will be shown for illustration.

We begin by observing that for any MDCS instance A, the scalar inner bound $\mathcal{R}_{s,q}(\mathsf{A})$ defined in (3.23) is a convex cone whose dimensions are associated with the variables $[h_{X_k}, R_e, k \in [[K]], e \in \mathcal{E}]$. For any polyhedral cone $\mathcal{P}$, let $\mathrm{Extr}(\mathcal{P})$ denote the set of representative vectors of its extreme rays.

Theorem 2: Let $\mathbf{R} \in \mathcal{R}_{s,q}$, there exists a $\mathbf{R}' \in \mathcal{R}_{s,q}$ with $R'_e \leq R_e, e \in \mathcal{E}$ and $H'(X_k) = H(X_k), k \in [[K]]$, and $\mathcal{T} \subseteq \mathrm{Extr}(\Gamma_N^q), \alpha_i \geq 0$ such that $\mathbf{R}' = \sum_{\mathbf{r}_i \in \mathcal{T}} \alpha_i \mathsf{Proj}_{H(X_k),H(U_e),k\in[[K]],e\in\mathcal{E}} \mathbf{r}_i$.

Proof: Let $\mathbf{R} \in \mathcal{R}_{s,q}$, from (3.16) we see that there exists a point $\mathbf{R}' \in \mathcal{R}_{s,q}$, and random variables $U_e, e \in \mathcal{E}$ with $R_e \geq R'_e = H(U_e), e \in \mathcal{E}$ and $H'(X_k) = H(X_k), k \in [[K]]$. (3.16) also shows that there exists a $\mathbf{h} \in \Gamma_N^q$ such that $\mathbf{R}' = \mathrm{Proj}_{H(X_k),H(U_e),k\in[[K]],e\in\mathcal{E}} \mathbf{h}$, and $\mathbf{h}$ satisfies the network constraints. Next observe that, all network constraints are Shannon-type inequalities set equal to zero. For instance, $\mathcal{L}_1$, representing the source independence constraint, is the Shannon inequality $\sum_{k=1}^K H(X_k) - H(\mathbf{X}_{[[K]]}) \geq 0$ set to zero. $\mathcal{L}_2$, representing the encoding constraints, is the Shannon inequalities $H(U_e|\mathbf{X}_{[[K]]]}) \geq 0, e \in \mathcal{E}$ set to zeros. $\mathcal{L}_5$, representing the decoding functions, is the Shannon inequalities $H(\mathbf{X}_{1:\mathrm{Lev}(D_d)}|\mathrm{In}(D_d)) \geq 0, d \in \mathcal{D}$ set to zeros. Since $\Gamma_N^q \subseteq \Gamma_N$ and all points in Γ_N^q must obey Shannon-type inequalities, no point in Γ_N^q can lie in the negative side of the network constraints, hence, $\mathcal{T} = \mathrm{Extr}(\Gamma_N^q \cap \mathcal{L}_{1,2,5}) \subseteq \mathrm{Extr}(\Gamma_N^q)$. Therefore, $\mathbf{h}$ can be expressed as a conic combination of extreme rays in $\mathcal{T} \subseteq \mathrm{Extr}(\Gamma_N^q)$, i.e., $\mathbf{h} = \sum_{\mathbf{r}_i \in \mathcal{T}} \alpha_i \mathbf{r}_i$ with $\alpha_i \geq 0$. Then

$$\begin{aligned}
\mathbf{R}' &= \mathrm{Proj}_{H(X_k),H(U_e),k\in[[K]],e\in\mathcal{E}} \mathbf{h} && (2.41)\\
&= \mathrm{Proj}_{H(X_k),H(U_e),k\in[[K]],e\in\mathcal{E}} \sum_{\mathbf{r}_i \in \mathcal{T}} \alpha_i \mathbf{r}_i && (2.42)\\
&= \sum_{\mathbf{r}_i \in \mathcal{T}} \alpha_i \mathrm{Proj}_{H(X_k),H(U_e),k\in[[K]],e\in\mathcal{E}} \mathbf{r}_i. && (2.43)
\end{aligned}$$

■

Before we show the construction of a code to achieve an arbitrary point in the rate region, it is necessary to show that a rank function of $\mathbb{F}_q$-representable matroid M is associated with a linear

network code in $\mathbb{F}_q$.

Given a particular MDCS instance, we first define a *network-$\mathbb{F}_q$-matroid mapping*, by loosening some conditions in [36], to be $f : \mathbf{X}_{1:K} \cup \{U_e, e \in \mathcal{E}\} \to \mathcal{S}'$, which associates each source variable and encoded message variable with a collection of elements forming one set in a partition $\mathcal{S}'$ of a ground set $\mathcal{S}$ of a $\mathbb{F}_q$-representable matroid M with rank function r_M, such that:

1. f is an one to one map;
2. $r_M(\bigcup_{k=1}^{K} f(X_k)) = \sum_{i=1}^{K} r_M(f(X_k))$,
3. $r_M(f(\mathrm{In}(v))) = r_M(f(\mathrm{In}(v)) \cup f(\mathrm{Out}(v))), \forall v \in \mathcal{E} \cup \mathcal{D}$, due to the encoder and decoder functions. Here, $\mathrm{In}(v)$ is a collection of input variables to v and $\mathrm{Out}(v)$ is a collection of output variables from v.

If all of the elements in $\mathcal{S}'$ are singletons, then f is an one-to-one mapping to $\mathcal{S}$, and the matrix representation of M can be used as linear code in $\mathbb{F}_q$ for this MDCS instance, since the mapping guarantees the MDCS network constraints are obeyed. Such a coding solution is called a *basic scalar solution*. If $\mathcal{S}'$ contains some elements that have cardinalities greater than 1, the representation of M is interpreted as a collection of bases of $|\mathcal{S}'|$ subspaces, which can also be used as a linear code and such a solution is called a *basic vector solution*.

We first construct the code for basic solutions, where entropies of network variables are ranks of associated elements in the matroid ($H_q(X_k) = r_M(f(X_k)), k \in [[K]]$ and $H(U_e) = r_M(f(U_e)), e \in \mathcal{E}$). In particular, there are $\sum_{k\in[[K]]} r_M(f(X_k))$ q-ary digits $\mathbf{X}_{k\in\mathcal{K}^1} r_M(f(X_k))$, where $\mathcal{K}^1 = \{k \in [[K]] | H(X_k) \neq 0\}$.

There exists a representation $\bar{\mathbb{C}}$ with dimension $(\sum_{k\in[[K]]} r_M(f(X_k))) \times (\sum_{k\in[[K]]} r_M(f(X_k)) + \sum_{e\in\mathcal{E}} r(f(U_e)))$ associated with the rank function r_M of M, where $\bar{\mathbb{C}} = [\mathbb{I}_{\sum_{k\in[[K]]} r_M(f(X_k))}\ \mathbb{C}]$ and the identity matrix $\mathbb{I}_{\sum_{k\in[[K]]} r_M(f(X_k))}$ is mapped to the source digits with non-zero entropies and the rest $\mathbb{C}$ is mapped to coded messages such that $U_e = \mathbf{X}_{k\in\mathcal{K}^1} \mathbb{C}_{:,I(U_e)}, e \in \mathcal{E}$, where $I(U_e)$ indicates the columns mapped to message U_e ($\mathbb{C}_{:,I(U_e)} = \mathbb{O}_{\sum_{k\in[[K]]} r_M(f(X_k))\times 1}$ if $H(U_e) = 0$). $\mathbb{C}$ is a *semi-simplified solution* by deleting rows which are associated with sources with zero entropy but keeping the column size as $\sum_{e\in\mathcal{E}} r_M(f(U_e))$. In the following context, basic solutions are semi-simplified.

Now let us consider a point $\mathbf{R} \in \mathcal{R}_q$ associated with source entropies $H_q(X_k), k \in [[K]]$. Suppose $\mathbf{R} = \sum_{\mathbf{r}_i\in\mathcal{T}} \alpha_i \mathrm{Proj}_{h_{X_k}, h_{U_e}, k\in[[K]], e\in\mathcal{E}} \mathbf{r}_i, \alpha_i \geq 0$, where $\mathcal{T} \subseteq \mathrm{Extr}(\Gamma_N^q)$, and for every $\mathbf{r}_i \in \mathcal{T}$, there exists an associated semi-simplified basic scalar solution $\mathbb{C}^{(i)}$ for the network.

Let $H_q(X_{k,i}), k \in [[K]]$ be the source entropies, $R_{e,i}, e \in \mathcal{E}, i = 1, \ldots, |\mathcal{T}|$ be the rates associated

with $\mathbf{r}_i$. According to Theorem 2, $H_q(X_k) = \sum_{i=1}^{|\mathcal{T}|} \alpha_i H_q(X_{k,i})$ and $R_e = \sum_{i=1}^{|\mathcal{T}|} \alpha_i R_{e,i}$, where $H_q(X_{k,i}) \in \{0,1\}$ and $R_{e,i} \in \{0,1\}$ because they are from matroids.

The construction of a code to achieve $\mathbf{R}$ is as follows.

1. Find rational numbers $\tilde{\alpha}_i = \frac{t_i}{n_i} \approx \alpha_i,\ i = 1, \ldots, |\mathcal{T}|$, then $\tilde{H}_q(X_k) = \sum_{i=1}^{|\mathcal{T}|} \tilde{\alpha}_i H_q(X_{k,i})$ and $\tilde{R}_e = \sum_{i=1}^{|\mathcal{T}|} \tilde{\alpha}_i R_{e,i}$ are the approximation, which can be arbitrarily close, of source entropies and rates, respectively;

2. Let $L = \mathrm{LCM}(\{n_i\})$ be the block length;

3. Suppose L blocks of all K source variables $\mathbf{X}_{1:K}^{1:L}$ are losslessly converted to uniformly distributed q-ary digits by some fix-length source code using a sufficiently large number of outer blocks. We gather these q-ary digits formed by individually compressing the original source variables into a row vector $\tilde{\mathbf{X}}$, $\mathrm{length}(\tilde{\mathbf{X}}) = L \sum_{k=1}^{K} \tilde{H}_q(X_k)$.

4. Let $\tilde{t}_i = L\tilde{\alpha}_i$ be the number of times we will use code $\mathbb{C}^{(i)}$. For every time we use $\mathbb{C}^{(i)}$, the number of q-ary digits encoded is equal to the number of rows in $\mathbb{C}^{(i)}$ (note that $\mathbb{C}^{(i)}$ is semi-simplified). So there exists a partition of $\tilde{\mathbf{X}}$ consisting of $\sum_{i=1}^{|\mathcal{T}|} \tilde{t}_i$ elements in total and all $\tilde{t}_i$ elements mapped with $\mathbb{C}^{(i)}$ have the same cardinality which is the number of rows in $\mathbb{C}^{(i)}, \forall i = 1, \ldots, |\mathcal{T}|$. More specifically, we are drawing $\tilde{t}_i H_q(X_{k,i})$ samples from X_k's buffer for the $\tilde{t}_i$ repetitions of the basic solution $\mathbb{C}^{(i)}$.

5. Let $\tilde{\mathbf{X}}' = \tilde{\mathbf{X}}\mathbb{G}$ ($\mathbb{G}$ is a shuffled identity matrix to relocate the q-ary digits in $\tilde{\mathbf{X}}$) be a rearrangement of $\tilde{\mathbf{X}}$ such that the source digits are mapped in the same order as the basic solutions in the constructed code $\tilde{\mathbb{C}}$ which repeats $\mathbb{C}^{(i)}$ for $\tilde{t}_i$ times, $\forall i \in \{1, 2, \ldots, |\mathcal{T}|\}$ in the way as follows.

$$\tilde{\mathbf{U}} = \tilde{\mathbf{X}}' \times \mathrm{BlkDiag}(\underbrace{\mathbb{C}^{(1)}, \ldots}_{\tilde{t}_1 \text{ times}}, \underbrace{\mathbb{C}^{(i)}, \ldots}_{\tilde{t}_i \text{ times}}, \underbrace{\mathbb{C}^{(m)}, \ldots}_{\tilde{t}_m \text{ times}}) \tag{2.44}$$

   where $\mathrm{BlkDiag}(\cdot)$ is a block diagonalizing function.

6. Note that all $\mathbb{C}^{(i)}$ have the same column size and the column indices are mapped to $e \in \mathcal{E}$. Therefore, we can rearrange the columns in $\tilde{\mathbb{C}}$ to group all columns containing $\mathbb{C}^{i}_{:,I_i(U_e)}, i = 1, \ldots, |\mathcal{T}|$ to be an encoding function for e. That is, $\mathbb{C} = \mathrm{concatenation}(\mathbb{C}_{:,I(U_e)})$, $\mathbb{C}_{:,I(U_e)} = \tilde{\mathbb{C}}_{:,I\left(e+|\mathcal{E}|\cdot\left[0:\sum_{i=1}^{|\mathcal{T}|}\tilde{t}_i-1\right]\right)}$. $\mathbb{C}$ can be further simplified by deleting all-zero columns.

Indeed, we can see that the code constructed this way can achieve the point $\mathbf{R} \in \mathcal{R}_q$ by examining

$$\begin{aligned} H_q(\tilde{U}_e) &= \text{rank}(\mathbb{C}_{:,I(U_e)}) && (2.45)\\ &= \sum_{i=1}^{|\mathcal{T}|} \tilde{t}_i \text{rank}(\mathbb{C}^{(i)}_{:,I(U_e)}) && (2.46)\\ &= \sum_{i=1}^{|\mathcal{T}|} \tilde{t}_i R_{i,e} && (2.47)\\ &= L \sum_{i=1}^{|\mathcal{T}|} \tilde{\alpha}_i R_{i,e} && (2.48)\\ &= L\tilde{R}_e. && (2.49) \end{aligned}$$

Therefore, the actual rate per source variable is

$$\hat{R}_e = \frac{H_q(\tilde{U}_e)}{L} = \tilde{R}_e \approx R_e, \tag{2.50}$$

with arbitrarily small offset if the fraction approximations are arbitrarily close. If Γ_N^q is used in obtaining the rate region, $\mathbb{C}^{(i)}, \forall i = 1, \ldots, |\mathcal{T}|$ are basic scalar solution, we call the constructed code a *scalar representation solution*. Similarly, if $\Gamma^q_{N,N'}$ is used in obtaining the rate region, some basic vector solutions $\mathbb{C}^{(i)}$, some $i = 1, \ldots, |\mathcal{T}|$ may be needed in constructing the code $\mathbb{C}$. We call such a code involving basic vector solution(s) a *vector representation solution*.

Example 1: Consider a 2-level-3-encoder MDCS instance, shown in Fig. 2.3.

There are two sources X, Y, three encoders E_1, E_2, E_3 with corresponding coded message variables $U_i, i = 1, 2, 3$ and rate constraints $R_i, i = 1, 2, 3$, four decoders classified into two levels with access to encoders as shown in the following table. Level 1 means this decoder recovers X, while level 2 means it recovers X, Y.

level 1	level 2
{(1)}	{(1, 2), (1, 3), (2, 3)}

The outer $\mathcal{R}_{\text{out}}$ and scalar binary inner bound $\mathcal{R}_{s,2}$ on rate region obtained from Shannon outer

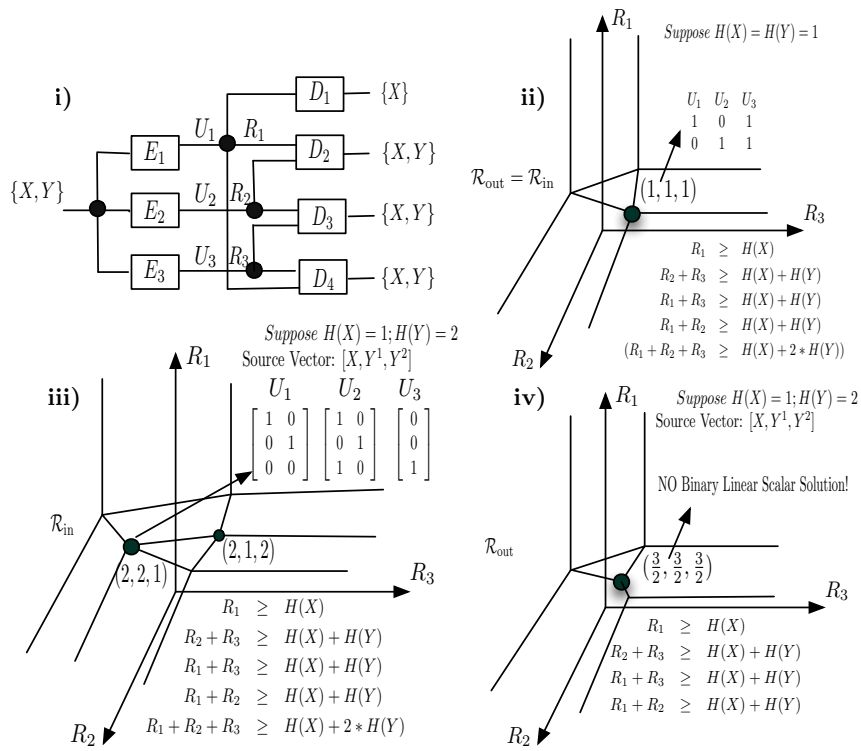


Figure 2.3: ***i)*** *Example MDCS instance.* ***ii)*** *Rate region and corresponding codes.* ***iii)*** *Scalar codes for inner bound.* ***iv)*** *No scalar code for outer bound.*

bound and binary inner bound are:

$$\begin{aligned}
\mathcal{R}_{\text{out}} &= \left\{ \mathbf{R} : \begin{array}{rcl} R_1 & \geq & H(X) \\ R_2 + R_3 & \geq & H(X) + H(Y) \\ R_1 + R_3 & \geq & H(X) + H(Y) \\ R_1 + R_2 & \geq & H(X) + H(Y) \end{array} \right\} \\
\mathcal{R}_{s,2} &= \mathcal{R}_{\text{out}} \cap \{\mathbf{R} : R_1 + R_2 + R_3 \geq H(X) + 2H(Y)\}
\end{aligned} \tag{2.51}$$

where $\mathbf{R} = (R_1, R_2, R_3)$. Note that when $H(X) = H(Y)$, $\mathcal{R}_{\text{out}} = \mathcal{R}_{s,2}$. This is a case where the inner and outer bounds match, and thus we are able to find a coding solution for this network. The scalar binary code

$$U_1 U_2 U_3 = XY \times \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \tag{2.52}$$

achieves the extreme point $(1, 1, 1)$ in $\mathcal{R}$ when $H(X) = H(Y) = 1$, as shown in Fig. 2.3.

If $H(X) = H(Y) = h$, the corresponding extreme point (h, h, h) will be achieved by repetition of

the basic solution:

$$U_1U_2U_3 = X^1 \dots X^h Y^1 \dots Y^h \times \begin{bmatrix} \mathbb{I}_h & \vdots & \mathbb{O}_h & \vdots & \mathbb{I}_h \\ \mathbb{O}_h & \vdots & \mathbb{I}_h & \vdots & \mathbb{I}_h \end{bmatrix}. \tag{2.53}$$

If h is not integer, we can easily approximate it with arbitrarily precision using fractions $\frac{t}{n} \approx h$ and then decide the block size as n to construct the block code by repeating the above basic solution for t times in block diagonal manner. Then, on average we will achieve (h, h, h).

However, $\mathcal{R}_{\text{out}} \neq \mathcal{R}_{s,2}$ in general for this example, if $H(X) \neq H(Y)$. For $H(X) = 1, H(Y) = 2$ there is a gap between the inner and outer bounds. For the inner bound, we can find a scalar code solution. Fig. 2.3 shows a binary code to achieve the inner bound extreme point $\mathbf{R} = (2, 2, 1)$, which is a conic combination of two basic solutions. Let $H(U_1) = R_1, H(U_2) = R_2, H(U_3) = R_3$, the point $(H(X), H(Y), H(U_1), H(U_2), H(U_3)) = (1, 2, 2, 2, 1) = (1, 1, 1, 1, 1) + (0, 1, 1, 1, 0)$. The solution corresponding to $(0, 1, 1, 1, 0)$ is

$$U_1^1U_2^1U_3^1 = Y^1 \times [1\ 1\ 0] \tag{2.54}$$

and the solution corresponding to $(1, 1, 1, 1, 1)$ is

$$U_1^2U_2^2U_3^2 = XY^2 \times \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}. \tag{2.55}$$

The final scalar representation solution with shuffling of columns is shown in Fig. 2.3.

For $\mathcal{R}_{\text{out}}$, we know there does not exist scalar binary coding solution for some extreme point. For example, as shown in Fig. 2.3, there is no scalar solution for the outer bound extreme point $(\frac{3}{2}, \frac{3}{2}, \frac{3}{2})$. However, $\Gamma_{5,6}^2$ makes up the gap and we know there must exist a solution to achieve this point. Actually, we can find a binary vector representation solution for this point. Note that we only need to group two outcomes of source variables and encode them together. Suppose we have source vector $\mathbf{v} = [X_1, X_2, Y_1^1, Y_1^2, Y_2^1, Y_2^2]$ where the lower index indicates two outcomes in time while upper index indicates

the position in one outcome. One vector representation coding solution (with columns shuffled) is

$$U_1U_2U_3 = \mathbf{v} \times \left[\begin{array}{ccc|ccc|ccc} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \end{array}\right], \tag{2.56}$$

which can also be expressed as a conic combination of two basic solutions. Let $H(U_1) = R_1, H(U_2) = R_2, H(U_3) = R_3$, the point $2 \times (H(X), H(Y), H(U_1), H(U_2), H(U_3)) = (2, 4, 3, 3, 3) = (1, 1, 1, 1, 1) + (1, 3, 2, 2, 2)$. The solution corresponding to $(1, 1, 1, 1, 1)$ is

$$U_1^1U_2^1U_3^1 = X_1Y_2^1 \times \left[\begin{array}{ccc} 1 & 1 & 0 \\ 0 & 1 & 1 \end{array}\right], \tag{2.57}$$

and the solution corresponding to $(1, 3, 2, 2, 2)$ is

$$U_1^2U_2^2U_3^2 = X_2Y_1^1Y_1^2Y_2^2 \times \left[\begin{array}{cc|cc|cc} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \end{array}\right]. \tag{2.58}$$

Having provided code constructions in these examples, we now pass to investigating embedding operations for smaller MDCS instances into larger MDCS instances such that the larger MDCS instances inherits the insufficiency of a class of codes from the smaller MDCS instances.

2.5 Embedded MDCS Instances and the Preservation of Coding Class Sufficiency

In [1], a definition of *embedded MDCS* instances was given for $(2, 3)$ and $(3, 3)$ MDCS instances where a $(2, 3)$ MDCS instance A is embedded in a $(3, 3)$ MDCS instance A' if it can be obtained by deleting one source variable in A', as we will define in Definition 14. We would like to extend the definition of embedded MDCS instances, because we are interested in the relationships between different $(K, |\mathcal{E}|)$ MDCS problems with respect to sufficiency of certain linear codes, as will be shown in §2.6. We would like to show that the insufficiency of certain classes of codes will be preserved when one extends a smaller MDCS instance to a bigger one. For that, we first define some operations

on MDCS instances that can obtain a smaller MDCS instance from a bigger one.

2.5.1 Embedding Operation Definitions

We generalize the definition of source deletion first. When a source is deleted, the decoders that demand it will no longer demand it after deletion.

Definition 6 (Source Deletion $\mathsf{A}\backslash X_k$)**:** Suppose a MDCS instance $\mathsf{A} = (\{X_1, \ldots, X_K\}, \mathcal{E}, \mathcal{D}, \mathbf{L}', \mathcal{G}')$. When a source X_k is deleted, denoted as $\mathsf{A}\backslash X_k$, in the new MDCS instance $\mathsf{A}' = (\{X_1, \ldots, X_K\} \setminus X_k, \mathcal{E}', \mathcal{D}', \mathbf{L}', \mathcal{G}')$, we will have:

1. $\mathcal{E}' = \mathcal{E}$;
2. $\mathcal{D}' = \mathcal{D} \setminus \{D_j | \exists D_i \in \mathcal{D}, i \neq j, \mathrm{Out}_{\mathsf{A}}(D_i) = \mathrm{Out}_{\mathsf{A}}(D_j) \setminus X_k,\ \mathrm{Fan}_{\mathsf{A}}(D_i) \subseteq \mathrm{Fan}_{\mathsf{A}}(D_j)\}$;
3. For $\mathbf{L}'$, $\mathrm{Lev}_{\mathsf{A}'}(D_d) = \mathrm{Lev}_{\mathsf{A}}(D_d) - 1, \forall D_d$ such that $X_k \in \mathrm{Out}_{\mathsf{A}}(D_d)$;
4. $\mathcal{G}' = \{(E_i, D_d) | E_i \in \mathcal{E}', D_d \in \mathcal{D}', (E_i, D_d) \in \mathcal{G}\}$.

This is straightforward because the deletion of a source just changes the decoding requirements of decoders. Fig. 2.4(a) demonstrates the deletion of a source. When source Z is deleted, D_5 will no longer require Z and thus becomes a level-2 decoder. However, since D_2 only has access to E_1, E_2 but is also a level-2 decoder, D_5' becomes redundant and is deleted.

Next, we consider the operation of contracting an encoder. When an encoder is contracted, all of the decoders in its fan will be deleted, as well as all the edges associated with the contracted decoders.

Definition 7 (Encoder Contraction (A/E_e))**:** Suppose a MDCS instance $\mathsf{A} = (\{X_1, \ldots, X_K\}, \mathcal{E}, \mathcal{D}, \mathbf{L}, \mathcal{G})$. A smaller MDCS instance $\mathsf{A}' = (\{X_1, \ldots, X_K\}, \mathcal{E}', \mathcal{D}', \mathbf{L}', \mathcal{G}')$ is obtained by contracting $E_e, e \in \mathcal{E}$, denoted by A/E_e, if:

1. $\mathcal{E}' = \mathcal{E} \setminus E_e$;
2. $\mathcal{D}' = \mathcal{D} \setminus \{D_d | D_d \in \mathrm{Fan}_{\mathsf{A}}(E_e)\}$;
3. For $\mathbf{L}'$, $\mathrm{Lev}_{\mathsf{A}'}(D_i) = \mathrm{Lev}_{\mathsf{A}}(D_i), \forall D_i \in \mathcal{D}'$;
4. $\mathcal{G}' = \mathcal{G} \setminus \{(E_i, D_d) | E_i \in \mathcal{E}, D_d \in \mathrm{Fan}_{\mathsf{A}}(E_e), (E_i, D_d) \in \mathcal{G}\}$.

This operation assumes that when an encoder is contracted, its fan will directly have access to its input, all the sources, which makes the decoding requirements obviously satisfied. Fig. 2.4(b)

demonstrates the contraction of an encoder. As it shows, when encoder E_4 is contracted, all decoders which have access to E_4, i.e., fan of E_4, become redundant and are deleted.

Next, we define deletion of an encoder as follows.

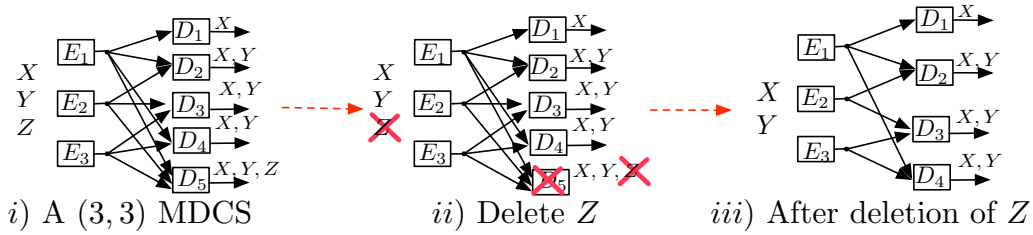
Definition 8 (Encoder Deletion ($\mathsf{A}\backslash E_e$))**:** Suppose a MDCS instance $\mathsf{A} = (\{X_1, \ldots, X_K\}, \mathcal{E}, \mathcal{D}, \mathbf{L}, \mathcal{G})$. When an encoder $E_e \in \mathcal{E}$ is deleted, denoted as $\mathsf{A}\backslash E_e$, in the new MDCS instance $\mathsf{A}' = (\{X_1, \ldots, X_K\}, \mathcal{E}', \mathcal{D}', \mathbf{L}', \mathcal{G}')$, we will have:

1. $\mathcal{E}' = \mathcal{E} \setminus E_e$;
2. $\mathcal{D}' = \mathcal{D}\backslash\{D_j \in \mathcal{D} | \exists D_i \in \mathrm{Fan}_{\mathsf{A}}(E_e)\backslash D_j, \mathrm{Fan}_{\mathsf{A}}(D_i)\backslash E_e \subseteq \mathrm{Fan}_{\mathsf{A}}(D_j)$ and $\mathrm{Lev}_{\mathsf{A}}(D_j) \leq \mathrm{Lev}_{\mathsf{A}}(D_i)\}$;
3. For $\mathbf{L}'$, $\mathrm{Lev}_{\mathsf{A}'}(D_d) = \mathrm{Lev}_{\mathsf{A}}(D_d), \forall D_d \in \mathcal{D}'$;
4. $\mathcal{G}' = \{(E_i, D_d) | E_i \in \mathcal{E}', D_d \in \mathcal{D}', (E_i, D_d) \in \mathcal{G}\}$.

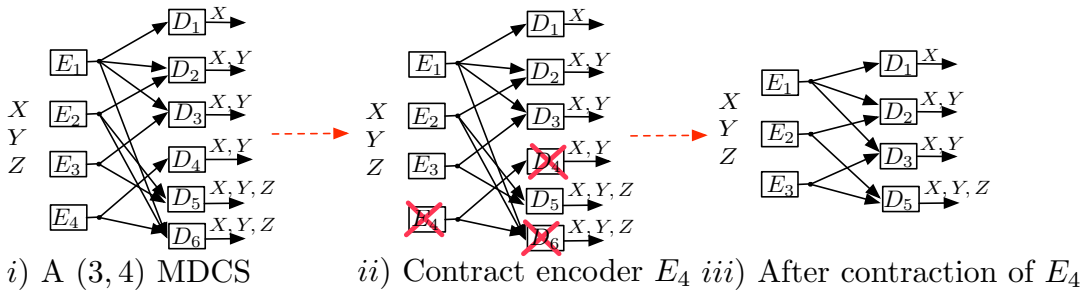
The essence of this definition is to keep the dependence relationship between input and output of the decoders when an encoder is deleted. In other words, when an encoder is deleted, the decoders that have access to it should function as before. Note that, if there exists a decoder D_i that only has access to E_e, after the deletion of E_e, it has access to no encoders but needs to keep the same decoding capability, which means the sources $X_1, \ldots, X_{\mathrm{Lev}(D_i)}$ will also be deleted.

Fig. 2.4(c) demonstrates the deletion of an encoder. When encoder E_4 is deleted, D_6 no longer has access to E_4 but still has access to E_1, E_2. Note that, since D_2 also has access to E_1, E_2 but is a level-2 decoder, D_2 becomes redundant and is deleted.

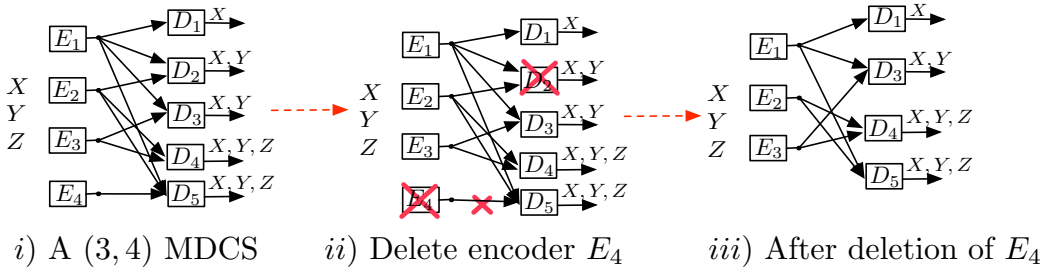
When the reverse operation of encoder contraction is considered, a MDCS instance can be extended by adding a new encoder with some new decoders that must talk with the new encoder obeying (**C1**)–(**C5**). If some class of codes does not suffice in the smaller network, it will not suffice in the bigger one either. Similarly, the insufficiency can be preserved by considering to extend a smaller MDCS instance by adding some arbitrary redundant encoder and decoders, which is the reverse of encoder deletion. There exists some other ways to preserve the non-sufficiency of certain class of codes. For instance, if a class of codes does not suffice for a smaller network, there does not exist a construction of codes for at least one encoder to satisfy all the network constraints. If a new encoder dependent on that encoder and some other redundant decoding requirements are constructed obeying the conditions (**C1**)–(**C5**), that class codes still cannot be sufficient for the new network. We define another operation based on this intuition as follows.



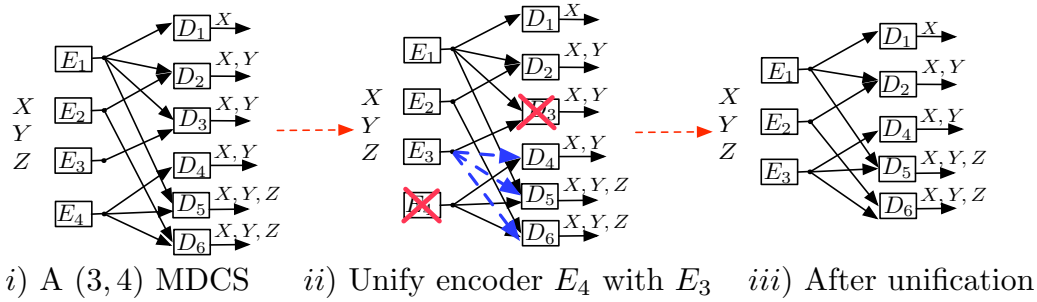
(a) Demonstration of source deletion: when source Z is deleted, decoders that previously required it will no longer require it.



(b) Demonstration of encoder contraction: when E_4 is contracted, the fan of it will be deleted.



(c) Demonstration of encoder deletion: when E_4 is deleted, the fan of it will keep the same decoding abilities and lower-level decoders will be superseded if they have same fan as the fan of E_4 after deletion.



(d) Demonstration of encoder unification: when E_4 is unified with $E_3 = E_4$, the fan of E_4 will also have access to E_3 after E_4 is removed. If conflicts occur, existing decoders become redundant. For instance, D_3 is superseded because D_4 only has access to E_3 but is able to decode X, Y.

Figure 2.4: Demonstration of operations on MDCS instances

Definition 9 (Encoder Unification ($\mathsf{A} \cap \{E_i = E_j\}$)**:** Suppose a MDCS instance $\mathsf{A} = (\{X_1, \ldots, X_K\}, \mathcal{E}, \mathcal{D}, \mathbf{L}, \mathcal{G})$. When an encoder $E_j \in \mathcal{E}$ is unified with $E_i = E_j, i \neq j$, denoted as $\mathsf{A} \cap \{E_i = E_j\}$, in the new MDCS instance $\mathsf{A}' = (\{X_1, \ldots, X_K\}, \mathcal{E}', \mathcal{D}', \mathbf{L}', \mathcal{G}')$, we will have:

1. $\mathcal{E}' = \mathcal{E} \setminus E_j$;
2. $\mathcal{D}' = \mathcal{D} \setminus \{D_l | \exists D_k \in \mathcal{D}, k \neq l, D_k \in \mathrm{Fan}_{\mathsf{A}}(E_j), D_l \in \mathrm{Fan}_{\mathsf{A}}(E_i), (\mathrm{Fan}_{\mathsf{A}}(D_k) \setminus \{E_i, E_j\}) \subseteq (\mathrm{Fan}_{\mathsf{A}}(D_l) \setminus \{E_i, E_j\})$ and $\mathrm{Lev}_{\mathsf{A}}(D_l) \leq \mathrm{Lev}_{\mathsf{A}}(D_k)\}$;
3. For $\mathbf{L}'$, $\mathrm{Lev}_{\mathsf{A}'}(D_d) = \mathrm{Lev}_{\mathsf{A}}(D_d), \forall D_d \in \mathcal{D}'$;
4. $\mathcal{G}' = \{(E_e, D_d) | E_e \in \mathcal{E}', D_d \in \mathcal{D}', (E_e, D_d) \in \mathcal{G}\} \cup \{(E_i, D_d) | D_d \in \mathrm{Fan}_{\mathsf{A}}(E_j) \cap \mathcal{D}'\}$.

After the unification of E_j, all decoders who have access to E_j will have access to E_i. Fig. 2.4(d) demonstrates the unification of an encoder. As it shows, when encoder E_4 is unified with $E_3 = E_4$, all decoders which have access to E_4, i.e., fan of E_4, will have access to E_3. Decoder D_3 becomes redundant because if D_4, D_5 are given access to E_3 instead of E_4, they both will supersede D_3.

2.5.2 Operation Order does not Matter

Next, we consider the order of different operations. It is not difficult to see that if a collection of sources are deleted, it does not matter which source is deleted first. Similarly, if a collection of encoders are contracted, deleted or unified, the operation order on different elements does not matter. Next we would like to show that different orders of operations give equivalent results.

Theorem 3: Let $\mathcal{O}_1, \mathcal{O}_2$ be two different operations on different elements, among the four operations defined in Definition 14– Definition 9. Applying $\mathcal{O}_1$ first and $\mathcal{O}_2$ second on a MDCS instance A is equivalent to applying $\mathcal{O}_2$ first and $\mathcal{O}_1$ second. That is, $\mathsf{A} \cap \mathcal{O}_1 \cap \mathcal{O}_2 = \mathsf{A} \cap \mathcal{O}_2 \cap \mathcal{O}_1$.

Proof: We need to consider the $\binom{4}{2} = 6$ combinations of operations.

Encoder deletion and encoder contraction: Let $\mathsf{A}' = (\mathsf{A} \backslash E_1)/E_2$ and $\mathsf{A}'' = (\mathsf{A}/E_2) \backslash E_1$. We need to show $\mathsf{A}' = \mathsf{A}''$. If E_1 is deleted at first, we have that $\forall D_i \in \mathrm{Fan}(E_1)$, the decoder D_i will have access to encoders in $\mathrm{Fan}(D_i) \setminus E_1$ and $\mathrm{Lev}_{\mathsf{A}'}(D_i) = \mathrm{Lev}_{\mathsf{A}}(D_i)$. If $\exists D_j$ such that $\mathrm{Fan}_{\mathsf{A}}(D_j) = \mathrm{Fan}_{\mathsf{A}}(D_i) \setminus E_1$ and $\mathrm{Lev}_{\mathsf{A}}(D_j) < \mathrm{Lev}_{\mathsf{A}}(D_i)$, $D_j \notin \mathsf{A}'$. We only need to consider the case when E_2 is a fan of D_i and/or D_j. If $E_2 \in \mathrm{Fan}_{\mathsf{A}}(D_i)$ and $E_2 \in \mathrm{Fan}_{\mathsf{A}}(D_j)$, no matter which operation is first, both D_i and D_j are gone. If $E_2 \in \mathrm{Fan}_{\mathsf{A}}(D_j)$, no matter deletion or contraction is done first, D_j will be gone. If $E_2 \in \mathrm{Fan}_{\mathsf{A}}(D_i)$, we must have $E_2 \in \mathrm{Fan}_{\mathsf{A}}(D_j)$ since $\mathrm{Fan}_{\mathsf{A}}(D_j) = \mathrm{Fan}_{\mathsf{A}}(D_i) \setminus E_1$ and we assume that $E_1 \neq E_2$. Thus, $\mathsf{A}' = \mathsf{A}''$.

Encoder deletion and source deletion: Let $\mathsf{A}' = (\mathsf{A}\backslash E_1)\backslash X_k$ and $\mathsf{A}'' = (\mathsf{A}\backslash X_k)\backslash E_1$. We need to show $\mathsf{A}' = \mathsf{A}''$. If E_1 is deleted first, we have that $\forall D_i \in \mathrm{Fan}_{\mathsf{A}}(E_1)$, $\mathrm{Fan}_{\mathsf{A}'}(D_i) = \mathrm{Fan}_{\mathsf{A}}(D_i) \setminus E_1$ and $\mathrm{Lev}_{\mathsf{A}'}(D_i) = \mathrm{Lev}(D_i)$. If $\exists D_j$ such that $\mathrm{Fan}_{\mathsf{A}}(D_j) = \mathrm{Fan}_{\mathsf{A}}(D_i) \setminus E_1$ and $\mathrm{Lev}_{\mathsf{A}}(D_j) < \mathrm{Lev}_{\mathsf{A}}(D_i)$, $D_j \notin \mathsf{A}'$. We only need to consider the case when X_k is an output of D_i and/or D_j. If $X_k \in \mathrm{Out}_{\mathsf{A}}(D_i)$ and $X_k \in \mathrm{Out}_{\mathsf{A}}(D_j)$, the deletion of X_k does not affect the deletion of E_1. Hence the order does not matter. If $X_k \in \mathrm{Out}_{\mathsf{A}}(D_i)$, no matter which operation is done first, D_j will be gone. If $X_k \in \mathrm{Out}_{\mathsf{A}}(D_j)$, we must have $X_k \in \mathrm{Out}_{\mathsf{A}}(D_i)$ since $\mathrm{Out}_{\mathsf{A}}(D_j) \subseteq \mathrm{Out}_{\mathsf{A}}(D_i)$. Thus, $\mathsf{A}' = \mathsf{A}''$.

Encoder contraction and source deletion: Let $\mathsf{A}' = (\mathsf{A}\backslash X_k)/E_1$ and $\mathsf{A}'' = (\mathsf{A}/E_1)\backslash X_k$. We need to show $\mathsf{A}' = \mathsf{A}''$. If X_k is deleted at first, we have that $\forall D_i$ such that $X_k \in \mathrm{Out}_{\mathsf{A}}(D_i)$, $\mathrm{Out}_{\mathsf{A}'}(D_i) = \mathrm{Out}(D_i) \setminus X$ and if $\exists D_j$ such that $\mathrm{Fan}_{\mathsf{A}}(D_j) \subseteq \mathrm{Fan}_{\mathsf{A}}(D_i)$ and $\mathrm{Out}_{\mathsf{A}}(D_j) = \mathrm{Out}_{\mathsf{A}'}(D_i)$, $D_i \notin \mathsf{A}'$. We only need to consider the case when E_1 is a fan of D_i and/or D_j. If $E_1 \in \mathrm{Fan}_{\mathsf{A}}(D_i)$ and $E_1 \in \mathrm{Fan}_{\mathsf{A}}(D_j)$, the contraction of E_1 will make both encoders gone no matter which operation is first. If $E_1 \in \mathrm{Fan}_{\mathsf{A}}(D_i)$ only, no matter which operation is done first, D_i will be gone. If $E_1 \in \mathrm{Fan}_{\mathsf{A}}(D_j)$, we must have $E_1 \in \mathrm{Fan}_{\mathsf{A}}(D_i)$ since $\mathrm{Fan}_{\mathsf{A}}(D_j) \subseteq \mathrm{Fan}_{\mathsf{A}}(D_i)$. Thus, $\mathsf{A}' = \mathsf{A}''$.

Encoder unification and encoder contraction: Let $\mathsf{A}' = (\mathsf{A} \cap \{E_2 = E_3\})/E_1$ and $\mathsf{A}'' = (\mathsf{A}/E_1) \cap \{E_2 = E_3\}$. We need to show $\mathsf{A}' = \mathsf{A}''$. In unification of E_3 with $E_2 = E_3$, all decoders having access to E_3 will also have access to E_2 and then E_3 is removed. A decoder D_l is deleted if $\exists D_k, D_l \in \mathcal{D}, k \neq l$, such that $D_k \in \mathrm{Fan}_{\mathsf{A}}(E_3)$, $D_l \in \mathrm{Fan}_{\mathsf{A}}(E_2)$, $\mathrm{Fan}_{\mathsf{A}}(D_k) \setminus \{E_2, E_3\} \subseteq \mathrm{Fan}(D_l)_{\mathsf{A}} \setminus \{E_2, E_3\}$ and $\mathrm{Lev}_{\mathsf{A}}(D_l) \leq \mathrm{Lev}_{\mathsf{A}}(D_k)$. If $E_1 \in \mathrm{Fan}_{\mathsf{A}}(D_l)$, clearly contraction of E_1 will also delete D_l. If $E_1 \in \mathrm{Fan}_{\mathsf{A}}(D_k)$, we will also have $E_1 \in \mathrm{Fan}_{\mathsf{A}}(D_l)$ since $\mathrm{Fan}_{\mathsf{A}}(D_k) \setminus \{E_2, E_3\} \subseteq \mathrm{Fan}_{\mathsf{A}}(D_l) \setminus \{E_2, E_3\}$. No matter E_1 is contracted first or not, the resulting MDCS instance will be the same. That is, $\mathsf{A}' = \mathsf{A}''$.

Encoder unification and encoder deletion: Let $\mathsf{A}' = (\mathsf{A} \cap \{E_2 = E_3\})\backslash E_1$ and $\mathsf{A}'' = (\mathsf{A}\backslash E_1) \cap \{E_2 = E_3\}$. We need to show $\mathsf{A}' = \mathsf{A}''$. In unification of E_3 with $E_2 = E_3$, all decoders having access to E_3 will also have access to E_2 and then E_3 is removed. A decoder D_l is deleted if $\exists D_k, D_l \in \mathcal{D}, k \neq l$, such that $D_k \in \mathrm{Fan}_{\mathsf{A}}(E_3)$, $D_l \in \mathrm{Fan}_{\mathsf{A}}(E_2)$, $\mathrm{Fan}_{\mathsf{A}}(D_k) \setminus \{E_2, E_3\} \subseteq \mathrm{Fan}(D_l)_{\mathsf{A}} \setminus \{E_2, E_3\}$ and $\mathrm{Lev}_{\mathsf{A}}(D_l) \leq \mathrm{Lev}_{\mathsf{A}}(D_k)$. If $E_1 \in \mathrm{Fan}_{\mathsf{A}}(D_l)$, we can see that when E_1 is deleted first, even though D_l could survive after deleting E_1, it will also be deleted after the unification of E_3 because we have $\mathrm{Fan}_{\mathsf{A}}(D_k) \setminus \{E_2, E_3\} \subseteq \mathrm{Fan}_{\mathsf{A}}(D_l) \setminus \{E_1, E_2, E_3\}$. If $E_1 \in \mathrm{Fan}_{\mathsf{A}}(D_k)$, we will also have $E_1 \in \mathrm{Fan}_{\mathsf{A}}(D_l)$ since $\mathrm{Fan}_{\mathsf{A}}(D_k) \setminus \{E_2, E_3\} \subseteq \mathrm{Fan}_{\mathsf{A}}(D_l) \setminus \{E_2, E_3\}$. No matter E_1 is deleted first or not, the resulting MDCS instance will be the same. That is, $\mathsf{A}' = \mathsf{A}''$.

Encoder unification and source deletion: Let $\mathsf{A}' = (\mathsf{A}\backslash X_k) \cap \{E_1 = E_2\}$ and $\mathsf{A}'' = (\mathsf{A} \cap \{E_1 = E_2\})\backslash X_k$. We need to show $\mathsf{A}' = \mathsf{A}''$. If X_k is deleted at first, we have that $\forall D_i$ such that

$X_k \in \mathrm{Out}_{\mathsf{A}}(D_i)$, $\mathrm{Out}_{\mathsf{A}'}(D_i) = \mathrm{Out}_{\mathsf{A}}(D_i) \setminus X_k$ and if $\exists D_j$ such that $\mathrm{Fan}_{\mathsf{A}}(D_j) \subseteq \mathrm{Fan}_{\mathsf{A}}(D_i)$ and $\mathrm{Out}_{\mathsf{A}}(D_j) = \mathrm{Out}_{\mathsf{A}}(D_i)$, $D_i \notin \mathsf{A}'$. We only need to consider the case when E_2 is a fan of D_i and/or D_j. If $E_2 \in \mathrm{Fan}_{\mathsf{A}}(D_i)$ and $E_2 \in \mathrm{Fan}_{\mathsf{A}}(D_j)$ and the unification of E_2 is conducted first, D_i, D_j will have access to E_1. This unification does not affect the subset relationship between $\mathrm{Fan}_{\mathsf{A}}(D_j)$ and $\mathrm{Fan}_{\mathsf{A}}(D_i)$. Thus, D_j is deleted no matter which operation is conducted first. Similarly, if $E_2 \in \mathrm{Fan}_{\mathsf{A}}(D_i)$ only or $E_2 \in \mathrm{Fan}_{\mathsf{A}}(D_i)$ only, no matter which operation is done first, D_j will be gone because the unification does not affect the conditions of deleting D_j. Thus, $\mathsf{A}' = \mathsf{A}''$. ■

Based on these operations and Theorem 1, we can define an *embedded* MDCS instance.

Definition 10 (Embedded MDCS instances)**:** An MDCS instance A' is said *embedded* in MDCS instance A, i.e., A' is a *minor* of A, denoted as $\mathsf{A}' \prec \mathsf{A}$, if A' can be obtained by a series of operations of source deletion, encoder deletion/ contraction/ unification on A. Equivalently, we say that A is an extension of A', denoted as $\mathsf{A} \succ \mathsf{A}'$.

Note that, as will be discussed in §2.6, we consider all four operations for the preservation of the insufficiency of $\mathbb{F}_q$ vector linear codes (§2.4) and the embedding relationship is denoted as $\mathsf{A}' \prec_v \mathsf{A}$ or $\mathsf{A} \succ_v \mathsf{A}'$. When insufficiency of scalar linear codes (superposition coding) are considered, the encoder unification is not considered and the embedding relationship is denoted as $\mathsf{A}' \prec_s \mathsf{A}$ ($\mathsf{A}' \prec_{sp} \mathsf{A}$) or $\mathsf{A} \succ_s \mathsf{A}'$ ($\mathsf{A} \succ_{sp} \mathsf{A}'$). Fig. 2.4 demonstrates different operations and thus shows four examples of embedded MDCS instances.

2.5.3 Inheritance of Code Class Sufficiency under Extension

Recall the definitions of deletion/contraction of encoders/sources in §2.5.1, which connect MDCS instances for different $(K, |\mathcal{E}|)$ pairs. For example, if $K' \leq K, |\mathcal{E}|' \leq |\mathcal{E}|$, then for a $(K', |\mathcal{E}|')$ MDCS instance, A' , there exists a $(K, |\mathcal{E}|)$ MDCS instance A and a series of operations of source/encoder deletion/contraction such that A' can be obtained by applying these operations on A, i.e., $\mathsf{A}' \prec \mathsf{A}$.

This definition of an MDCS minor is motivated by the definition of a matroid minor, where if a matroid is not representable over $\mathbb{F}_q$, then its extensions will also not be $\mathbb{F}_q$-representable, because extensions also have the same forbidden minor(s) characterizing $\mathbb{F}_q$-representability. One interesting question is if there also exists similar forbidden minor characterizations for sufficiency of $\mathbb{F}_q$ codes in MDCS instances as the forbidden minor characterizations for representability of $\mathbb{F}_q$ in matroids. Not surprisingly, we have the following theorem.

Theorem 4: Suppose a MDCS instance $\mathsf{A}' = (\mathbf{X}_{\setminus k}, \mathcal{E}', \mathcal{D}', \mathbf{L}', \mathcal{G}')$ is obtained by deleting X_k from

another MDCS instance $\mathsf{A} = (\mathbf{X}_{[[K]]}, \mathcal{E}, \mathcal{D}, \mathbf{L}, \mathcal{G})$, then

$$\mathcal{R}(\mathsf{A}') = \mathrm{Proj}_{\mathbf{X}_{\setminus k}, \mathbf{R}_{\mathcal{E}'}} \left(\mathcal{R}(\mathsf{A}) \cap \{H(X_k) = 0\} \right), \tag{2.59}$$

$$\mathcal{R}_q(\mathsf{A}') = \mathrm{Proj}_{\mathbf{X}_{\setminus k}, \mathbf{R}_{\mathcal{E}'}} \left(\mathcal{R}_q(\mathsf{A}) \cap \{H(X_k) = 0\} \right), \tag{2.60}$$

and similarly

$$\mathcal{R}_{s,q}(\mathsf{A}') = \mathrm{Proj}_{\mathbf{X}_{\setminus k}, \mathbf{R}_{\mathcal{E}'}} \left(\mathcal{R}_{s,q}(\mathsf{A}) \cap \{H(X_k) = 0\} \right). \tag{2.61}$$

$$\mathcal{R}_{sp}(\mathsf{A}') = \mathrm{Proj}_{\mathbf{X}_{\setminus k}, \mathbf{R}_{\mathcal{E}'}} \left(\mathcal{R}_{sp}(\mathsf{A}) \cap \{H(X_k) = 0\} \right). \tag{2.62}$$

Proof: Select any point $\mathbf{R}' \in \mathcal{R}(\mathsf{A}')$, then there exist random variables $\{\mathbf{X}_{\setminus k}, U_i, i \in \mathcal{E}'\}$ such that their entropies satisfy all the constraints in (2.15) determined by A'. Define X_k to be the empty sources, $H(X_k) = 0$. Then the entropies of random variables $\{\mathbf{X}_{\setminus k}, U_i, i \in \mathcal{E}'\} \cup X_k$ will satisfy the constraints in A with $H(X_k) = 0$. Hence, the associated rate point $\mathbf{R} \in \mathcal{R}(\mathsf{A}) \cap \{H(X_k) = 0\}$. Thus, we have $\mathcal{R}(\mathsf{A}') \subseteq \mathrm{Proj}_{\mathbf{X}_{\setminus k}, \mathbf{R}_{\mathcal{E}'}}(\mathcal{R}(\mathsf{A}) \cap \{H(X_k) = 0\})$. If $\mathbf{R}'$ is achievable by $\mathbb{F}_q$ codes or superposition coding, since letting $H(X_k) = 0$ does not affect the other sources and codes, the same $\mathbb{F}_q$ code, or superposition coding, will also achieve the point $\mathbf{R}$ with $H(X_k) = 0$. Thus, we have

$$\mathcal{R}_q(\mathsf{A}') \subseteq \mathrm{Proj}_{\mathbf{X}_{\setminus k}, \mathbf{R}_{\mathcal{E}'}}(\mathcal{R}_q(\mathsf{A}) \cap \{H(X_k) = 0\}), \tag{2.63}$$

$$\mathcal{R}_{s,q}(\mathsf{A}') \subseteq \mathrm{Proj}_{\mathbf{X}_{\setminus k}, \mathbf{R}_{\mathcal{E}'}}(\mathcal{R}_{s,q}(\mathsf{A}) \cap \{H(X_k) = 0\}), \tag{2.64}$$

$$\mathcal{R}_{sp}(\mathsf{A}') \subseteq \mathrm{Proj}_{\mathbf{X}_{\setminus k}, \mathbf{R}_{\mathcal{E}'}}(\mathcal{R}_{sp}(\mathsf{A}) \cap \{H(X_k) = 0\}). \tag{2.65}$$

On the other hand, if we select any point $\mathbf{R} \in \mathcal{R}(\mathsf{A}) \cap \{H(X_k) = 0\}$, we can see that $\mathbf{R}' = \mathrm{Proj}_{\mathbf{X}_{\setminus k}, \mathbf{R}_{\mathcal{E}'}}(\mathbf{R}) \in \mathcal{R}(\mathsf{A}')$ because $\mathbf{R}'$ is still entropic and the entropies of $\{\mathbf{X}_{\setminus k}, U_i, i \in \mathcal{E}'\}$ satisfy all constraints determined by A'. Thus, we have

$$\mathrm{Proj}_{\mathbf{X}_{\setminus k}, \mathbf{R}_{\mathcal{E}'}}(\mathcal{R}(\mathsf{A}) \cap \{H(X_k) = 0\}) \subseteq \mathcal{R}(\mathsf{A}'). \tag{2.66}$$

If $\mathbf{R}$ is achievable by $\mathbb{F}_q$ code $\mathbb{C}$, then the code to achieve $\mathbf{R}'$ could be the code $\mathbb{C}$ with deletion of rows associated with source X_k, i.e., $\mathbb{C}' = \mathbb{C}_{\setminus I(X_k),:}$. Similarly, if $\mathbf{R}$ is achievable by superposition

coding, $\mathbf{R}'$ can be achieved by same superposition coding without coding X_k. Thus,

$$\begin{aligned}
\mathrm{Proj}_{\mathbf{X}_{\setminus k},\mathbf{R}_{\mathcal{E}'}}(\mathcal{R}_q(\mathsf{A}) \cap \{H(X_k)=0\}) &\subseteq \mathcal{R}_q(\mathsf{A}'), && (2.67)\\
\mathrm{Proj}_{\mathbf{X}_{\setminus k},\mathbf{R}_{\mathcal{E}'}}(\mathcal{R}_{s,q}(\mathsf{A}) \cap \{H(X_k)=0\}) &\subseteq \mathcal{R}_{s,q}(\mathsf{A}'), && (2.68)\\
\mathrm{Proj}_{\mathbf{X}_{\setminus k},\mathbf{R}_{\mathcal{E}'}}(\mathcal{R}_{sp}(\mathsf{A}) \cap \{H(X_k)=0\}) &\subseteq \mathcal{R}_{sp}(\mathsf{A}'). && (2.69)
\end{aligned}$$

■

Theorem 5: Suppose a MDCS instance $\mathsf{A}' = (\mathbf{X}_{[[K]]}, \mathcal{E}', \mathcal{D}', \mathbf{L}', \mathcal{G}')$ is obtained by contracting E_e from another MDCS instance $\mathsf{A} = (\mathbf{X}_{[[K]]}, \mathcal{E}, \mathcal{D}, \mathbf{L}, \mathcal{G})$, i.e., $\mathsf{A}' = \mathsf{A} \setminus E_e$, then

$$\begin{aligned}
\mathcal{R}(\mathsf{A}') &= \mathrm{Proj}_{H(X_k),k\in[[K]],\mathbf{R}_{\mathcal{E}'}}\mathcal{R}(\mathsf{A}), && (2.70)\\
\mathcal{R}_q(\mathsf{A}') &= \mathrm{Proj}_{H(X_k),k\in[[K]],\mathbf{R}_{\mathcal{E}'}}\mathcal{R}_q(\mathsf{A}), && (2.71)\\
\mathcal{R}_{s,q}(\mathsf{A}') &\supseteq \mathrm{Proj}_{H(X_k),k\in[[K]],\mathbf{R}_{\mathcal{E}'}}\mathcal{R}_{s,q}(\mathsf{A}), && (2.72)\\
\mathcal{R}_{sp}(\mathsf{A}') &= \mathrm{Proj}_{H(X_k),k\in[[K]],\mathbf{R}_{\mathcal{E}'}}\mathcal{R}_{sp}(\mathsf{A}), && (2.73)
\end{aligned}$$

Proof: Select any point $\mathbf{R}' \in \mathcal{R}'$, then there exist random variables $\{\mathbf{X}_{[[K]]}, U_i, i \in \mathcal{E}'\}$ such that their entropies satisfy all the constraints in (2.15) determined by A'. Define U_e to be the concatenation of all sources, $U_e = \mathbf{X}_{[[K]]}$. Then the entropies of random variables $\{\mathbf{X}_{[[K]]}, U_i, i \in \mathcal{E}'\} \cup U_e$ will satisfy the constraints in A, and additionally obey $H(U_e) = \sum_{k=1}^{K} H(X_k)$. Hence, the associated rate point $\mathbf{R} \in \mathcal{R}(\mathsf{A}) \cap \left\{H(U_e) \geq \sum_{k=1}^{K} H(X_k)\right\}$. Thus, we have

$$\mathcal{R}(\mathsf{A}') \subseteq \mathrm{Proj}_{H(X_k),k\in[[K]],\mathbf{R}_{\mathcal{E}'}}(\mathcal{R}(\mathsf{A}) \cap \{H(U_e) \geq \sum_{k=1}^{K} H(X_k)\}) \subseteq \mathrm{Proj}_{H(X_k),k\in[[K]],\mathbf{R}_{\mathcal{E}'}}\mathcal{R}(\mathsf{A}). \quad (2.74)$$

If $\mathbf{R}'$ is achievable by general $\mathbb{F}_q$ codes or superposition coding, since concatenation of all sources is a valid $\mathbb{F}_q$ code and is superposition coding, we have

$$\mathcal{R}_q(\mathsf{A}') \subseteq \mathrm{Proj}_{H(X_k),k\in[[K]],\mathbf{R}_{\mathcal{E}'}}(\mathcal{R}_q(\mathsf{A}) \cap \{H(U_e) \geq \sum_{k=1}^{K} H(X_k)\}) \subseteq \mathrm{Proj}_{H(X_k),k\in[[K]],\mathbf{R}_{\mathcal{E}'}}\mathcal{R}_q(\mathsf{A}) \quad (2.75)$$

$$\mathcal{R}_{sp}(\mathsf{A}') \subseteq \mathrm{Proj}_{H(X_k),k\in[[K]],\mathbf{R}_{\mathcal{E}'}}(\mathcal{R}_{sp}(\mathsf{A}) \cap \{H(U_e) \geq \sum_{k=1}^{K} H(X_k)\}) \subseteq \mathrm{Proj}_{H(X_k),k\in[[K]],\mathbf{R}_{\mathcal{E}'}}\mathcal{R}_{sp}(\mathsf{A}) \quad (2.76)$$

However, we cannot establish same relationship when scalar $\mathbb{F}_q$ codes are considered, because for the point $\mathbf{R}'$, the associated $\mathbf{R}$ with $H(U_e)$ may not be scalar $\mathbb{F}_q$ achievable.

On the other hand, if we select any point $\mathbf{R} \in \mathcal{R}(\mathsf{A})$, we can see that $\mathbf{R}' = \text{Proj}_{H(X_k),k\in[[K]],\mathbf{R}_{\mathcal{E}'}}\mathbf{R} \in \mathcal{R}(\mathsf{A}')$ because $\mathbf{R}'$ is still entropic and the entropies of $\{\mathbf{X}_{[[K]]}, U_i, i \in \mathcal{E}'\}$ satisfy all constraints determined by A', since they are a subset of the constraints from A. Thus, we have

$$\text{Proj}_{H(X_k),k\in[[K]],\mathbf{R}_{\mathcal{E}'}}\mathcal{R}(\mathsf{A}) \subseteq \mathcal{R}(\mathsf{A}'). \tag{2.77}$$

If $\mathbf{R} \in \mathcal{R}(\mathsf{A})$ is achievable by $\mathbb{F}_q$ code $\mathbb{C}$, then the code to achieve $\mathbf{R}' = \text{Proj}_{H(X_k),k\in[[K]],\mathbf{R}_{\mathcal{E}'}}\mathbf{R} \in \mathcal{R}(\mathsf{A}')$ could be the code $\mathbb{C}$ with deletion of columns associated with encoder E_e, i.e., $\mathbb{C}' = \mathbb{C}_{:,\setminus E_e}$. Thus, we have

$$\text{Proj}_{H(X_k),k\in[[K]],\mathbf{R}_{\mathcal{E}'}}\mathcal{R}_q(\mathsf{A}) \subseteq \mathcal{R}_q(\mathsf{A}'). \tag{2.78}$$

If $\mathbf{R}$ is achievable by scalar $\mathbb{F}_q$ code $\mathbb{C}^1$, then the code to achieve $\mathbf{R}'$ could be the code $\mathbb{C}^1$ with deletion of the column associated with encoder E_e, i.e., $\mathbb{C}^{1'} = \mathbb{C}^1_{:,\setminus I(U_e)}$. Thus, we have

$$\text{Proj}_{H(X_k),k\in[[K]],\mathbf{R}_{\mathcal{E}'}}\mathcal{R}_{s,q}(\mathsf{A}) \subseteq \mathcal{R}_{s,q}(\mathsf{A}'). \tag{2.79}$$

Similarly, if $\mathbf{R}$ is achievable by superposition coding, the same superposition codes can achieve $\mathbf{R}'$ without coding in E_e. Thus, we have

$$\text{Proj}_{H(X_k),k\in[[K]],\mathbf{R}_{\mathcal{E}'}}\mathcal{R}_{sp}(\mathsf{A}) \subseteq \mathcal{R}_{sp}(\mathsf{A}'). \tag{2.80}$$

Combination of (3.48) and (3.50) gives (3.45). Combination of (3.49) and (3.51) gives (3.46). (3.52) indicates (3.47). ■

Theorem 6: Suppose a MDCS instance $\mathsf{A}' = (\mathbf{X}_{[[K]]}, \mathcal{E}', \mathcal{D}', \mathbf{L}', \mathcal{G}')$ is obtained by deleting E_e from another MDCS instance $\mathsf{A} = (\mathbf{X}_{[[K]]}, \mathcal{E}, \mathcal{D}, \mathbf{L}, \mathcal{G})$, then

$$\mathcal{R}(\mathsf{A}') = \text{Proj}_{H(X_k),k\in[[K]],\mathbf{R}_{\mathcal{E}'}}(\mathcal{R}(\mathsf{A}) \cap \{H(U_e) = 0\}), \tag{2.81}$$

$$\mathcal{R}_q(\mathsf{A}') = \text{Proj}_{H(X_k),k\in[[K]],\mathbf{R}_{\mathcal{E}'}}(\mathcal{R}_q(\mathsf{A}) \cap \{H(U_e) = 0\}), \tag{2.82}$$

and similarly

$$\mathcal{R}_{s,q}(\mathsf{A}') = \text{Proj}_{H(X_k),k\in[[K]],\mathbf{R}_{\mathcal{E}'}}(\mathcal{R}_{s,q}(\mathsf{A}) \cap \{H(U_e) = 0\}), \tag{2.83}$$

$$\mathcal{R}_{sp}(\mathsf{A}') = \text{Proj}_{H(X_k),k\in[[K]],\mathbf{R}_{\mathcal{E}'}}(\mathcal{R}_{sp}(\mathsf{A}) \cap \{H(U_e) = 0\}). \tag{2.84}$$

Proof: Select any point $\mathbf{R}' \in \mathcal{R}(\mathsf{A}')$, then there exist random variables $\{\mathbf{X}_{[[K]]}, U_i, i \in \mathcal{E}'\}$ such that their entropies satisfy all the constraints in (2.15) determined by A'. Let U_e be empty set or encode all sources with the all-zero vector, $U_e = \emptyset$. Then the entropies of random variables $\{\mathbf{X}_{[[K]]}, U_i, i \in \mathcal{E}'\} \cup U_e$ will satisfy the constraints in A, and additionally obey $H(U_e) = 0$. Note that, even in the special case where there $\exists D_d, d \in \mathcal{D}$ such that $\text{Fan}(D_d) = E_e$, according to Definition 16, after contraction of E_e, the decoder D'_d will have no access to any decoder but needs to keep the same decoding capability, i.e., $\text{Lev}(D'_d) = \text{Lev}(D_d)$. However, since $\text{Fan}(D'_d) = \emptyset$, deletion of E_e will result in deletion of all decoders of level $\text{Lev}(D_d)$ or less, and deletion of sources $\mathbf{X}_{1:\text{Lev}(D_d)}$, i.e., $H(X_k) = 0, k = 1, \ldots, \text{Lev}(D_d)$. Then it is still true that entropies of random variables $\{\mathbf{X}_{[[K]]}, U_i, i \in \mathcal{E}'\} \cup U_e$ will satisfy the constraints in A. Hence, the associated rate point $\mathbf{R} \in \mathcal{R}(\mathsf{A}) \cap \{H(U_e) = 0\}$. Thus, we have

$$\mathcal{R}(\mathsf{A}') \subseteq \text{Proj}_{H(X_k), k \in [[K]], \mathbf{R}_{\mathcal{E}'}}(\mathcal{R}(\mathsf{A}) \cap \{H(U_e) = 0\}). \tag{2.85}$$

If $\mathbf{R}'$ is achievable by $\mathbb{F}_q$ linear vector or scalar codes, or superposition coding, since all-zero code is a valid $\mathbb{F}_q$ code and superposition code, we have

$$\mathcal{R}_q(\mathsf{A}') \subseteq \text{Proj}_{H(X_k), k \in [[K]], \mathbf{R}_{\mathcal{E}'}}(\mathcal{R}_q(\mathsf{A}) \cap \{H(U_e) = 0\}), \tag{2.86}$$

$$\mathcal{R}_{s,q}(\mathsf{A}') \subseteq \text{Proj}_{H(X_k), k \in [[K]], \mathbf{R}_{\mathcal{E}'}}(\mathcal{R}_{s,q}(\mathsf{A}) \cap \{H(U_e) = 0\}), \tag{2.87}$$

$$\mathcal{R}_{sp}(\mathsf{A}') \subseteq \text{Proj}_{H(X_k), k \in [[K]], \mathbf{R}_{\mathcal{E}'}}(\mathcal{R}_{sp}(\mathsf{A}) \cap \{H(U_e) = 0\}). \tag{2.88}$$

On the other hand, if we select any point $\mathbf{R} \in \mathcal{R}(\mathsf{A}) \cap \{H(U_e) = 0\}$, we can see that $\mathbf{R}' = \text{Proj}_{H(X_k), k \in [[K]], \mathbf{R}_{\mathcal{E}'}}(\mathbf{R}) \in \mathcal{R}(\mathsf{A}')$ because $\mathbf{R}'$ is still entropic and the entropies of $\{\mathbf{X}_{[[K]]}, U_i, i \in \mathcal{E}'\}$ satisfy all constraints determined by A', which is still true when the special case happens. Thus, we have

$$\text{Proj}_{H(X_k), k \in [[K]], \mathbf{R}_{\mathcal{E}'}}(\mathcal{R}(\mathsf{A}) \cap \{H(U_e) = 0\}) \subseteq \mathcal{R}(\mathsf{A}'). \tag{2.89}$$

If $\mathbf{R}$ is achievable by a $\mathbb{F}_q$ code, vector or scalar, or a superposition code, $\mathbb{C}$, then the code to achieve $\mathbf{R}'$ could be the code $\mathbb{C}$ with the deletion of the columns associated with encoder E_e, i.e., $\mathbb{C}' = \mathbb{C}_{:,\setminus I(U_e)}$, because E_e is sending nothing. Thus, we have

$$\text{Proj}_{H(X_k), k \in [[K]], \mathbf{R}_{\mathcal{E}'}}(\mathcal{R}_q(\mathsf{A}) \cap \{H(U_e) = 0\}) \subseteq \mathcal{R}_q(\mathsf{A}'), \tag{2.90}$$

$$\mathrm{Proj}_{H(X_k),k\in[[K]],\mathbf{R}_{\mathcal{E}'}}(\mathcal{R}_{s,q}(\mathsf{A})\cap\{H(U_e)=0\})\subseteq\mathcal{R}_{s,q}(\mathsf{A}'),\tag{2.91}$$

$$\mathrm{Proj}_{H(X_k),k\in[[K]],\mathbf{R}_{\mathcal{E}'}}(\mathcal{R}_{sp}(\mathsf{A})\cap\{H(U_e)=0\})\subseteq\mathcal{R}_{sp}(\mathsf{A}').\tag{2.92}$$

Combination of (3.56) and (3.59) gives (3.53). Combination of (3.57) and (3.60) gives (3.54). Combination of (3.58) and (3.61) gives (3.55). ■

Theorem 7: Suppose a MDCS instance $\mathsf{A}'=(\mathbf{X}_{[[K]]},\mathcal{E}',\mathcal{D}',\mathbf{L}',\mathcal{G}')$ is obtained by unifying E_f with E_e, from another MDCS instance $\mathsf{A}=(\mathbf{X}_{[[K]]},\mathcal{E},\mathcal{D},\mathbf{L},\mathcal{G})$. Let $\mathcal{L}_1,\mathcal{L}_2,\mathcal{L}_5,\mathcal{L}_4''$ be the constraints for A used in (3.14) to obtain $\mathcal{R}(\mathsf{A})$. Let

$$\mathcal{L}_4^0=\{(\mathbf{h}^T,\mathbf{R}^T)^T\in\mathbb{R}_+^{2^N-1+|\mathcal{E}|}:R_{e'}\geq H(U_e,U_f),R_i\geq H(U_i),i\in\mathcal{E}\setminus e\}\tag{2.93}$$

then

$$\begin{aligned}\mathcal{R}(\mathsf{A}') &= \mathrm{Proj}_{H(X_k),k\in[[K]],R_{\mathcal{E}\setminus\{e,f\}},R_{e'}}\overline{\mathrm{con}(\Gamma_{\mathrm{N}}^*\cap\mathcal{L}_{12})}\cap\mathcal{L}_5\cap\mathcal{L}_4^0, &(2.94)\\ \mathcal{R}_q(\mathsf{A}') &= \mathrm{Proj}_{H(X_k),k\in[[K]],R_{\mathcal{E}\setminus\{e,f\}},R_{e'}}(\Gamma_{N,\infty}^q\cap\mathcal{L}_{125}\cap\mathcal{L}_4^0), &(2.95)\\ \mathcal{R}_{sp}(\mathsf{A}') &= \mathrm{Proj}_{H(X_k),k\in[[K]],R_{\mathcal{E}\setminus\{e,f\}},R_{e'}}(\overline{\mathrm{con}(\Gamma_{\mathrm{N}}^*\cap\mathcal{L}_{12})}\cap\mathcal{L}_5'\cap\mathcal{L}_4^0). &(2.96)\end{aligned}$$

where the dimension R_e is replaced with $R_{e'}$ in $\mathcal{L}_4^0$ and the projection.

Proof: Select any point $\mathbf{R}'\in\mathcal{R}(\mathsf{A}')$, then there exist random variables $\{\mathbf{X}_{[[K]]},U_i,i\in\mathcal{E}'\}$ such that their entropies satisfy all the constraints determined by A' and $R_i\geq H(U_i),i\in\mathcal{E}'$. Let $U_e=U_{e'},U_f=U_{e'}$, so that $H(U_e)=H(U_f)=H(U_{e'})=H(U_e,U_f)$ and $R_e=R_f=R_{e'}$. Then the entropies of random variables $\{\mathbf{X}_{[[K]]},U_i,i\in\mathcal{E}'\setminus e'\}\cup\{U_e,U_f\}$ will satisfy the constraints in A, and additionally $R_i,H(U_i),i\in\mathcal{E}$ will obey $\mathcal{L}_4^0$. Hence, the associated rate point $\mathbf{R}\in\mathrm{Proj}_{H(X_k),k\in[[K]],R_{\mathcal{E}\setminus\{e,f\}},R_{e'}}\overline{\mathrm{con}(\Gamma_{\mathrm{N}}^*\cap\mathcal{L}_{12})}\cap\mathcal{L}_5\cap\mathcal{L}_4^0$. Thus, we have

$$\mathcal{R}(\mathsf{A}')\subseteq\mathrm{Proj}_{H(X_k),k\in[[K]],R_{\mathcal{E}\setminus\{e,f\}},R_{e'}}\overline{\mathrm{con}(\Gamma_{\mathrm{N}}^*\cap\mathcal{L}_{12})}\cap\mathcal{L}_5\cap\mathcal{L}_4^0.\tag{2.97}$$

If $\mathbf{R}'$ is achievable by $\mathbb{F}_q$ vector codes or superposition coding, since U_e,U_f are replicating $U_{e'}$ with exactly the same code, $\mathbf{R}$ is also $\mathbb{F}_q$ achievable or superposition coding achievable. Then we

have

$$\mathcal{R}_q(\mathsf{A}') \subseteq \mathrm{Proj}_{H(X_k),k\in[[K]],R_{\mathcal{E}\setminus\{e,f\}},R_{e'}}(\Gamma^q_{N,\infty}\cap\mathcal{L}_{125}\cap\mathcal{L}_4^0), \tag{2.98}$$

$$\mathcal{R}_{sp}(\mathsf{A}') \subseteq \mathrm{Proj}_{H(X_k),k\in[[K]],R_{\mathcal{E}\setminus\{e,f\}},R_{e'}}\overline{\mathrm{con}(\Gamma^*_{\mathrm{N}}\cap\mathcal{L}_{12})}\cap\mathcal{L}_5'\cap\mathcal{L}_4^0. \tag{2.99}$$

On the other hand, if we select any point $\mathbf{R}\in\mathrm{Proj}_{H(X_k),k\in[[K]],R_{\mathcal{E}\setminus\{e,f\}},R_{e'}}\overline{\mathrm{con}(\Gamma^*_{\mathrm{N}}\cap\mathcal{L}_{12})}\cap\mathcal{L}_5\cap\mathcal{L}_4^0$, there exist random variables $\{\mathbf{X}_{[[K]]},U_i,i\in\mathcal{E}\}$ such that their entropies satisfy all the constraints determined by A together with $R_i\geq H(U_i),i\in\mathcal{E}$ and $R_{e'}\geq H(U_e,U_f)$. Let $U_{e'}$ be the concatenation of U_e,U_f so that $H(U_{e'})=H(U_e,U_f)$. After unification, since all decoders that are fan of E_e,E_f will have access to $E_{e'}$, $\mathbf{R}'$ will satisfy all constraints determined in A'. Thus, $\mathbf{R}'\in\mathcal{R}(\mathsf{A}')$. Hence, we have

$$\mathrm{Proj}_{H(X_k),k\in[[K]],R_{\mathcal{E}\setminus\{e,f\}},R_{e'}}\overline{\mathrm{con}(\Gamma^*_{\mathrm{N}}\cap\mathcal{L}_{12})}\cap\mathcal{L}_5\cap\mathcal{L}_4^0\subseteq\mathcal{R}(\mathsf{A}'). \tag{2.100}$$

If $\mathbf{R}$ is achievable by $\mathbb{F}_q$ vector codes $\mathbb{C}$ with time-sharing between basic solutions $\mathbb{C}^{(1)},\mathbb{C}^{(2)},\ldots,\mathbb{C}^{(m)}$, then the odes to achieve $\mathbf{R}'$ could be same time-sharing between basic solutions $\mathbb{C}^{(1')},\mathbb{C}^{(2')},\ldots,\mathbb{C}^{(m')}$, where

$$\mathbb{C}^{(i')}_{:,I_{i'}(U_j)} = \mathbb{C}^{(i)}_{:,I_i(U_j)}, j\in\mathcal{E}\setminus\{e,f\}, \tag{2.101}$$

$$\mathbb{C}^{(i')}_{:,I_{i'}(U_e)} = \left[\mathbb{C}^{(i)}_{:,I_i(U_e)}\ \mathbb{C}^{(i)}_{:,I_i(U_f)\setminus\mathcal{B}}\right], \tag{2.102}$$

$$\mathcal{B}=\left\{j\in I_i(U_f)\,\middle|\,\mathbb{C}^{(i)}_{:,j}\in\mathrm{Span}(\mathbb{C}^{(i)}_{:,I_i(U_e)})\right\}, i\in\{1,2,\ldots,m\}. \tag{2.103}$$

Thus,

$$\mathrm{Proj}_{H(X_k),k\in[[K]],R_{\mathcal{E}\setminus\{e,f\}},R_{e'}}\overline{\mathrm{con}(\Gamma^{\mathrm{q}}_{\mathrm{N}}\cap\mathcal{L}_{12})}\cap\mathcal{L}_5\cap\mathcal{L}_4^0\subseteq\mathcal{R}_q(\mathsf{A}'). \tag{2.104}$$

Similarly, if $\mathbf{R}\in\mathrm{Proj}_{H(X_k),k\in[[K]],R_{\mathcal{E}\setminus\{e,f\}},R_{e'}}(\Gamma^*_N\cap\mathcal{L}_{12})\cap\mathcal{L}_5'\cap\mathcal{L}_4^0)$ is achieved by superposition coding, one can use entropy approaching codes, e.g., Huffman code with sufficient number of blocks, to jointly encode each component in U_e,U_f for the sources so that $H(U_{e'}^{X_k})=H(U_e^{X_k},U_f^{X_k}),k\in[[K]]$. Thus, we have

$$\mathrm{Proj}_{H(X_k),k\in[[K]],R_{\mathcal{E}\setminus\{e,f\}},R_{e'}}\overline{\mathrm{con}(\Gamma^*_{\mathrm{N}}\cap\mathcal{L}_{12})}\cap\mathcal{L}_5'\cap\mathcal{L}_4^0\subseteq\mathcal{R}_{sp}(\mathsf{A}'). \tag{2.105}$$

■

The previous theorem presents a form of the rate regions which is best for proving the desired inheritance properties, but the form is not extendable to the scalar coding region. We now present

an alternate representation of the rate regions that also holds for the scalar coding region, but is not as useful for proving the desired inheritance properties.

Theorem 8: Suppose a MDCS instance $\mathsf{A}' = (\mathbf{X}_{[[K]]}, \mathcal{E}', \mathcal{D}', \mathbf{L}', \mathcal{G}')$ is obtained by unifying E_f with E_e, from another MDCS instance $\mathsf{A} = (\mathbf{X}_{[[K]]}, \mathcal{E}, \mathcal{D}, \mathbf{L}, \mathcal{G})$. Let $\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_5, \mathcal{L}_4''$ be the constraints for A used in (3.14) to obtain $\mathcal{R}(\mathsf{A})$. Let

$$\mathcal{L}_4^1 = \{(\mathbf{h}^T, \mathbf{R}^T)^T \in \mathbb{R}_+^{2^N-1+|\mathcal{E}|} : H(U_e) = H(U_f) = H(U_e, U_f), R_i \geq H(U_i), i \in \mathcal{E}\}, \tag{2.106}$$

then

$$\begin{aligned}
\mathcal{R}(\mathsf{A}') &= \mathrm{Proj}_{H(X_k), k\in[[K]], R_e, e\in\mathcal{E}'} \overline{\mathrm{con}(\Gamma_{\mathrm{N}}^* \cap \mathcal{L}_{12})} \cap \mathcal{L}_5 \cap \mathcal{L}_4^1, && (2.107)\\
\mathcal{R}_q(\mathsf{A}') &= \mathrm{Proj}_{H(X_k), k\in[[K]], R_e, e\in\mathcal{E}'} (\Gamma_{N,\infty}^q \cap \mathcal{L}_{125} \cap \mathcal{L}_4^1), && (2.108)\\
\mathcal{R}_{s,q}(\mathsf{A}') &= \mathrm{Proj}_{H(X_k), k\in[[K]], R_e, e\in\mathcal{E}'} (\Gamma_N^q \cap \mathcal{L}_{125}) \cap \mathcal{L}_4^1), && (2.109)\\
\mathcal{R}_{sp}(\mathsf{A}') &= \mathrm{Proj}_{H(X_k), k\in[[K]], R_e, e\in\mathcal{E}'} (\overline{\mathrm{con}(\Gamma_{\mathrm{N}}^* \cap \mathcal{L}_{12})} \cap \mathcal{L}_5' \cap \mathcal{L}_4^1). && (2.110)
\end{aligned}$$

Proof: Select any point $\mathbf{R}' \in \mathcal{R}(\mathsf{A}')$, then there exist random variables $\{\mathbf{X}_{[[K]]}, U_i, i \in \mathcal{E}'\}$ such that their entropies satisfy all the constraints determined by A' and $R_i \geq H(U_i), i \in \mathcal{E}'$. Let $U_e = U_f, R_f = R_e$, so that $H(U_e) = H(U_f) = H(U_e, U_f)$. Then the entropies of random variables $\{\mathbf{X}_{[[K]]}, U_i, i \in \mathcal{E}'\} \cup U_f$ will satisfy the constraints in A, and additionally $R_i, H(U_i), i \in \mathcal{E}$ will obey $\mathcal{L}_4^1$. Hence, the associated rate point $\mathbf{R} \in \mathrm{Proj}_{H(X_k), k\in[[K]], R_e, e\in\mathcal{E}'} \overline{\mathrm{con}(\Gamma_{\mathrm{N}}^* \cap \mathcal{L}_{12})} \cap \mathcal{L}_5 \cap \mathcal{L}_4^1$. Thus, we have

$$\mathcal{R}(\mathsf{A}') \subseteq \mathrm{Proj}_{H(X_k), k\in[[K]], R_e, e\in\mathcal{E}'} \overline{\mathrm{con}(\Gamma_{\mathrm{N}}^* \cap \mathcal{L}_{12})} \cap \mathcal{L}_5 \cap \mathcal{L}_4^1. \tag{2.111}$$

If $\mathbf{R}'$ is achievable by a $\mathbb{F}_q$ codes, scalar or vector, or a superposition code, since U_f is replicating U_e with exactly the same code, $\mathbf{R}$ is also $\mathbb{F}_q$ achievable. Then we have

$$\begin{aligned}
\mathcal{R}_q(\mathsf{A}') &\subseteq \mathrm{Proj}_{H(X_k), k\in[[K]], R_e, e\in\mathcal{E}'} (\Gamma_{N,\infty}^q \cap \mathcal{L}_{125} \cap \mathcal{L}_4^1), && (2.112)\\
\mathcal{R}_{s,q}(\mathsf{A}') &\subseteq \mathrm{Proj}_{H(X_k), k\in[[K]], R_e, e\in\mathcal{E}'} (\Gamma_N^q \cap \mathcal{L}_{125} \cap \mathcal{L}_4^1), && (2.113)\\
\mathcal{R}_{sp}(\mathsf{A}') &\subseteq \mathrm{Proj}_{H(X_k), k\in[[K]], R_e, e\in\mathcal{E}'} \overline{\mathrm{con}(\Gamma_{\mathrm{N}}^* \cap \mathcal{L}_{12})} \cap \mathcal{L}_5' \cap \mathcal{L}_4^1. && (2.114)
\end{aligned}$$

On the other hand, if we select any point $\mathbf{R} \in \mathrm{Proj}_{H(X_k), k\in[[K]], R_e, e\in\mathcal{E}'} \overline{\mathrm{con}(\Gamma_{\mathrm{N}}^* \cap \mathcal{L}_{12})} \cap \mathcal{L}_5 \cap \mathcal{L}_4^1$, there exist random variables $\{\mathbf{X}_{[[K]]}, U_i, i \in \mathcal{E}\}$ such that their entropies satisfy all the constraints determined by A together with $R_i \geq H(U_i), i \in \mathcal{E}$ and $H(U_e) = H(U_f) = H(U_e, U_f)$. After

unification, since all decoders that are fan of E_e, E_f will have access to E_e, $\mathbf{R}'$ will satisfy all constraints determined in A' because U_f is dependent on U_e. Thus, $\mathbf{R}' \in \mathcal{R}(\mathsf{A}')$. Hence, we have

$$\mathrm{Proj}_{H(X_k),k\in[[K]],R_e,e\in\mathcal{E}'}\overline{\mathrm{con}(\Gamma_{\mathrm{N}}^* \cap \mathcal{L}_{12})} \cap \mathcal{L}_5 \cap \mathcal{L}_4^1 \subseteq \mathcal{R}(\mathsf{A}'). \tag{2.115}$$

If $\mathbf{R}$ is achievable by a $\mathbb{F}_q$ code, scalar or vector, or a superposition code, $\mathbb{C}$, then the codes to achieve $\mathbf{R}'$ could be same as $\mathbb{C}$ with deletion of columns associated with encoder E_f. Thus,

$$\mathrm{Proj}_{H(X_k),k\in[[K]],R_e,e\in\mathcal{E}'}(\Gamma_{N,\infty}^q \cap \mathcal{L}_{125} \cap \mathcal{L}_4^1 \subseteq \mathcal{R}_q(\mathsf{A}'), \tag{2.116}$$

$$\mathrm{Proj}_{H(X_k),k\in[[K]],R_e,e\in\mathcal{E}'}(\Gamma_{N}^q \cap \mathcal{L}_{125} \cap \mathcal{L}_4^1 \subseteq \mathcal{R}_{s,q}(\mathsf{A}'), \tag{2.117}$$

$$\mathrm{Proj}_{H(X_k),k\in[[K]],R_e,e\in\mathcal{E}'}\overline{\mathrm{con}(\Gamma_{\mathrm{N}}^* \cap \mathcal{L}_{12})} \cap \mathcal{L}_5 \cap \mathcal{L}_4^1 \subseteq \mathcal{R}_{sp}(\mathsf{A}'). \tag{2.118}$$

■

Corollary 2: Given two MDCS instances A, A' such that $\mathsf{A}' \prec_v \mathsf{A}$. If $\mathbb{F}_q$ linear vector codes suffice for A, then $\mathbb{F}_q$ linear vector codes suffice for A'. Equivalently, if $\mathbb{F}_q$ linear vector codes do not suffice for A', then $\mathbb{F}_q$ linear vector codes do not suffice for A. Equivalently, if $\mathcal{R}_q(\mathsf{A}) = \mathcal{R}(\mathcal{A})$, then $\mathcal{R}_q(\mathcal{A}') = \mathcal{R}(\mathcal{A}')$.

Proof: From Definition 17 we know that A' is obtained by a series of operations of source deletion, encoder deletion, encoder contraction, and encoder unified contraction. Theorem 1 indicates that the order of the operations does not matter. Thus, it suffices to show that the statement holds when A' can be obtained by one of the operations of source deletion, encoder deletion, encoder contraction, or encoder unification on A.

Suppose $\mathbb{F}_q$ linear codes suffice to achieve every point in $\mathcal{R}(\mathsf{A})$, i.e., $\mathcal{R}_q(\mathsf{A}) = \mathcal{R}(\mathsf{A})$. If A' is obtained by contracting one encoder in A, (3.45) and (3.46) in Theorem 14 indicate $\mathcal{R}_q(\mathsf{A}') = \mathcal{R}(\mathsf{A})$. Similarly, same conclusion is obtained from (3.42) and (3.43) in Theorem 13, (3.53) and (3.54) in Theorem 15, when A' is obtained by source deletion or deletion deletion.

When A' is obtained by unifying two encoders in A and $\mathcal{R}_q(\mathsf{A}) = \mathcal{R}(\mathsf{A})$. Trivially, $\mathcal{R}_q(\mathsf{A}') \subseteq \mathcal{R}(\mathsf{A}')$ since $\Gamma_{N,\infty}^q \subseteq \bar{\Gamma}_N^*$. For the other direction, we pick a point $\mathbf{R}' \in \mathcal{R}(\mathsf{A}')$. Note that $\mathcal{R}(\mathsf{A}') \subseteq \mathrm{Proj}_{R_e,H(X_k),R_{e'},e\in\mathcal{E}\setminus\{e,f\},k\in[[K]]}\mathcal{R}(\mathsf{A})$ since $H(U_e, U_f) \geq H(U_e)$ and $\mathcal{L}_4^0$ makes the region smaller than $\mathcal{R}(\mathsf{A})$. Since $\mathcal{R}_q(\mathsf{A}) = \mathcal{R}(\mathsf{A})$, we see that $\mathbf{R}' \in \mathrm{Proj}_{R_e,H(X_k),R_{e'},e\in\mathcal{E}\setminus\{e,f\},k\in[[K]]}\mathcal{R}_q(\mathsf{A})$, which means that there exist $\mathbf{h} \in \Gamma_{N,\infty}^q$ and $R_i, i \in \mathcal{E}$ such that their entropies satisfy constraints determined by A and $R_{e'} \geq H(U_e, U_f)$. The $\mathbb{F}_q$ code for $U_{e'}$ is the concatenation (with removal of redundancy) of $\mathbb{F}_q$ codes for U_e, U_f. That is, $\mathbf{R}' \in \mathcal{R}_q(\mathsf{A}')$. Thus, $\mathcal{R}(\mathsf{A}') \subseteq \mathcal{R}_q(\mathsf{A}')$. Therefore, we

have $\mathcal{R}(\mathsf{A}') = \mathcal{R}_q(\mathsf{A}')$.

We see that for one-step operations, sufficiency of $\mathbb{F}_q$ codes is preserved. The statement holds in general. ■

Note that scalar codes are a spacial class of general linear codes. If we only consider scalar linear codes and operations of source deletion, encoder deletion/ contraction, we will have the following similar corollary.

Corollary 3: Given two MDCS instances A, A' such that $\mathsf{A}' \prec_s \mathsf{A}$. If $\mathbb{F}_q$ scalar linear codes suffice for A, then $\mathbb{F}_q$ scalar linear codes suffice for A'. Equivalently, if $\mathbb{F}_q$ scalar linear codes do not suffice for A', then $\mathbb{F}_q$ scalar linear codes do not suffice for A. Equivalently, if $\mathcal{R}_{s,q}(\mathsf{A}) = \mathcal{R}(\mathcal{A})$, then $\mathcal{R}_{s,q}(\mathcal{A}') = \mathcal{R}(\mathcal{A}')$.

Proof: If $\mathcal{R}_{s,q}(\mathsf{A}) = \mathcal{R}(\mathcal{A})$, we can get $\mathcal{R}_{s,q}(\mathcal{A}') = \mathcal{R}(\mathcal{A}')$ from (3.53) and (3.55) in Theorem 15 ((3.42) and (3.44) in Theorem 13), if A' is obtained by deleting an encoder (source) from A.

Now we consider the case that A' is obtained by contracting an encoder from A. If $\mathcal{R}_{s,q}(\mathsf{A}) = \mathcal{R}(\mathcal{A})$, the projections $\mathrm{Proj}_{H(X_k),k\in[[K]],\mathbf{R}_{\mathcal{E}'}}\mathcal{R}_{s,q}(\mathsf{A}) = \mathrm{Proj}_{H(X_k),k\in[[K]],\mathbf{R}_{\mathcal{E}'}}\mathcal{R}(\mathcal{A})$. Together with (3.45) and (3.47), we will have $\mathcal{R}_{s,q}(\mathcal{A}') \supseteq \mathcal{R}(\mathcal{A}')$. It is trivial that $\mathcal{R}_{s,q}(\mathcal{A}') \subseteq \mathcal{R}(\mathcal{A}')$ because $\Gamma_N^q \subseteq \bar{\Gamma}_N^*$. Thus, we have $\mathcal{R}_{s,q}(\mathcal{A}') = \mathcal{R}(\mathcal{A}')$. ■

Similarly, if superposition coding is considered, we have the following corollary.

Corollary 4: Given two MDCS instances A, A' such that $\mathsf{A}' \prec_{sp} \mathsf{A}$. If superposition coding suffices for A, then it also suffices for A'. Equivalently, if superposition coding does not suffice for A', then it does not suffice for A. Equivalently, if $\mathcal{R}_{sp}(\mathsf{A}) = \mathcal{R}(\mathcal{A})$, then $\mathcal{R}_{sp}(\mathcal{A}') = \mathcal{R}(\mathcal{A}')$.

Proof: Suppose superposition coding suffices to achieve every point in $\mathcal{R}(\mathsf{A})$, i.e., $\mathcal{R}_q(\mathsf{A}) = \mathcal{R}(\mathsf{A})$. If A' is obtained by contracting one encoder in A, (3.45) and (2.73) in Theorem 14 indicate $\mathcal{R}_q(\mathsf{A}') = \mathcal{R}(\mathsf{A})$. Similarly, same conclusion is obtained from (3.42) and (2.62) in Theorem 13, (3.53) and (2.84) in Theorem 15, when A' is obtained by source deletion or encoder deletion.

When A' is obtained by unifying two encoders in A and $\mathcal{R}_{sp}(\mathsf{A}) = \mathcal{R}(\mathsf{A})$. Trivially, $\mathcal{R}_{sp}(\mathsf{A}') \subseteq \mathcal{R}(\mathsf{A}')$, since a point achieved by superposition coding must be in the rate region. For the other direction, we pick a point $\mathbf{R}' \in \mathcal{R}(\mathsf{A}')$. Note that $\mathcal{R}(\mathsf{A}') \subseteq \mathrm{Proj}_{R_e,H(X_k),R_{e'},e\in\mathcal{E}\setminus\{e,f\},k\in[[K]]}\mathcal{R}(\mathsf{A})$ since $H(U_e, U_f) \geq H(U_e)$ and $\mathcal{L}_4^0$ makes the region smaller than $\mathcal{R}(\mathsf{A})$. Then we have $\mathbf{R}' \in \mathrm{Proj}_{R_e,H(X_k),R_{e'},e\in\mathcal{E}\setminus\{e,f\},k\in[[K]]}\mathcal{R}(\mathsf{A})$. Since $\mathcal{R}_{sp}(\mathsf{A}) = \mathcal{R}(\mathsf{A})$, we see that

$$\mathbf{R}' \in \mathrm{Proj}_{R_e,H(X_k),R_{e'},e\in\mathcal{E}\setminus\{e,f\},k\in[[K]]}\mathcal{R}_{sp}(\mathsf{A}), \tag{2.119}$$

Table 2.2: Sufficiency of codes for MDCS instances: Columns 2–7 show the number of instances that the rate region inner bounds match with the Shannon outer bound.

$(K, \lvert\mathcal{E}\rvert)$	$\lvert\mathcal{M}\rvert$	$\mathcal{R}_{s,2}(\mathsf{A})$	$\mathcal{R}_{s,3}(\mathsf{A})$	$\mathcal{R}_2^{N,N+1}(\mathsf{A})$	$\mathcal{R}_2^{N,N+2}(\mathsf{A})$	$\mathcal{R}_2^{N,N+3}(\mathsf{A})$	$\mathcal{R}_{sp}(\mathsf{A})$
$(1,2)$	1	1	1	1	1	1	1
$(1,3)$	3	2	2	3	3	3	3
$(1,4)$	13	5	5	9	11	12	13
$(2,2)$	3	3	3	3	3	3	3
$(2,3)$	23	17	17	23	23	23	21
$(2,4)$	445	152	152	317	388	429	315
$(3,2)$	1	1	1	1	1	1	1
$(3,3)$	68	55	55	68	68	68	56
$(3,4)$	6803	1692	1692	4336	5766	6326	3094

which means that there exist $\mathbf{h} \in \Gamma_N^*$ and $R_e = \sum_{i=1}^K R_e^{X_k}, k \in [[K]], e \in \mathcal{E}$ such that their entropies satisfy constraints determined by A with $\mathcal{L}_5'$ in equation (2.39) and $R_{e'} \geq H(U_e, U_f)$. One can use an entropy approaching code, e.g., Huffman code with a sufficient large block length, to code the concatenation $U_e^{X_k}, U_f^{X_k}$ for each $k \in [[K]]$ to obtain an overall superposition code for $U_{e'}$. That is, $\mathbf{R}' \in \mathcal{R}_{sp}(\mathsf{A}')$. Thus, $\mathcal{R}(\mathsf{A}') \subseteq \mathcal{R}_{sp}(\mathsf{A}')$. Therefore, we have $\mathcal{R}(\mathsf{A}') = \mathcal{R}_{sp}(\mathsf{A}')$. ■

2.6 Rate region results on MDCS problems

In this section, experimental results on thousands of MDCS instances are presented. We investigate rate regions for 7360 non-isomorphic MDCS instances which represent 134617 isomorphic instances including the cases when $(K, |\mathcal{E}|) = (2,2), (2,3), (2,4), (3,2), (3,3), (3,4)$. For each non-isomorphic MDCS instance, we calculated the bounds on its rate region using Shannon outer bound Γ_N, scalar binary representable matroid inner bound Γ_N^2, scalar ternary representable matroid inner bound Γ_N^3, vector binary representable matroid inner bounds $\Gamma_{N,N+1}^2, \Gamma_{N,N+2}^2, \Gamma_{N,N+3}^2$. If the outer bound on rate region obtained from Shannon outer bound matches with any inner bound from the corresponding inner bound on region of entropic vectors, we not only know the exact rate region but also know the codes which suffice to achieve any point in it.

A summary of results can be found in Table 2.2. Since there are thousands of networks, it is impossible to state the rate regions one by one. Interested readers are referred to our website [37] for the complete list of rate regions and other interesting results such as the enumeration of MDCS instances and converse proofs generated by computer.

Some interpretations of the results from several perspectives with some example networks are presented as follows.

2.6.1 Tightness of Shannon outer bound

The first question we are interested in is if the Shannon outer bound (i.e., the LP bound) is tight for the considered MDCS rate regions, i.e., if non-Shannon type inequalities are necessary in order to get the rate regions. Trivially, for $(1,2),(1,3),(1,4)$ MDCS problems, the Shannon outer bound is tight since the rate region for single source problems can be determined by the min-cut bound. Our earlier results presented in [22, 23] proved that rate regions obtained from the Shannon outer bound are tight for all the cases for $(2,2),(2,3),(3,2),(3,3)$ MDCS instances as well (the Shannon outer bound turns out to be tight for the two cases miss-counted in [1] as $(3,3)$ MDCS instances). For $(2,4)$ MDCS, we have proven that the Shannon outer bound is tight for 429 out of 455 non-isomorphic instances by using up to $\Gamma^2_{6,9}$. Similarly, for 3-level 4-encoder MDCS, we have proven that the Shannon outer bound is tight for 6326 out of 6803 non-isomorphic instances by using up to $\Gamma^2_{7,10}$. By grouping more variables on representable matroid inner bounds over various fields $\mathbb{F}_q$, say using $\Gamma^2_{6,10}, \Gamma^3_{6,10}, \Gamma^2_{7,11}, \Gamma^3_{7,11}$,, etc., we may expect that the Shannon outer bound will be tight for additional 2-level and 3-level 4-encoder MDCS instances. As will be discussed later, for these instances where the Shannon outer bound on the region of entropic vectors is tight, the rate regions can be achieved by various codes, such as superposition and scalar/vector binary/ternary codes.

2.6.2 Sufficiency of superposition

In superposition coding, i.e., source separation, the data sources are encoded separately and the output from an encoder is just the concatenation of those separated codewords. In this manner, each encoder can be viewed as a combination of several sub-encoders, and thus the coding rate of an encoder is the sum of coding rate of each sub-encoder. If every point in the rate region can be achieved by superposition coding, we say superposition coding suffices.

When there is only one source in the network, there is no distinguish between superposition and linear coding. Therefore, superposition suffices for all $(1,2),(1,3),(1,4)$ MDCS instances, as shown in Table 2.2.

For the 2-level 2-encoder MDCS instances, superposition coding suffices. However, it is shown in [1, 31] that superposition coding is not sufficient for all the 100 non-isomorphic 3-encoder MDCS instances[1]. There are only 86 out of them[2] are achievable by superposition coding [1, 31]. The remaining instances have rate regions for which every point can be achieved by linear coding between sources. We found that superposition suffices for 315 out of the 455 non-isomorphic $(2,4)$ instances,

[1] actually, after correcting the errors we discussed in §2.2, we found 95, including 3 cases for $(2,2)$, 1 case for $(3,2)$, 23 cases for $(2,3)$ and 68 cases for $(3,3)$ MDCS instances

[2] actually 81 out of 95

and suffices for 3094 out of 6803 non-isomorphic $(3, 4)$ MDCS instances. Superposition suffices for a significant fraction of all non-isomorphic MDCS instances in these classes.

2.6.3 Sufficiency of scalar codes

When coding across sources is necessary, we first would like to see if simple codes suffice. [22, 23] showed that scalar binary codes are insufficient for 6 out of the 23 cases and 15 instances out of the 68 cases (at the time of these publications, we believed the number of $(2, 3)$ and $(3, 3)$ MDCS instances to be the same as found in [1]). The 6 instances for $(2, 3)$ can be found in Table 2.3. The 15 instances for $(2, 3)$ MDCS include numbers 8, 14, 28, 32, 37, 42, 47, 49, 53, 55, 57, 59, 63, 65, 69 from the list in [1]. Binary codes turn out to suffice for the two instances missed in [1].

One natural question is whether scalar linear codes over a larger field size can eliminate the gap in any of the cases where scalar linear binary codes were insufficient. Our calculations showed that exactly the same achievable rate regions for 2-level and 3-level MDCS instances with 3 encoders MDCS instances are obtained by considering the larger inner bound of matroids, i.e. by replacing Γ_N^2 with Γ_N^3 and Γ_N^{mat} for $N \in \{5, 6\}$, where Γ_N^{mat} is the conic hull of all matroid ranks on N elements. Since for $N \in \{5, 6, 7\}$, all matroids are representable in some field, Γ_N^{mat} is also an inner bound on $\bar{\Gamma}_N^*$ but tighter than Γ_N^2 and Γ_N^3.

For 2-level and 3-level 4-encoder MDCS instances, we can still observe that if scalar binary codes suffice, then scalar ternary will also be sufficient and if scalar binary codes do not suffice then neither will scalar ternary codes. In addition, we observe that for all MDCS instances we considered, the scalar ternary inner bounds match exactly with the matroid inner bound. Therefore, we have the following observation:

Observation 1: If there exists some field size such that scalar linear codes over that field obtain the entire rate region then in all 7360 MDCS instances we considered, that field size may be taken to be binary.

However, ternary codes do not give same rate regions as binary codes for some cases when neither of them suffice, since some networks (or points in the rate region) can be achievable by scalar ternary codes but not by scalar binary codes.

Example 2: One example network where ternary codes give tighter inner bound is shown in Fig. 2.5.

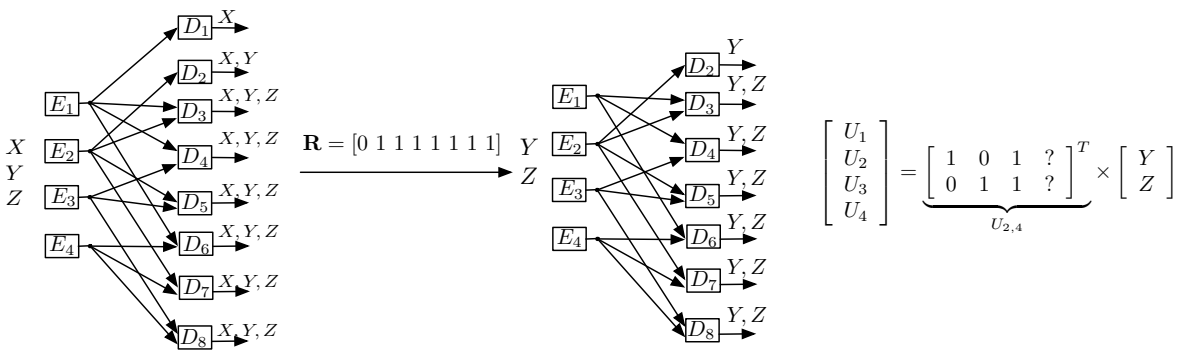


Figure 2.5: A $(3,4)$ MDCS instance for which neither scalar binary codes nor scalar ternary codes suffice, but scalar ternary codes give a tighter inner bound than do scalar binary codes. The extreme ray which is not scalar binary achievable is shown. It is not scalar binary achievable because in the simplified network, a code associated with $U_{2,4}$, which is the forbidden minor for a matroid to be binary achievable, is required.

The outer bound on the rate region $\mathcal{R}_{\text{out}}$ is

$$\mathcal{R}_{\text{out}} = \left\{ \mathbf{R} : \begin{array}{rcl} R_1 & \geq & H(X) \\ R_2 & \geq & H(X) + H(Y) \\ R_1 + R_2 & \geq & 2H(X) + H(Y) + H(Z) \\ R_3 + R_4 & \geq & H(X) + H(Y) + H(Z) \\ R_1 + R_3 & \geq & H(X) + H(Y) + H(Z) \\ R_1 + R_4 & \geq & H(X) + H(Y) + H(Z) \\ R_2 + R_4 & \geq & H(X) + H(Y) + H(Z) \\ R_2 + R_3 & \geq & H(X) + H(Y) + H(Z) \end{array} \right\}. \tag{2.120}$$

The binary achievable rate region $\mathcal{R}_{\text{bin}}$ is

$$\mathcal{R}_{\text{bin}} = \mathcal{R}_{\text{out}} \cap \left\{ \mathbf{R} : \begin{array}{rcl} R_1 + R_2 + R_3 & \geq & 2H(X) + 2H(Y) + 2H(z) \\ R_1 + R_2 + R_4 & \geq & 2H(X) + 2H(Y) + 2H(z) \\ R_1 + R_3 + R_4 & \geq & 2H(X) + 2H(Y) + 2H(z) \\ R_2 + R_3 + R_4 & \geq & 3H(X) + 3H(Y) + 3H(z) \\ R_1 + R_2 + R_3 + R_4 & \geq & 2H(X) + 2H(Y) + 2H(z) \end{array} \right\}, \tag{2.121}$$

while the ternary achievable rate region $\mathcal{R}_{\text{ter}}$, which is tighter than binary achievable rate region, is

$$\mathcal{R}_{\text{ter}} = \mathcal{R}_{\text{out}} \cap \left\{ \mathbf{R}: \begin{array}{rcl} R_1+R_2+R_3 & \geq & 2H(X)+H(Y)+2H(z) \\ R_1+R_2+R_4 & \geq & 2H(X)+H(Y)+2H(z) \\ R_1+R_3+R_4 & \geq & 2H(X)+H(Y)+2H(z) \\ R_1+R_3+R_4 & \geq & 2H(X)+2H(Y)+H(z) \\ 2R_1+R_3+R_4 & \geq & 3H(X)+2H(Y)+2H(z) \\ R_2+R_3+R_4 & \geq & 2H(X)+H(Y)+2H(z) \\ R_1+R_2+R_3+R_4 & \geq & 3H(X)+H(Y)+3H(z) \end{array} \right\}. \tag{2.122}$$

The extreme ray in the ternary bound that violates additional inequalities in the binary bound is

$$\mathbf{R} = [H(X)\ H(Y)\ H(Z)\ R_1\ R_2\ R_3\ R_4] = [0\ 1\ 1\ 1\ 1\ 1\ 1].$$

It is not hard to see why this extreme ray is not binary achievable but is ternary achievable. We can assign the entropies in the extreme ray $\mathbf{R}$ to the variables in the network shown in Fig. 2.5. Since $H(X) = 0$, it is equivalent to delete source X. The network is then simplified to a $(2, 4)$ MDCS instance where the six decoders receiving messages from size-two subsets of encoders demand the two sources Y, Z. It is known that sources Y, Z are independent, so if a binary code achieves this extreme ray, every collection of two codewords must be able to decode Y, Z, i.e., the joint entropy of every two codewords is 2. Equivalently, the matroids associated with achieving codes must contain $U_{2,4}$ as a minor. However, it is known that any matroid containing $U_{2,4}$ is not binary representable, as shown in Fig. 2.5. Therefore, this network is not binary achievable.

This example shows that for an extreme ray (or point) in the rate region of a network, if the matroids associated with its achieving codes are not binary representable, we conclude that the rate region is not scalar binary sufficient. In general, we have the following theorem.

Theorem 9: Let $\mathcal{R}$ be the rate region of a network with N random variables and $\text{Extr}(\mathcal{R})$ represent the (minimum integer) extreme rays of $\mathcal{R}$. Linear codes in $\mathbb{F}_q$ suffice to achieve $\mathcal{R}$ if and only if $\forall \mathbf{R} \in \text{Extr}(\mathcal{R})$, $\exists\, \mathbf{R}' \in \Gamma^q_{N,N'}$, for some $N' \geq N$, such that $\mathbf{R}'$ satisfies all network constraints and $\mathbf{R} = \text{Proj}_{h_{X_k}, h_{U_e}, k\in[[K]], e\in\mathcal{E}}(\mathbf{R}')$.

Proof: First we know that if achieving codes for two points in the rate region are given, time sharing between these two codes can achieve any point between theses two points. Therefore, it

suffices to only consider the achievability of extreme rays (points) of the rate region.

For an extreme ray $\mathbf{R} \in \text{Extr}(\mathcal{R})$ scaled to minimum integer representation, if $\exists\ \mathbf{R}' \in \Gamma^q_{N,N'}$, for some $N' \geq N$, such that $\mathbf{R}'$ satisfies all network constraints and $\mathbf{R} = \text{Proj}_{h_{U_e}, h_{X_k}, e\in\mathcal{E}, k\in[[K]]}\mathbf{R}'$, we can construct the code to achieve it following the method in §2.4 using the corresponding representation for $\mathbf{R}'$. Note that when $N' = N$, we construct scalar codes and when $N' > N$, we construct vector codes. Thus, $\mathbf{R}$ is achievable by $\mathbb{F}_q$ linear codes and so is the entire region $\mathcal{R}$. ■

From this theorem we see that, in order to prove a network that is not $\mathbb{F}_q$ representable, one just needs to show the non-existence of $\mathbb{F}_q$ codes for one extreme ray in the rate region.

Fig. 2.3 shows an example where scalar-binary codes are not optimal. Different from the proof in §2.4, one alternate proof for insufficiency of scalar-binary codes works as follows.

Alternate proof for scalar binary insufficiency of example in Fig. 2.3: We are given that the (Shannon outer bound on) rate region for this MDCS instance is $\mathcal{R}$:

$$\mathcal{R} = \left\{ \mathbf{R} : \begin{array}{rcl} R_1 & \geq & H(X) \\ R_2 + R_3 & \geq & H(X) + H(Y) \\ R_1 + R_3 & \geq & H(X) + H(Y) \\ R_1 + R_2 & \geq & H(X) + H(Y) \end{array} \right\}. \tag{2.123}$$

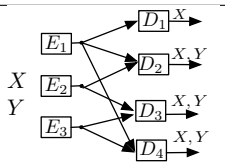
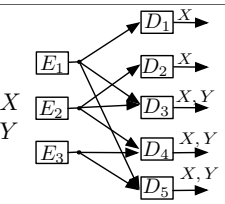
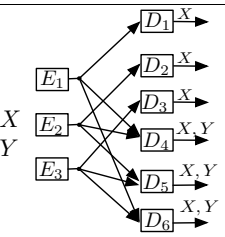
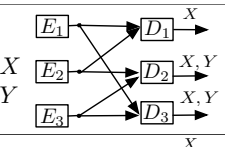
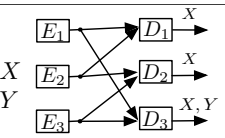
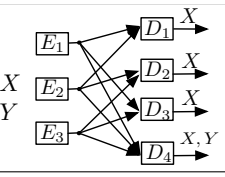
One extreme ray of it is $\mathbf{R} = [0\ 2\ 1\ 1\ 1]$ with the entries corresponding to $(H(X)\ H(Y)\ R_1\ R_2\ R_3)$ respectively. Since $H(Y) = 2$, there does not exist a scalar binary code such that the coded messages have entropy 1. This completes the proof. ■

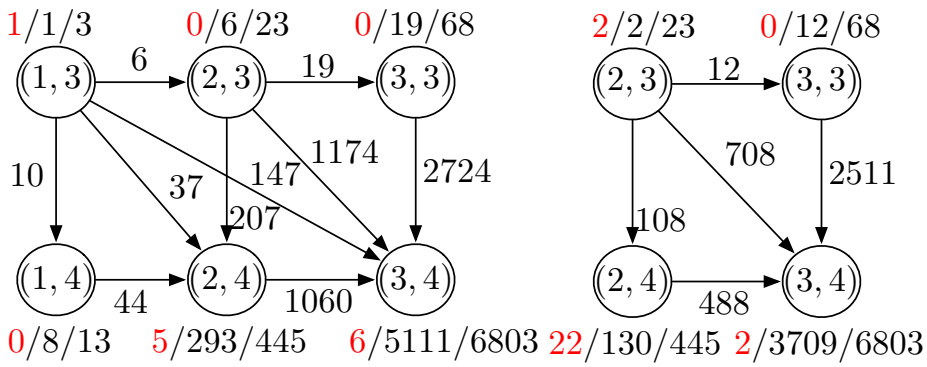
Similarly, one can prove non-achievability of binary codes for other networks by showing the non-achievability of some extreme ray in the rate region. Due to the large number of cases and our limited space, only the proofs for the 6 cases in $(2, 3)$ MDCS instances, where binary codes are not optimal, are shown in Table 2.3 as examples.

2.6.4 Sufficiency of vector codes

As mentioned above, simple scalar codes are not always sufficient to achieve every point in the rate region, even for these simple small MDCS instances. A natural alternative is to employ vector linear codes instead, which means encoding a group of outcomes of source variables for several time steps together. Recall that vector linear codes are corresponding to vector representable matroids by grouping elements together as one new variable. As we will show in §2.4, the construction of codes are based on the representation matrix of the original matroid before grouping elements. Therefore,

Table 2.3: Six $(2,3)$ *MDCS instances where scalar binary inner bound and Shannon outer bound do not match. The inequalities where outer bound and inner bound do not match are in bold. The listed extreme rays with entries corresponding to* $[H(X)\ H(Y)\ R_1\ R_2\ R_3]$ *violate bolded inequalities of binary inner bounds.*

Case diagrams	Exact rate regions	Scalar binary inner Bounds	Violating extreme rays
	$R_1 \geq H(X)$ $R_1+R_2 \geq H(X)+H(Y)$ $R_1+R_3 \geq H(X)+H(Y)$ $R_2+R_3 \geq H(X)+H(Y)$	$R_1 \geq H(X)$ $R_1+R_2 \geq H(X)+H(Y)$ $R_1+R_3 \geq H(X)+H(Y)$ $R_2+R_3 \geq H(X)+H(Y)$ $\mathbf{R_1+R_2+R_3 \geq H(X)+2H(Y)}$	[0 2 1 1 1]
	$R_1 \geq H(X)$ $R_2 \geq H(X)$ $R_1+R_2 \geq 2H(X)+H(Y)$ $R_1+R_3 \geq H(X)+H(Y)$ $R_2+R_3 \geq H(X)+H(Y)$	$R_1 \geq H(X)$ $R_2 \geq H(X)$ $R_1+R_2 \geq 2H(X)+H(Y)$ $R_1+R_3 \geq H(X)+H(Y)$ $R_2+R_3 \geq H(X)+H(Y)$ $\mathbf{R_1+R_2+R_3 \geq 2H(X)+2H(Y)}$	[0 2 1 1 1]
	$R_1 \geq H(X)$ $R_2 \geq H(X)$ $R_3 \geq H(X)$ $R_1+R_2 \geq 2H(X)+H(Y)$ $R_1+R_3 \geq 2H(X)+H(Y)$ $R_2+R_3 \geq 2H(X)+H(Y)$	$R_1 \geq H(X)$ $R_2 \geq H(X)$ $R_3 \geq H(X)$ $R_1+R_2 \geq 2H(X)+H(Y)$ $R_1+R_3 \geq 2H(X)+H(Y)$ $R_2+R_3 \geq 2H(X)+H(Y)$ $\mathbf{R_1+R_2+R_3 \geq 3H(X)+2H(Y)}$	[0 2 1 1 1]
	$R_1+R_2 \geq H(X)$ $R_1+R_3 \geq H(X)+H(Y)$ $R_2+R_3 \geq H(X)+H(Y)$	$R_1+R_2 \geq H(X)$ $R_1+R_3 \geq H(X)+H(Y)$ $R_2+R_3 \geq H(X)+H(Y)$ $\mathbf{R_1+R_2+R_3 \geq 2H(X)+H(Y)}$	[2 0 1 1 1]
	$R_1+R_2 \geq H(X)$ $R_1+R_3 \geq H(X)+H(Y)$ $R_2+R_3 \geq H(X)$ $\mathbf{R_1+2R_2+R_3 \geq 2H(X)+H(Y)}$	$R_1+R_2 \geq H(X)$ $R_1+R_3 \geq H(X)+H(Y)$ $R_2+R_3 \geq H(X)$ $\mathbf{R_1+R_2+R_3 \geq 2H(X)+H(Y)}$	[2 0 1 1 1]
	$R_1+R_2 \geq H(X)$ $R_1+R_3 \geq H(X)$ $R_2+R_3 \geq H(X)$ $\mathbf{R_1+R_2+2R_3 \geq 2H(X)+H(Y)}$ $\mathbf{R_1+2R_2+2R_3 \geq 2H(X)+H(Y)}$ $\mathbf{2R_1+R_2+R_3 \geq 2H(X)+H(Y)}$ $\mathbf{3R_1+2R_2+2R_3 \geq 3H(X)+2H(Y)}$	$R_1+R_2 \geq H(X)$ $R_1+R_3 \geq H(X)$ $R_2+R_3 \geq H(X)$ $\mathbf{R_1+R_2+R_3 \geq 2H(X)+H(Y)}$	[2 0 1 1 1]



(a) Scalar binary codes (b) Superposition coding

Figure 2.6: Nesting relationships between $(1,3),(1,4),(2,3),(3,3),(2,4),(3,4)$ *MDCS instances regarding the sufficiency of classes of codes. The three numbers at each node indicate the number of forbidden minors, number of instances that the codes do not suffice, and the total number of non-isomorphic MDCS instances. The numbers on every edge indicates how many scalar binary insufficient head MDCS instances have predecessors in the tail MDCS instances.*

we allow coding occurs not only between different portions of a source (i.e, superposition) but also portions of different sources (i.e, vector linear combinations).

Passing from scalar codes to vector codes, in this sense, by replacing Γ_N^2 with $\Gamma_{N,N+1}^2, N = 5, 6$ in our 2-level 3-encoder and 3-level 3-encoder achievable rate regions, closes all of gaps, hence proving that the exact rate regions for 2-level 3-encoder and 3-level 3-encoder MDCS instances are the same as that obtained from the Shannon outer bound. This proves that vector linear codes (in the sense of §2.4) suffice to achieve all of the fundamental rate regions of 2-level 3-encoder and 3-level 3-encoder MDCS instances. Also note that, only one extra bit is necessary to use as vector binary codes to achieve the entire rate regions for 2-level and 3-level 3-encoder MDCS instances.

With up to three extra bits, we are able to close gaps for almost all of the 455 $(2,4)$ and 6803 $(3,4)$ MDCS instances, except a small fraction of them, which can be further reduced if vector ternary inner bounds or vector binary inner bounds with more extra bits are applied. The example presented in §2.4 shows the benefit of using vector linear codes instead of scalar codes in obtaining exact rate regions.

2.6.5 Forbidden Minors for Code Class Sufficiency

Fig. 2.6 summarizes our observations on the relationships between $(1,3),(1,4),(2,3),(2,4),(3,3),(3,4)$ MDCS instances in terms of the sufficiency of different classes of codes, including scalar binary codes and superposition coding.

Fig. 2.6(a) shows the relationships regarding the sufficiency of scalar binary codes. The operations we utilized were source deletion, encoder deletion, and encoder contraction. We observed that all

19 binary insufficient $(3,3)$ MDCS instances have minors of one of the 6 scalar binary insufficient $(2,3)$ MDCS instances, which themselves all have the same predecessor, the scalar binary insufficient $(1,3)$ MDCS instance. All the 8 scalar binary insufficient $(1,4)$ MDCS instances also have the same predecessor, the scalar binary insufficient $(1,3)$ MDCS instance. However, for $(2,4)$ ($(3,4)$) MDCS, there are 5 (6) instances that we cannot find predecessors for them. We list the 12 forbidden network minors for scalar binary sufficiency in Table 2.4 and Table 2.5. If encoder unification is also considered, the number of forbidden minors can be reduced to 10 in total.

Fig. 2.6(b) shows the relationships regarding the sufficiency of superposition codes. The operations we utilized include source deletion, encoder deletion, encoder contraction, and encoder unification. We observed that all 12 superposition insufficient $(3,3)$ MDCS instances have minors of one of the 2 superposition insufficient $(2,3)$ MDCS instances. However, for $(2,4)$ ($(3,4)$) MDCS, there are 22 (2) instances that we cannot find predecessors for them. The numbers on every edge indicates how many superposition insufficient head MDCS instances have predecessors in the tail MDCS instances.

2.7 Computer aided converse proof

Recall that an inequality in the rate region of a network is in terms of rate variables and source entropies. Sometimes, the inequalities in an MDCS rate region are easy to derive. However, for many MDCS instances, it is not easy to derive by hand all of the inequalities for the rate region. For instance, as was shown in the MDCS instances discussed in §2.2, it is tedious, and some instances difficult, to calculate the converses for all non-isomorphic MDCS instances of even small problem sizes manually. Motivated by these observations, and inspired by a technique discussed in [5, 25], in this section, we will show how to use a computer to generate converse proofs automatically.

As discussed in §2.3, the projection of the Shannon outer bound plus network constraints gives an outer bound on the coding rate region, which is in principle equivalent to a converse proof. This can be seen from a property of the polar of a polyhedral cone (c.f. Proposition $B.16.d$ in [38] and page 122 in [39]): suppose a polyhedral cone $\mathcal{P}$ is determined by $\mathbb{A}\mathbf{x} \geq \mathbf{0}$ and for an inequality implied by the projected cone, $\mathbf{b}^T\mathbf{x} \geq 0$, where $\mathbf{b}^T$ has zeros on eliminated dimensions, then there exists a vector $\boldsymbol{\lambda} \geq \mathbf{0}$ such that $\mathbb{A}^T\boldsymbol{\lambda} = \mathbf{b}$. The vector $\boldsymbol{\lambda}$ gives coefficients for the weighted sum of inequalities in $\mathcal{P}$.

In the calculation of the Shannon outer bound on rate regions, the Shannon outer bound and network constraints form a polyhedral cone $\mathcal{P} = \{\mathbf{x}|\mathbb{A}\mathbf{x} \geq \mathbf{0}\}$ in the dimension of $2^N - 1 + |\mathcal{E}|$, which

Table 2.4: The only one $(1,3)$ *and 5* $(2,4)$ *MDCS instances for which scalar binary codes do not suffice. The inequalities where outer bound and inner bound do not match are in bold. The listed extreme rays with entries corresponding to* $[H(X_k), k \in [[K]]\ R_e, e \in \mathcal{E}]$ *violate bolded inequalities of binary inner bounds.*

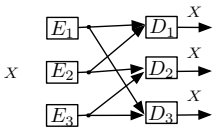
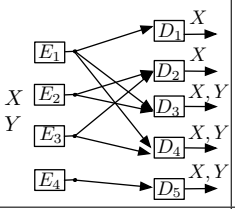
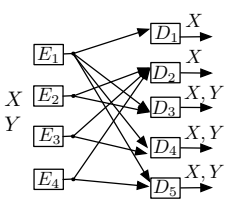
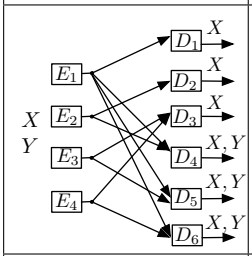
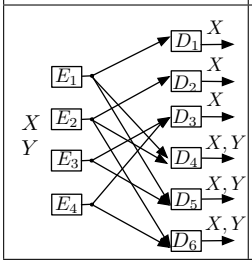
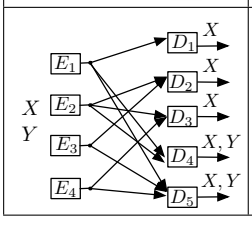
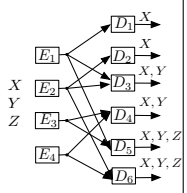
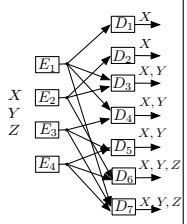
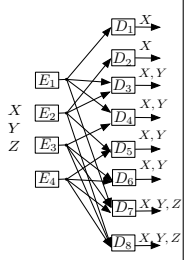
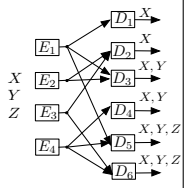
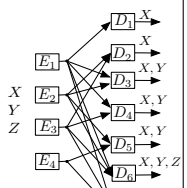
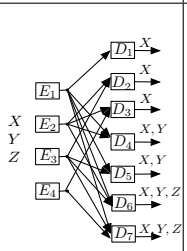
Case diagrams	Exact rate regions	Scalar binary inner Bounds	Violating rays
	$R_1+R_2 \geq H(X)$ $R_1+R_3 \geq H(X)$ $R_2+R_3 \geq H(X)$	$R_1+R_2 \geq H(X)$ $R_1+R_3 \geq H(X)$ $R_2+R_3 \geq H(X)$ $\mathbf{R_1+R_2+R_3 \geq 2H(X)}$	$\begin{bmatrix} 2 \\ 1 \\ 1 \\ 1 \end{bmatrix}$
	$R_1 \geq H(X)$ $R_1+R_2 \geq H(X)+H(Y)$ $R_1+R_3 \geq H(X)+H(Y)$ $R_2+R_3 \geq H(X)$ $R_4 \geq H(X)+H(Y)$ $R_1+R_2+R_3 \geq 2H(X)+H(Y)$	$R_1 \geq H(X)$ $R_1+R_2 \geq H(X)+H(Y)$ $R_1+R_3 \geq H(X)+H(Y)$ $R_2+R_3 \geq H(X)$ $R_4 \geq H(X)+H(Y)$ $\mathbf{2R_1+R_2+R_3 \geq 3H(X)+2H(Y)}$	$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 2 \end{bmatrix}$
	$R_1 \geq H(X)$ $R_2+R_3+R_4 \geq H(X)$ $R_1+R_2 \geq H(X)+H(Y)$ $R_1+R_3 \geq H(X)+H(Y)$ $R_1+R_4 \geq H(X)+H(Y)$ $R_1+R_2+R_3+R_4 \geq 2H(X)+H(Y)$	$R_1 \geq H(X)$ $R_2+R_3+R_4 \geq H(X)$ $R_1+R_2 \geq H(X)+H(Y)$ $R_1+R_3 \geq H(X)+H(Y)$ $R_1+R_4 \geq H(X)+H(Y)$ $\mathbf{2R_1+R_2+R_3 \geq 3H(X)+2H(Y)}$	$\begin{bmatrix} 2 \\ 1 \\ 2 \\ 1 \\ 1 \\ 1 \end{bmatrix}$
	$R_1 \geq H(X)$ $R_2 \geq H(X)$ $R_3+R_4 \geq H(X)$ $R_1+R_2 \geq 2H(X)+H(Y)$ $R_1+R_3 \geq H(X)+H(Y)$ $R_1+R_4 \geq H(X)+H(Y)$ $R_1+R_3+R_4 \geq 2H(X)+H(Y)$	$R_1 \geq H(X)$ $R_2 \geq H(X)$ $R_3+R_4 \geq H(X)$ $R_1+R_2 \geq 2H(X)+H(Y)$ $R_1+R_3 \geq H(X)+H(Y)$ $R_1+R_4 \geq H(X)+H(Y)$ $\mathbf{2R_1+R_3+R_4 \geq 3H(X)+2H(Y)}$	$\begin{bmatrix} 1 \\ 2 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$
	$R_1 \geq H(X)$ $R_2 \geq H(X)$ $R_3+R_4 \geq H(X)$ $R_1+R_3 \geq H(X)+H(Y)$ $R_2+R_4 \geq H(X)+H(Y)$ $R_1+R_2 \geq 2H(X)+H(Y)$ $R_1+R_3+R_4 \geq 2H(X)+H(Y)$ $R_2+R_3+R_4 \geq 2H(X)+H(Y)$	$R_1 \geq H(X)$ $R_2 \geq H(X)$ $R_3+R_4 \geq H(X)$ $R_1+R_3 \geq H(X)+H(Y)$ $R_2+R_4 \geq H(X)+H(Y)$ $R_1+R_2 \geq 2H(X)+H(Y)$ $R_1+R_3+R_4 \geq 2H(X)+H(Y)$ $R_2+R_3+R_4 \geq 2H(X)+H(Y)$ $\mathbf{R_1+2R_2+R_3+R_4 \geq 3H(X)+2H(Y)}$	$\begin{bmatrix} 1 \\ 2 \\ 2 \\ 2 \\ 1 \\ 1 \end{bmatrix}$
	$R_1 \geq H(X)$ $R_2+R_3 \geq H(X)$ $R_2+R_4 \geq H(X)$ $R_1+R_2 \geq H(X)+H(Y)$ $R_1+R_3+R_4 \geq H(X)+H(Y)$ $R_1+R_2+R_3 \geq 2H(X)+H(Y)$ $R_1+R_2+R_4 \geq 2H(X)+H(Y)$	$R_1 \geq H(X)$ $R_2+R_3 \geq H(X)$ $R_2+R_4 \geq H(X)$ $R_1+R_2 \geq H(X)+H(Y)$ $R_1+R_3+R_4 \geq H(X)+H(Y)$ $R_1+R_2+R_3 \geq 2H(X)+H(Y)$ $R_1+R_2+R_4 \geq 2H(X)+H(Y)$ $\mathbf{2R_1+R_2+R_3 \geq 3H(X)+2H(Y)}$	$\begin{bmatrix} 1 \\ 2 \\ 1 \\ 2 \\ 1 \\ 1 \end{bmatrix}$

Table 2.5: The six $(3,4)$ *MDCS instances for which scalar binary codes do not suffice. The inequalities where outer bound and inner bound do not match are in bold. The listed extreme rays with entries corresponding to* $[H(X_k), k \in [[K]]\ R_e, e \in \mathcal{E}]$ *violate bolded inequalities of binary inner bounds.*

Case diagrams	Shannon outer bounds	Scalar binary inner Bounds	Violating ray
	$R_1 \geq H(X)$ $R_2 \geq H(X)$ $R_1 + R_2 \geq 2H(X) + H(Y)$ $R_3 + R_4 \geq H(X) + H(Y)$ $R_1 + R_3 \geq H(X) + H(Y) + H(Z)$ $R_2 + R_4 \geq H(X) + H(Y) + H(Z)$ $R_1 + R_3 + R_4 \geq 2H(X) + H(Y) + H(Z)$ $R_2 + R_3 + R_4 \geq 2H(X) + H(Y) + H(Z)$ $R_1 + R_2 + R_3 + R_4 \geq 3H(X) + 2H(Y) + H(Z)$	$R_1 \geq H(X)$ $R_2 \geq H(X)$ $R_1 + R_2 \geq 2H(X) + H(Y)$ $R_3 + R_4 \geq H(X) + H(Y)$ $R_1 + R_3 \geq H(X) + H(Y) + H(Z)$ $R_2 + R_4 \geq H(X) + H(Y) + H(Z)$ $R_1 + R_3 + R_4 \geq 2H(X) + H(Y) + H(Z)$ $R_2 + R_3 + R_4 \geq 2H(X) + H(Y) + H(Z)$ $\mathbf{R_1 + R_2 + 2R_3 + 2R_4 \geq 4H(X) + 3H(Y) + 2H(Z)}$	$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 2 \\ 2 \\ 1 \\ 1 \end{bmatrix}$
	$R_1 \geq H(X)$ $R_2 \geq H(X)$ $R_1 + R_2 \geq 2H(X) + H(Y)$ $R_1 + R_3 \geq H(X) + H(Y)$ $R_3 + R_4 \geq H(X) + H(Y)$ $R_2 + R_4 \geq H(X) + H(Y) + H(Z)$ $R_1 + R_3 + R_4 \geq 2H(X) + H(Y) + H(Z)$ $R_1 + 2R_3 + R_4 \geq 2H(X) + 2H(Y) + H(Z)$ $R_2 + R_3 + R_4 \geq 2H(X) + H(Y) + H(Z)$ $R_1 + R_2 + R_3 + R_4 \geq 3H(X) + 2H(Y) + H(Z)$	$R_1 \geq H(X)$ $R_2 \geq H(X)$ $R_1 + R_2 \geq 2H(X) + H(Y)$ $R_1 + R_3 \geq H(X) + H(Y)$ $R_3 + R_4 \geq H(X) + H(Y)$ $R_2 + R_4 \geq H(X) + H(Y) + H(Z)$ $R_1 + R_3 + R_4 \geq 2H(X) + H(Y) + H(Z)$ $R_1 + 2R_3 + R_4 \geq 2H(X) + 2H(Y) + H(Z)$ $R_2 + R_3 + R_4 \geq 2H(X) + H(Y) + H(Z)$ $\mathbf{R_1 + R_2 + R_3 + R_4 \geq 4H(X) + 3H(Y) + 2H(Z)}$	$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 2 \\ 2 \\ 1 \\ 1 \end{bmatrix}$
	$R_1 \geq H(X)$ $R_2 \geq H(X)$ $R_1 + R_2 \geq 2H(X) + H(Y)$ $R_1 + R_3 \geq H(X) + H(Y)$ $R_2 + R_4 \geq H(X) + H(Y)$ $R_3 + R_4 \geq H(X) + H(Y)$ $R_1 + R_3 + R_4 \geq 2H(X) + H(Y) + H(Z)$ $R_1 + 2R_3 + R_4 \geq 2H(X) + 2H(Y) + H(Z)$ $R_2 + R_3 + R_4 \geq 2H(X) + H(Y) + H(Z)$ $R_2 + 2R_3 + R_4 \geq 2H(X) + H(Y) + H(Z)$ $R_1 + R_2 + R_3 + R_4 \geq 3H(X) + 2H(Y) + H(Z)$ $R_1 + R_2 + 2R_3 + R_4 \geq 3H(X) + 3H(Y) + 2H(Z)$	$R_1 \geq H(X)$ $R_2 \geq H(X)$ $R_1 + R_2 \geq 2H(X) + H(Y)$ $R_1 + R_3 \geq H(X) + H(Y)$ $R_2 + R_4 \geq H(X) + H(Y)$ $R_3 + R_4 \geq H(X) + H(Y)$ $R_1 + R_3 + R_4 \geq 2H(X) + H(Y) + H(Z)$ $R_1 + 2R_3 + R_4 \geq 2H(X) + 2H(Y) + H(Z)$ $R_2 + R_3 + R_4 \geq 2H(X) + H(Y) + H(Z)$ $R_2 + 2R_3 + R_4 \geq 2H(X) + H(Y) + H(Z)$ $\mathbf{R_1 + R_2 + 2R_3 + 2R_4 \geq 4H(X) + 3H(Y) + 2H(Z)}$	$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 2 \\ 2 \\ 1 \\ 1 \end{bmatrix}$
	$R_1 \geq H(X)$ $R_2 + R_3 \geq H(X)$ $R_1 + R_2 \geq H(X) + H(Y)$ $R_4 \geq H(X) + H(Y)$ $R_1 + R_4 \geq 2H(X) + H(Y) + H(Z)$ $R_3 + R_4 \geq H(X) + H(Y) + H(Z)$ $R_1 + R_2 + R_3 \geq 2H(X) + H(Y)$ $R_1 + R_2 + R_4 \geq 2H(X) + 2H(Y) + H(Z)$ $R_2 + R_3 + R_4 \geq 2H(X) + H(Y) + H(Z)$ $R_1 + R_2 + R_3 + R_4 \geq 3H(X) + 2H(Y) + H(Z)$	$R_1 \geq H(X)$ $R_2 + R_3 \geq H(X)$ $R_1 + R_2 \geq H(X) + H(Y)$ $R_4 \geq H(X) + H(Y)$ $R_1 + R_4 \geq 2H(X) + H(Y) + H(Z)$ $R_3 + R_4 \geq H(X) + H(Y) + H(Z)$ $R_1 + R_2 + R_3 \geq 2H(X) + H(Y)$ $R_1 + R_2 + R_4 \geq 2H(X) + 2H(Y) + H(Z)$ $R_2 + R_3 + R_4 \geq 2H(X) + H(Y) + H(Z)$ $\mathbf{R_1 + R_2 + R_3 + 2R_4 \geq 4H(X) + 3H(Y) + 2H(Z)}$	$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 2 \\ 1 \\ 1 \\ 2 \end{bmatrix}$
	$R_1 \geq H(X)$ $R_2 + R_3 \geq H(X)$ $R_1 + R_2 \geq H(X) + H(Y)$ $R_1 + R_3 \geq H(X) + H(Y)$ $R_2 + R_4 \geq H(X) + H(Y)$ $R_1 + R_2 + R_3 \geq 2H(X) + H(Y) + H(Z)$ $R_1 + R_3 + R_4 \geq H(X) + 2H(Y) + H(Z)$ $2R_1 + R_2 + R_3 \geq 2H(X) + 2H(Y) + H(Z)$ $R_1 + R_2 + R_3 + R_4 \geq 2H(X) + 2H(Y) + H(Z)$	$R_1 \geq H(X)$ $R_2 + R_3 \geq H(X)$ $R_1 + R_2 \geq H(X) + H(Y)$ $R_1 + R_3 \geq H(X) + H(Y)$ $R_2 + R_4 \geq H(X) + H(Y)$ $R_1 + R_2 + R_3 \geq 2H(X) + H(Y) + H(Z)$ $R_1 + R_3 + R_4 \geq H(X) + 2H(Y) + H(Z)$ $2R_1 + R_2 + R_3 \geq 2H(X) + 2H(Y) + H(Z)$ $R_1 + R_2 + R_3 + R_4 \geq 2H(X) + 2H(Y) + H(Z)$ $\mathbf{3R_1 + R_2 + 2R_3 + R_4 \geq 4H(X) + 3H(Y) + 2H(Z)}$	$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 2 \\ 1 \\ 1 \end{bmatrix}$
	$R_1 \geq H(X)$ $R_2 + R_3 \geq H(X)$ $R_2 + R_4 \geq H(X)$ $R_1 + R_2 \geq H(X) + H(Y)$ $R_1 + R_3 \geq H(X) + H(Y)$ $R_1 + R_2 + R_4 \geq 2H(X) + H(Y)$ $R_1 + R_2 + R_3 \geq 2H(X) + H(Y) + H(Z)$ $R_1 + R_3 + R_4 \geq H(X) + H(Y) + H(Z)$ $2R_1 + R_2 + R_3 \geq 2H(X) + 2H(Y) + H(Z)$ $2R_1 + R_2 + R_3 + R_4 \geq 3H(X) + 2H(Y) + H(Z)$	$R_1 \geq H(X)$ $R_2 + R_3 \geq H(X)$ $R_2 + R_4 \geq H(X)$ $R_1 + R_2 \geq H(X) + H(Y)$ $R_1 + R_3 \geq H(X) + H(Y)$ $R_1 + R_2 + R_4 \geq 2H(X) + H(Y)$ $R_1 + R_2 + R_3 \geq 2H(X) + H(Y) + H(Z)$ $R_1 + R_3 + R_4 \geq H(X) + H(Y) + H(Z)$ $2R_1 + R_2 + R_3 \geq 2H(X) + 2H(Y) + H(Z)$ $\mathbf{3R_1 + R_2 + 2R_3 + R_4 \geq 4H(X) + 3H(Y) + 2H(Z)}$	$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 2 \\ 1 \\ 1 \end{bmatrix}$

is then projected onto only $|\mathcal{E}| + K$ dimensions, as shown in (2.33). After this projection, the rate region, which must still be a polyhedral cone, has some inequality description $\mathbb{B}\mathbf{x} \geq \mathbf{0}$, where $\mathbb{B}_{:,j} = \mathbf{0}$, for all j-th columns such that $\mathbf{x}_j$ is eliminated in the projection. For each inequality $\mathbf{b}^T\mathbf{x} \geq 0$ in the rate region, the weighted sum $\mathbf{b} = \boldsymbol{\lambda}^T\mathbb{A}$ with coefficients $\boldsymbol{\lambda}$ is actually the converse proof for this inequality, deriving it as a sum of network constraints and Shannon information inequalities.

Such a coefficient vector $\boldsymbol{\lambda}$ need not be unique. Hence, among the various possibilities for $\boldsymbol{\lambda}$ satisfying $\mathbf{b} = \boldsymbol{\lambda}^T\mathbb{A}$, we want to select one that gives the simplest proof, which means it involves the smallest number of inequalities and constraints. This vector will thus be the sparsest vector $\boldsymbol{\lambda}$, having the fewest non-zero entries, i.e. the lowest l_0-norm. Obtaining such a vector $\boldsymbol{\lambda}$ becomes the optimization problem

$$\begin{aligned} \underset{\boldsymbol{\lambda}}{\text{minimize}} \quad & \| \boldsymbol{\lambda} \|_0 \\ \text{subject to} \quad & \mathbb{A}^T\boldsymbol{\lambda} = \mathbf{b} \\ & \boldsymbol{\lambda} \geq \mathbf{0}. \end{aligned} \tag{2.124}$$

Because the l_0-norm is a complicated non-differentiable and non-linear function, this is a difficult optimization problem to solve directly. Hence we use the typical relaxation that replaces the l_0-norm with the l_1-norm. Furthermore, since $\boldsymbol{\lambda} \geq \mathbf{0}$, we do not need the absolute values in the l_1-norm, and the problem in (2.124) is approximated by the linear program

$$\begin{aligned} \underset{\boldsymbol{\lambda}}{\text{minimize}} \quad & \| \boldsymbol{\lambda} \|_1 \\ \text{subject to} \quad & \mathbb{A}^T\boldsymbol{\lambda} = \mathbf{b} \\ & \boldsymbol{\lambda} \geq \mathbf{0}, \end{aligned} \tag{2.125}$$

which is equivalent to

$$\begin{aligned} \underset{\boldsymbol{\lambda}}{\text{minimize}} \quad & \mathbf{1}^T\boldsymbol{\lambda} \\ \text{subject to} \quad & \mathbb{A}^T\boldsymbol{\lambda} = \mathbf{b} \\ & \boldsymbol{\lambda} \geq \mathbf{0}. \end{aligned} \tag{2.126}$$

If the goal is solely to calculate the rate region, when projecting Shannon outer bound and network constraints, it is best to start with only the non-redundant inequalities to reduce the complexity of the projection calculation. However, if we want to generate the human readable proof, by solving the problem in (2.126), including many redundant information inequalities can aid us in finding a sparser vector $\boldsymbol{\lambda}$ than the process without redundant inequalities. In particular, the non-redundant form of the Shannon outer bound only involves inequalities of the form $I(X_i, X_j|\mathbf{X}_{\mathcal{K}})$ and

$H(X_i|\mathbf{X}_{\setminus i}) \geq 0$. Rather than expressing other Shannon information inequalities as sums of these elemental inequalities, it is preferable when solving (2.126) to add them to the list of inequalities in the Shannon outer bound, even though they are redundant, because they can lead to sparser $\boldsymbol{\lambda}$, and hence simpler human readable proofs. Similarly, some redundant inequalities from the network constraints are also helpful. For instance, for a constraint $H(\mathcal{A}|\mathcal{B}) = 0$, the redundant equalities $H(\mathcal{C}|\mathcal{B}) = 0, \mathcal{C} \subseteq \mathcal{A}$ can be very useful in generating short converse proofs.

After the vector $\boldsymbol{\lambda}$ is obtained, let $\boldsymbol{\eta}$ be the non-zeros entries in $\boldsymbol{\lambda}$. We know that the weighted sum of the inequalities associated with $\boldsymbol{\eta}$ will give the desired inequality in the rate region to prove. However, such a potentially large weighted sum is still not a form that is easily interpreted by a human. In order to get a conventional converse proof as would have been done by hand, we also need to select an order to apply the inequalities identified by $\boldsymbol{\eta}$ step by step.

An algorithm to determine the order on involved coefficients and inequalities is shown in Algorithm 3. This algorithm always finishes because every term not shown in the target inequality must be cancelled out by the weighted sum of all of the yet unselected inequalities involving this term.

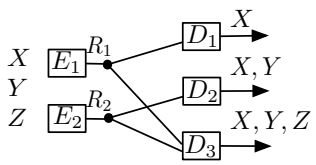
Input: Target inequality $\mathbf{b}^T\mathbf{x} \geq 0$, coefficient vector $\boldsymbol{\eta}$, involved original inequalities $\mathbb{A}_{\boldsymbol{\eta}}$ with row index set $\{1, 2, \ldots n\}$.
Output: Ordered index set of $\{1, 2, \ldots n\}$ that applying associated inequalities in $\mathbb{A}_{\boldsymbol{\eta}}$ gives human readable proof step by step.

Initialization: find inequalities which have non-zero entries at rate variables, put their indices to step 1;
mark these indices selected;
calculate the sum of inequalities in step 1 as the current inequality;
while *exists unselected indices* **do**
  get terms in current inequality that are not shown in target inequality;
  find inequalities which have non-zero entries at these terms;
  put their indices to next step and marked them as selected;
  calculate the weighted sum of current inequality and the newly selected inequalities to update current inequality;
end

Algorithm 3: Determination of the order of application of inequalities for a human readable proof.

Next, we would like to use two examples to show the procedure of obtaining the human readable converse proofs.

Example 3: A 3-level 2-encoder MDCS instance with block diagram and rate region shown in Fig. 2.7. The first two inequalities in the rate region are easy to prove and only very few inequalities are needed



$$\begin{aligned} R_1 &\geq H(X) \\ R_2 &\geq H(X)+H(Y) \\ R_1+R_2 &\geq 2H(X)+H(Y)+H(Z) \end{aligned}$$

Figure 2.7: Block diagram and rate region for Example 3: a 3-level 2-encoder MDCS instance

Table 2.6: Ordered inequalities with coefficients given by computer for Example 3

Step	Coefficients	Inequality or equality
1	1	$R_1 \geq H(U_1)$
1	1	$R_2 \geq H(U_2)$
2	1	$H(X\|U_1) = 0$
2	1	$H(X,Y\|U_2) = 0$
3	1	$I(U_1;YU_2\|X) \geq 0$
4	1	$H(X,Y\|U_1,U_2) = 0$
5	1	$H(X,Y,Z\|U_1,U_2) = 0$
6	1	$H(X,Y,Z) = H(X)+H(Y)+H(Z)$

for the converse proof. For the last inequality

$$R_1 + R_2 \geq 2H(X) + H(Y) + H(Z), \tag{2.127}$$

a computer, after running LP solver and Algorithm 3, could give inequalities with coefficients, and the order in Table 2.6. We can get the conventional converse proof by the steps given in Table 2.6.

Step 1, we start with

$$R_1 + R_2 \geq H(U_1) + H(U_2). \tag{2.128}$$

Then in step 2, we apply two equalities

$$H(X|U_1) = 0, \tag{2.129}$$

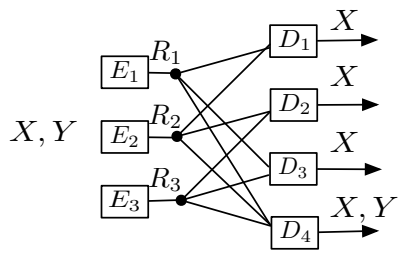
$$H(X,Y|U_2) = 0 \tag{2.130}$$

according to the decoding constraints of decoder D_1, D_2 and can get

$$R_1 + R_2 \geq H(U_1, X) + H(U_2, X, Y). \tag{2.131}$$

Next step, we use the trivial inequality

$$I(U_1; Y, U_2|X) \geq 0 \tag{2.132}$$



$$\begin{aligned} R_1 + R_2 &\geq H(X) \\ R_1 + R_3 &\geq H(X) \\ R_2 + R_3 &\geq H(X) \\ R_1 + 2R_2 + R_3 &\geq 2H(X) + H(Y) \\ R_1 + R_2 + 2R_3 &\geq 2H(X) + H(Y) \\ 2R_1 + R_2 + R_3 &\geq 2H(X) + H(Y) \\ 2R_1 + 2R_2 + 2R_3 &\geq 3H(X) + 2H(Y) \end{aligned}$$

Figure 2.8: Block diagram and rate region for Example 4: a 2-level 3-encoder MDCS instance

to get

$$R_1 + R_2 \geq H(X) + H(X, Y, U_1, U_2). \tag{2.133}$$

Since U_1, U_2 can decode X, Y, Z in decoder D_3, we can apply

$$H(X, Y|U_1, U_2) = 0 \tag{2.134}$$

$$H(X, Y, Z|U_1, U_2) = 0 \tag{2.135}$$

respectively in step 4 and step 5 to get

$$R_1 + R_2 \geq H(X) + H(X, Y, Z). \tag{2.136}$$

Finally, due to the independence of sources, we apply

$$H(X, Y, Z) = H(X) + H(Y) + H(Z) \tag{2.137}$$

to get the desired inequality and complete the proof.

Example 4: A 2-level 3-encoder MDCS instance with block diagram and rate region shown in Fig. 2.8.

We still pick the last inequality in the rate region as an example. The order and inequalities with coefficients for the converse proof are given in Table 2.7.

Step 1, we start with inequalities with coefficients

$$\begin{aligned} 2R_1 + 2R_2 + 2R_3 &\geq 2H(U_1) + 2H(U_2) + H(U_3) \\ &= (H(U_1) + H(U_2)) + (H(U_1) \\ &\quad + H(U_3)) + (H(U_2) + H(U_3)). \end{aligned}$$

Table 2.7: Ordered inequalities with coefficients given by computer for Example 4

Step	Coefficients	Inequality or equality
1	2	$R_1 \geq H(U_1)$
1	2	$R_2 \geq H(U_2)$
1	2	$R_3 \geq H(U_3)$
2	1	$I(U_1;U_2) \geq 0$
2	1	$I(U_1;U_3) \geq 0$
2	1	$I(U_2;U_3) \geq 0$
3	1	$H(X\|U_1,U_2) = 0$
3	1	$H(X\|U_1,U_3) = 0$
3	1	$H(X\|U_2,U_3) = 0$
4	1/3	$I(U_1;U_2,U_3\|X) \geq 0$
4	1/3	$I(U_1;U_2,U_3\|X) \geq 0$
4	1/3	$I(U_2;U_1,U_3\|X) \geq 0$
4	1/3	$I(U_1;U_2\|X,U_3) \geq 0$
4	1/3	$I(U_1;U_3\|X,U_2) \geq 0$
4	1/3	$I(U_2;U_3\|X,U_1) \geq 0$
5	2	$H(X\|U_1,U_2,U_3) = 0$
6	2	$H(X,Y\|U_1,U_2,U_3) = 0$
7	2	$H(X,Y) = H(X) + H(Y)$

Then in step 2, for each combined term, we apply the following three inequalities

$$I(U_1;U_2) \geq 0, \tag{2.138}$$

$$I(U_1;U_3) \geq 0, \tag{2.139}$$

$$I(U_2;U_3) \geq 0 \tag{2.140}$$

respectively to get

$$2R_1 + 2R_2 + 2R_3 \geq H(U_1,U_2) + H(U_1,U_3) + H(U_2,U_3). \tag{2.141}$$

Due to the decoding functions of D_1, D_2, D_3, we can use the three equalities

$$H(X|U_1,U_2) = 0, \tag{2.142}$$

$$H(X|U_1,U_3) = 0, \tag{2.143}$$

$$H(X|U_2,U_3) = 0 \tag{2.144}$$

to get

$$2R_1 + 2R_2 + 2R_3 \geq H(X,U_1,U_2) + H(X,U_1,U_3) + H(X,U_2,U_3). \tag{2.145}$$

Next, we notice that

$$I(U_1;U_2,U_3|X) = H(X,U_1) + H(X,U_2,U_3) - H(X,U_1,U_2,U_3) - H(X) \geq 0 \quad (2.146)$$

$$I(U_1;U_2,U_3|X) = H(X,U_1) + H(X,U_2,U_3) - H(X,U_1,U_2,U_3) - H(X) \geq 0 \quad (2.147)$$

$$I(U_2;U_1,U_3|X) = H(X,U_2) + H(X,U_1,U_3) - H(X,U_1,U_2,U_3) - H(X) \geq 0 \quad (2.148)$$

$$I(U_1;U_2|X,U_3) = H(X,U_1,U_3) + H(X,U_2,U_3) - H(X,U_1,U_2,U_3) - H(X,U_3) \geq 0 \quad (2.149)$$

$$I(U_1;U_3|X,U_2) = H(X,U_1,U_2) + H(X,U_2,U_3) - H(X,U_1,U_2,U_3) - H(X,U_2) \geq 0 \quad (2.150)$$

$$I(U_2;U_3|X,U_1) = H(X,U_1,U_2) + H(X,U_1,U_3) - H(X,U_1,U_2,U_3) - H(X,U_1) \geq 0 \quad (2.151)$$

The summation of (2.146) – (2.151) gives

$$3(H(X,U_1,U_2) + H(X,U_1,U_3) + H(X,U_2,U_3)) \geq 3(H(X) + 2H(X,U_1,U_2,U_3)) \quad (2.152)$$

Multiplying (2.152) with a coefficient $1/3$, we get

$$H(X,U_1,U_2) + H(X,U_1,U_3) + H(X,U_2,U_3) \geq H(X) + 2H(X,U_1,U_2,U_3). \quad (2.153)$$

Applying (2.153) to (2.145), we can get

$$2R_1 + 2R_2 + 2R_3 \geq H(X) + 2H(X,U_1,U_2,U_3). \quad (2.154)$$

Similarly, as U_1, U_2, U_3 can decode X, Y in decoder D_4, we can apply

$$H(X|U_1,U_2,U_3) = 0 \quad (2.155)$$

$$H(X,Y|U_1,U_2,U_3) = 0 \quad (2.156)$$

respectively in step 5 and step 6 to get

$$2R_1 + 2R_2 + 2R_3 \geq H(X) + 2H(X,Y). \quad (2.157)$$

Finally, by applying the source independence equality $H(X,Y) = H(X) + H(Y)$, we get

$$2R_1 + 2R_2 + 2R_3 \geq 3H(X) + 2H(Y) \quad (2.158)$$

to complete the proof.

2.8 Conclusion

This chapter investigated the rate regions of 7360 non-isomorphic MDCS instances, which represent 135043 isomorphic instances. Although the exact rate region of a general network coding or distributed storage instance is, in general, only expressible in terms of the (non-polyhedral) region of entropic vectors, for the class of MDCS problems considered in this chapter we are able to bypass this difficulty through the use of both i) the polyhedral Shannon outer bound, and ii) several polyhedral inner bounds (associated with binary matroid, ternary matroid, representable matroid, and superposition regions), by showing that the associated inner and outer bounds on the rate region are equal. Part of our contribution is the explicit construction of superposition, scalar, and vector codes over various fields using the polyhedral inner bounds. Inspired by the notion of a forbidden matroid minor, we introduced several MDCS contraction, deletion, and unification operations and demonstrated that these operations can be used to identify a set of forbidden MDCS instances for the achievability of various classes of codes, in the sense that any extension of such a forbidden MDCS instance will likewise have a rate region for which that class of codes is insufficient. Finally, we demonstrated that we can automate the creation of "human readable proofs" for the rate regions for these 7360 non-isomorphic MDCS instances by posing a suitable optimization problem asking for the smallest cardinality set of Shannon-type inequalities required to step-by-step construct the rate region "by hand".

Several remaining intriguing questions remain as future work. One important question is whether or not the Shannon outer bound is tight for the class of MDCS instances. For all the instances considered in this chapter this has been the case, although we note there are 492 instances for which we have not yet established the rate region. A second important question is whether or not there exists a finite list of forbidden minors for the classes of codes considered in this chapter. That is, our results have established some forbidden minor MDCS instances, but we have not established for any class of codes if we have all such forbidden minors for that class.

Chapter 3: General Multi-source Multi-sink Hyperedge Networks

3.1 Introduction

Many important practical problems, including efficient information transfer over networks [20, 36], the design of efficient distributed information storage systems [4,5], and the design of streaming media systems [6–8], have been shown to involve determining the capacity region of an abstracted network under network coding. Yan *et al.*'s celebrated paper [15], has provided an exact representation of these capacity regions of networks under network coding. Their essential result is that the capacity region of the network can be expressed by intersecting the region of entropic vectors [2, 40] with a series of linear equality constraints created by the network's topology and the problem demands, then projecting the result onto the entropies of sources and edge variables. However, this is only an implicit description of the rate region, because the involved region of entropic vectors $\bar{\Gamma}_N^*$ is still unknown for $N \geq 4$.

Nevertheless, we have previously demonstrated in [22, 23, 41] through the use of appropriate inner and outer bounds to $\bar{\Gamma}_N^*$ we will review in §3.2.3, this implicit formulation can be used to develop algorithms by which a computer can very rapidly calculate the capacity region, its proof, and the class of capacity achieving codes, for small networks, each of which would previously have taken a trained information theorist hours or longer to derive. While the development of this rate region calculation, code selection, and converse proof generation algorithm [41] is not the focus of the present chapter, it involves developing techniques to derive polyhedral inner and outer bound descriptions for constrained regions of entropic vectors, and polyhedral projection. When it comes to rate regions, the definitions in Sections §3.5 and §3.2.3 the chapter will focus more on exactly what was calculated, what can be calculated, and, later in the chapter, what can be learned from the resulting rate regions, rather than the exact computations by which the rate regions were reached, with the rate region algorithm design and specialization, which involve a separate and parallel line of investigation, left to discussion by another series of papers [33, 42, 43].

The ability to calculate network coding rate regions for small networks rapidly with a computer motivates an alternative, more computationally thinking oriented, agenda to the study of network

coding capacity regions. At the beginning, one's goal is to demonstrate the method's power by applying the algorithm to derive the capacity region of as many networks and applications of network coding as possible. To do this, in §3.2 we first slightly generalize Yeung's labeled directed acyclic graph (DAG) model for a network coding problem to a directed acyclic hypergraph context, then demonstrate how the enlarged model handles as special cases the wide variety of other models in applications in which network coding is being employed, including, but not limited to, index coding, multilevel diversity coding systems (MDCS), distributed storage, and so on.

With the slightly more general model in hand, the first issue in the computationally thinking oriented agenda is *network generation and enumeration*, i.e., how to list all of the networks falling in this model class and, if interested, its special subclasses. In order to avoid repetition work, thereby reaching the largest number of networks possible with a constant amount of computation, it is desirable to understand precisely when two instances of this model (i.e., two network coding problems) are equivalent to one another, in the sense that the solution to one directly provides a solution to another. This notion of network coding problem equivalence, which provides a very different approach but is in the same high level spirit as the transformation of network channel coding problems to network coding problems in [18] and the transformation of network coding problems to index coding problems in [19], will be revisited at multiple points of the chapter, beginning in this present context of enumeration, but also playing an important role in the discussion of hierarchy later.

The first notion of equivalence we develop is that of *minimality*, by removing any redundant or unnecessary parts in the network coding problem description. Partially owing to the generality of the network coding problem model, many valid instances of it include within a network parts which can be immediately detected as extraneous to the network coding problem. In this sense, the problem is directly reducible to another smaller problem by removing completely unnecessary and unhelpful sources, nodes, or edges. In order to provide the smallest possible problem description by not including these extraneous components, we formalize in §3.3 the notion of *network minimality*, listing a series of conditions which a network coding problem description must obey to not contain any obviously extraneous sources, nodes, or edges.

The next notion of equivalence looks to symmetry or isomorphism between problem descriptions. Beginning by observing that a network must be labeled to specify its graph, source availability, and demands to the computer, yet the underlying network coding problem is insensitive to the selection of these labels, we define in §3.4.2 a notion of network coding problem equivalence through the isomorphism associated with the selection of these labels. We review that the proper way to

formalize this notion is through identifying equivalence classes with orbits under group actions. A naïve algorithm to provide the list of network coding problems to the rate region computation software would simply list all possible labeled network coding problems, test for isomorphism, then narrow the list down to only those which are inequivalent to one another, keeping only one element, the canonical representative, of the equivalence class. However, the key reason for formalizing this notion of equivalence is that the number of labeled network coding problem instances explodes far faster than the number of network coding problem equivalence classes. Hence, we develop a better technique for generating lists of canonical network coding problem instances by harnessing techniques and algorithms from computational group theory that enable us to directly list the minimal canonical representatives of the network coding problem equivalence classes as described in §3.4.3.

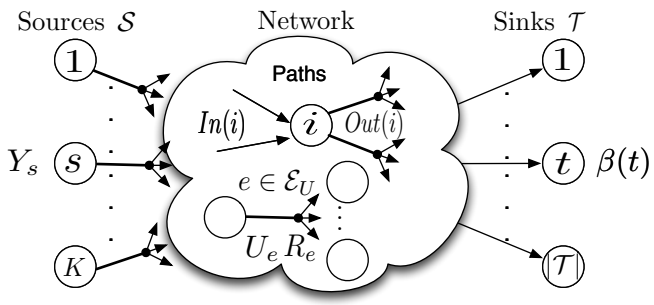
With the list of all minimal canonical network coding problems up to a certain size in hand, we can utilize our algorithm and software to calculate the rate region bounds, the Pareto optimal network codes, and the converse proofs, for each, building a very large database of capacity regions of network coding problems up to this size. Owing to the variety of the model, even for tiny problems, this database quickly grows very large relative to what a human would want to read through. For instance, our previous chapter applying this computational agenda to the narrower class of MDCS problems [41], yielded the capacity regions of 6,868 equivalence classes of MDCS problems and bounds for 492 more MDCS problem equivalence classes, while the database developed in this chapter contains the capacity regions of 635,481 equivalence classes of network coding problems and bounds for 90,075 more equivalence classes of network coding problems. The equivalence classes of networks correspond to solutions for 7,557,810 network coding problems with graphs specified via edge dependences and 545,885,972,444 network coding problems specified in the typical node representation of a graph, and bounds for 1,061,254 network coding problems specified via edge dependencies and 1,835,126,442,560 network coding problems represented in the standard node representation of the graph. While it is possible to use the database to report statistics regarding the sufficiency of certain classes of codes and/or the necessity of non-Shannon inequalities as will be done in §3.5, in order to more meaningfully enable humans to learn from the database, as well as from the computational research, one must utilize some notion of network structure to organize it for analysis.

Our method of endowing the structure on the set of network coding problems is through *hierarchy*, in which we explain the properties and/or rate regions of larger networks as being inherited from smaller networks (or vice-versa). Of course, part of a network coding problem is the network graph, and further, network coding and entropy is related to matroids, and these nearby fields of graph

theory and matroid theory have both undergone a thorough study of hierarchy which directly inspires our approach to it. In graph theory, this notion of hierarchy is achieved by recognizing within large graphs smaller graphs which can be created by deleting or contracting the larger graph's edges, called *minors* and is directly associated with a crowning achievement. Namely, the celebrated well-quasi-ordering result of graph theory [13, 14], showed that any minor closed family of graphs (i.e., ones for which any minor of a graph in that family is also in the family) has at most a finite list of forbidden minors, which no graphs in that family can contain. While the families of minor closed graphs are typically infinite, they are then, in a sense, capable of being studied through a finite object which is their forbidden minors – if a graph does not has one of these forbidden minors, it is then in the family. In matroid theory, which in a certain sense extends graph theory, one has a similar notion of hierarchy endowed through matroid minors, generated through matroid contraction and deletion. While one can generate minor closed families of matroids with an infinite series of forbidden minors, the celebrated, and possibly recently proved, Rota's conjecture [11,44], stated that those matroids capable of being represented over a particular finite field have at most a finite list of forbidden minors. In this chapter, inspired by these hierarchy theories in graphs and matroids, we aim to derive a notion of network minors created from a series of contraction and deletion-like operators that shrink network coding problems, called *embedding operators*, as well as operations for building larger network coding problems from smaller ones, called *combination operators*. These operators work together to build our notion of minors and a sense of hierarchy among network coding problems.

Developing a notion of network coding problem hierarchy is important for several reasons. First of all, as explained above, even after one has calculated the capacity regions of all networks up to a certain size, it is of interest to make sense of this very large quantity of information by studying its structure, and hierarchy is one way of creating a notion of structure. Second of all, the computational techniques for proving network rate regions can only handle networks with tens of "variables", the sum of the number of sources and number of hyperedges in the graph, and hence are limited to direct computation of fairly small problem instances. If one wants to be able to utilize the information gathered about these small networks to understand the capacity regions of networks at scale, one needs methods for putting the smaller networks together into larger networks in a way such that the capacity region of the larger network can be directly calculated from those of the smaller networks.

Our embedding operators, defined and discussed in §3.6, extend the series of embedding operations we had for MDCS problems in [41], and augment them, to provide methods for obtaining small networks from big networks in such a way that the rate region of the smaller network, and its



$$\mathbb{A}[R_1, R_2, \ldots]^T \geq \mathbb{B}[H(Y_1), H(Y_2), \ldots]^T$$

Figure 3.1: A general network model A.

properties, are directly inherited from the larger network. Our combinations operators, discussed in §3.7, work in the opposite direction: they provide methods for putting together smaller networks to make larger networks in such a way that the rate region, and its properties, of the larger network can be directly calculated from the rate region of the smaller networks. Both of our lists of operators are small and somewhat simple, however, when they work together, they provide a very powerful way of endowing hierarchical structure in network coding problems. In particular, the joint use of the combination and embedding operators provide a very powerful way for obtaining rate regions of large networks from small ones, as well as describing the properties of families of network coding problems, as we demonstrate in §3.8. They open the door to many new avenues of network coding research, and we shall describe briefly some of the related future problems for investigation in §3.9.

3.2 Background: Network Coding Problem Model, Capacity Region, and Bounds

The class of problems under study in this chapter are the capacity regions of multi-source multi-sink network coding problems with hyperedges (hyperedge MSNC problems), which we define presently. A network coding problem in this class, denoted by the symbol A, includes a directed acyclic hypergraph $(\mathcal{V}, \mathcal{E})$ [45] as in Fig. 3.1, consisting of a set of nodes $\mathcal{V}$ and a set $\mathcal{E}$ of directed hyperedges in the form of ordered pairs $e = (v, \mathcal{A})$ with $v \in \mathcal{V}$ and $\mathcal{A} \subseteq \mathcal{V} \setminus \{v\}$. The nodes $\mathcal{V}$ in the graph are partitioned into the set of source nodes $\mathcal{S}$, intermediate nodes $\mathcal{G}$, and sink nodes $\mathcal{T}$, i.e., $\mathcal{V} = \mathcal{S} \cup \mathcal{G} \cup \mathcal{T}$. Each of the source nodes $s \in \mathcal{S}$ will have a single outgoing edge $(s, \mathcal{A}) \in \mathcal{E}$. The source nodes in $\mathcal{S}$ have no in edges, the sink nodes $\mathcal{T}$ have no out edges, and the intermediate nodes $\mathcal{G}$ have both in and out edges. The number of sources will be denoted by $|\mathcal{S}| = K$, and each source node $s \in \mathcal{S}$ will be associated with an independent random variable Y_s, $s \in \mathcal{S}$, with entropy $H(Y_s)$, and an associated independent and identically distributed (IID) temporal sequence of random values. For every source

$s \in \mathcal{S}$, define $\mathrm{Out}(s)$ to be its single outgoing edge, which is connected to a subset of intermediate nodes and sink nodes. A hyperedge $e \in \mathcal{E}$ connects a source to a subset of intermediate nodes, or connects an intermediate node to a subset of intermediate and sink nodes, i.e., $e = (i, \mathcal{F})$,where $i \in \mathcal{S}, \mathcal{F} \subseteq \mathcal{G}$, or $i \in \mathcal{G}, \mathcal{F} \subset \mathcal{G} \cup \mathcal{T}$ and $i \notin \mathcal{F}$. For brevity, we will refer to hyperedges as edges if there is no confusion. For an intermediate node $i \in \mathcal{G}$, we denote its incoming edges as $\mathrm{In}(i)$ and outgoing edges as $\mathrm{Out}(i)$. For each edge $e \in \mathcal{E}$, the associated random variable $U_e = f_e(\mathrm{In}(i))$ is a function of all the inputs of node i, obeying the edge capacity constraint $R_e \geq H(U_e)$. The tail (head) node of edge e is denoted as $\mathrm{Tl}(e)$ ($\mathrm{Hd}(e)$). For notational simplicity, the unique edge of out each source node will be the sources' random variable, $U_e = Y_s$ if $\mathrm{Tl}(e) = s$, denoting $\mathcal{E}_S = \{e \in \mathcal{E} | \mathrm{Tl}(e) = s, s \in \mathcal{S}\}$ to be the source random variables, and $\mathcal{E}_U = \mathcal{E} \setminus \mathcal{E}_S$ to be the edge random variables. In the following text, an edge variable is referred to an element in $\mathcal{E}_U$, unless stated otherwise. For each sink $t \in \mathcal{T}$, the collection of sources this sink will demand will be labeled by $\beta(t) \subseteq \mathcal{S}$. Thus, a network A can be represented as a tuple $\mathsf{A} = (\mathcal{S}, \mathcal{G}, \mathcal{T}, \mathcal{E}, (\beta(t), t \in \mathcal{T}))$. For convenience, networks with K sources and $|\mathcal{E}_U|$ edges are referred as $(K, |\mathcal{E}_U|)$ instances.

As our focus in the manuscript will be on rate regions of a very similar form to those in [2, 15], this network coding problem model is as close to the original one in [2, 15] as possible while covering the multiple instances of applications in network coding in which the same message can be overheard by multiple parties. These applications include index coding, wireless network coding, independent distributed source coding, and distributed storage, and the simplest and most direct model change to incorporate this capability is to switch from directed acyclic graphs to directed acyclic hypergraphs. As we shall see in §3.2.2, this small change to the model is easily reconciled with the network coding rate region expression and its proof from [2, 15].

3.2.1 Special Network Classes

The network coding problem model just described has been selected to be general enough to include a variety of models from the applications of network coding as special cases. A few of these special cases that will be of interest in examples later in the chapter are reviewed here, including a description of the extra restrictions on the model to fall into this subclass of problems.

Example 5 (Yan Yeung Zhang MSNC)**:** The network model in [2], where the edges are not hyperedges and output of a source node can be different functions of the source, can be viewed as a special class of networks in our model. This is because the sources can be viewed as intermediate nodes and a virtual source node connecting to each of them can be added. For instance, a small network instance of the

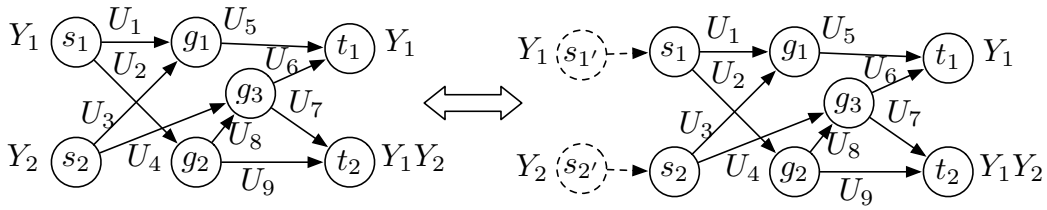


Figure 3.2: A normal MSNC in [2] can be viewed as a special instance of the hyperedge MSNC.

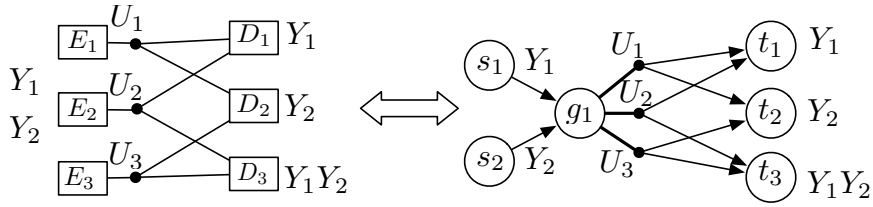


Figure 3.3: An IDSC problem can be viewed as a hyperedge MSNC.

model in [2], as shown in Fig. 3.2, can be viewed as a hyperedge network instance introduced in this chapter, by adding virtual source nodes $s_{1'}, s_{2'}$ to sources s_1, s_2, respectively.

Example 6 (Independent Distributed Source Coding)**:** The independent distributed source coding (IDSC) problems, which was motivated from satellite communication systems [3], can be viewed as a special class of networks in our model. They are three-layer networks, where sources are connected with some intermediate nodes and those intermediate nodes will transmit coded messages to sinks. For instance, the IDSC problem in Fig. 3.3 can be converted to a hyperedge multi-source network coding problem. As a special class of IDSC problems with decoding priorities among sources, the multi-level diversity coding systems [1, 41] are naturally a class of networks in our current general model. In the experimental results section, we will not only show results on general hyperedge networks, but also some results on IDSC problems.

Example 7 (Index Coding)**:** Since direct access to sources as side information is allowed in our network model, the index coding problems are also a special class of our model with only one intermediate edge. That is, a K-source index coding problem can be viewed as a $(K, 1)$ hyperedge MSNC and vice versa. For instance, an index coding problem with 3 sources, as shown in Fig. 3.4, is a $(3, 1)$ instance in our model.

3.2.2 Rate Region

Having clarified the definition of a network coding problem, we must now identify the precise definition of a network code and the network coding capacity region we will utilize.

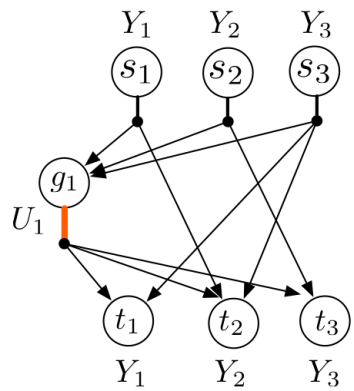


Figure 3.4: An index coding problem is a three-layer hyperedge MSNC with only one intermediate edge.

Each source node $s \in \mathcal{S}$ is associated with an IID sequence $Y_s^1, Y_s^2, \ldots$ distributed according to its independent random variable Y_s taking values in $\mathcal{Y}_s$. Let $\mathbf{R} = (H(Y_1), \ldots, H(Y_K), R_1, \ldots, R_{|\mathcal{E}_U|}) \in \mathbb{R}_+^{|\mathcal{E}|}$ be a vector of source and edge rates. A $(n, \mathbf{R})$ block code over $\mathbb{F}_q$ consists of a series of block encoders, one for each edge $e \in \mathcal{E}_U$, which are functions that map a block of n source observations from any sources in $\mathcal{E}_\mathcal{S} \cap \mathrm{In}(\mathrm{Tl}(e))$, and the incoming messages associated with the edges $\mathcal{E}_U \cap \mathrm{In}(\mathrm{Tl}(e))$ including sources and some other edges, to one of $\lceil q^{nR_e} \rceil$ different descriptions in $\eta_e = \{0, 1, \ldots, \lceil q^{nR_e} \rceil - 1\}$

$$f_e^{(n)} : \prod_{s \in \mathcal{E}_\mathcal{S} \cap \mathrm{In}(\mathrm{Tl}(e))} \mathcal{Y}_k^n \times \prod_{i \in \mathcal{E}_U \cap \mathrm{In}(\mathrm{Tl}(e))} \eta_e \to \eta_e, \ e \in \mathcal{E}_U. \tag{3.1}$$

and a series of decoders associated with the sinks $t \in \mathcal{T}$, which are functions

$$g_t^{(n)} : \prod_{e \in \mathrm{In}(t) \cap \mathcal{E}_U} \eta_e \times \prod_{e \in \mathrm{In}(t) \cap \mathcal{E}_S} \mathcal{Y}_e^n \to \prod_{k \in \beta(t)} \mathcal{Y}_k^n, \ t \in \mathcal{T}. \tag{3.2}$$

Denote by $U_e^n \in \eta_e$ the message on edge e, $e \in \mathcal{E}_U$, which is the result of the encoding function f_e^n.

The achievable rate region of a network A, denoted as $\mathcal{R}_*(\mathsf{A})$, consists of all rate vectors $\mathbf{R} = (H(Y_1), \ldots, H(Y_K), R_1, \ldots, R_{|\mathcal{E}_U|})$ such that there exist sequences of encoding functions $f^n = (f_e^n, e \in \mathcal{E})$ and decoding functions $g^n = (g_t^n, t \in \mathcal{T})$ for which the probability of error at each sink can be made arbitrarily small as $n \to \infty$, and the closure of this achievable region is the capacity region. Specifically, define the probability of error for each sink $t \in \mathcal{T}$ as

$$p_t^{n,\mathrm{err}}(\mathbf{R}) = \mathbb{P}(g_t^n(U_{\mathrm{In}(t)}^n) \neq \beta(t)^{1:n}), \tag{3.3}$$

and the maximum over these as

$$p^{n,\mathrm{err}}(\mathbf{R}) = \max_{t\in\mathcal{T}} p_t^{n,err}. \tag{3.4}$$

A rate vector is in the rate region, $\mathbf{R} \in \mathcal{R}_*(\mathsf{A})$, is achievable provided there exists a sequence of encoders $\{f_e^n\}$ and decoders $\{g_t^n\}$ such that $p^{n,\mathrm{err}}(\mathbf{R}) \to 0$ as $n \to \infty$, and the closure of the set of achievable rate vectors is the capacity region.

The rate region $\mathcal{R}_*(\mathsf{A})$ can be expressed in terms of the region of entropic vectors, Γ_N^*, as in [2, 15]. For the hyperedge MSNC problem, denote a series of edge random variables U_e, $e \in \mathcal{E}_U$, and collect them into the set $\mathcal{N} = \{Y_k, k \in \{1, \ldots, K\}\} \cup \{U_e, e \in \mathcal{E}_U\}$ and define $N = |\mathcal{N}|$, where $Y_k, k \in \{1, \ldots, K\}$ is the auxiliary random variable associated with source k and its single outgoing hyperedge, and U_e is a random variable associated with edge $e \in \mathcal{E}_U$. The rate region $\mathcal{R}_*(\mathsf{A})$ is the set of rate vectors $\mathbf{R}$ such that there exists $\mathbf{h} \in \Gamma_N^*$ satisfying the following

$$h_{\mathbf{Y}_\mathcal{S}} = \sum_{s\in\mathcal{S}} h_{Y_s} \tag{3.5}$$

$$h_{\mathbf{U}_{\mathrm{Out}(g)}|(\mathbf{Y}_{\mathcal{S}\cap\mathrm{In}(g)}\cup\mathbf{U}_{\mathcal{E}_U\cap\mathrm{In}(g)})} = 0,\ g \in \mathcal{G} \tag{3.6}$$

$$h_{Y_s} \geq H(Y_s),\ s \in \mathcal{S} \tag{3.7}$$

$$h_{\beta(t)|\mathbf{U}_{\mathrm{In}(t)}} = 0,\ t \in \mathcal{T} \tag{3.8}$$

$$R_e \geq h_{U_e},\ e \in \mathcal{E}_U \tag{3.9}$$

where h_A is the coordinate in the rate vector $\mathbf{h}$ associated with A and the conditional entropies $h_{A|B}$ are naturally equivalent to $h_{AB} - h_B$. These constraints can be interpreted as follows: (3.5) represents that sources are independent; (3.6) represents that outgoing messages of an intermediate node are functions of all its incoming sources and edges; (3.7) represents that the associated entropies in the entropic vector must obey the actual source rate constraints; (3.8) represents that recovered source messages at a sink are a function of the input edges available to it; and (3.9) represents the edge capacity constraints.

Since the network constraints involve edge capacities $R_e, e \in \mathcal{E}_U$, we will consider the space in $\mathbb{R}^M$, where $M = 2^N - 1 + |\mathcal{E}_U|$, $N = |\mathcal{E}| = |\mathcal{E}_S| + |\mathcal{E}_U|$, associated with (combinations of) achievable source entropies and edge capacities. We define $\mathcal{L}_i, i = 1, 3, 4', 5$ as network constraints representing source independence, coding by intermediate nodes, edge capacity constraints, and sink

nodes decoding, constraints respectively:

$$\mathcal{L}_1 = \{\mathbf{h} \in \mathbb{R}^M : h_{\mathbf{Y}_\mathcal{S}} = \Sigma_{s\in\mathcal{S}} h_{Y_s}\} \tag{3.10}$$

$$\mathcal{L}_3 = \{\mathbf{h} \in \mathbb{R}^M : h_{\mathbf{U}_{\mathrm{Out}(g)}|(\mathbf{Y}_{\mathcal{S}\cap\mathrm{In}(g)}\cup\mathbf{U}_{\mathcal{E}_U\cap\mathrm{In}(g)})} = 0, g \in \mathcal{G}\} \tag{3.11}$$

$$\mathcal{L}_{4'} = \{(\mathbf{h}^T, \boldsymbol{r}^T)^T \in \mathbb{R}_+^M : R_e \geq h_{U_e}, e \in \mathcal{E}\} \tag{3.12}$$

$$\mathcal{L}_5 = \{\mathbf{h} \in \mathbb{R}^M : h_{\beta(t)|\mathbf{U}_{\mathrm{In}(t)}} = 0, \forall t \in \mathcal{T}\}. \tag{3.13}$$

and we will denote $\mathcal{L}(\mathsf{A}) = \mathcal{L}_1 \cap \mathcal{L}_3 \cap \mathcal{L}_{4'} \cap \mathcal{L}_5$. Note that, we do not have $\mathcal{L}_2$ constraints (which represent the coding function at each source) as in [15], due to our different notation with $U_e = Y_s$ if $\mathrm{Tl}(e) = s$. Further, Γ_N^* and $\mathcal{L}_i, i = 1, 3, 4', 5$ are viewed as subsets of $\mathbb{R}^M$, $M = 2^N - 1 + |\mathcal{E}_U|$, $N = |\mathcal{E}| = |\mathcal{E}_S| + |\mathcal{E}_U|$, with coordinates $[\boldsymbol{h}^T, \boldsymbol{r}^T]^T$, with $\boldsymbol{h} \in \mathbb{R}^{2^N-1}$ indexed by subsets of $\mathcal{N}$ as is usual in entropic vectors, $\boldsymbol{r} \in \mathbb{R}^{|\mathcal{E}_U|}$ playing the role of the capacities of edges, and any unreferenced dimensions (e.g. $\boldsymbol{r}$ in Γ_N^*) are left unconstrained (e.g., $\boldsymbol{r} \in \mathbb{R}^{|\mathcal{E}_U|}$ in Γ_N^*). The following extension of the capacity region from [15] characterizes our slightly different capacity region formulation $\mathcal{R}_*(\mathsf{A})$ for our slightly different problem.

Theorem 10: The expression of the rate region of a network A is

$$\mathcal{R}_*(\mathsf{A}) = \mathrm{Proj}_{\boldsymbol{r},\boldsymbol{\omega}}(\overline{\mathrm{con}(\Gamma_\mathrm{N}^* \cap \mathcal{L}_{13})} \cap \mathcal{L}_5 \cap \mathcal{L}_4'), \tag{3.14}$$

where $\mathrm{con}(\mathcal{B})$ is the conic hull of $\mathcal{B}$, and $\mathrm{Proj}_{\boldsymbol{r},\boldsymbol{\omega}}(\mathcal{B})$ is the projection of the set $\mathcal{B}$ on the coordinates $\left[\boldsymbol{r}^T, \boldsymbol{\omega}^T\right]^T$ where $\boldsymbol{r} = [R_e | e \in \mathcal{E}_U]$ and $\boldsymbol{\omega} = [H(Y_s), s \in \mathcal{S}]$.

Proof: We present a sketch of the proof here and a detailed proof in Appendix A. First observe that the proof of Theorem 1 in [15] can be extended to networks presented above, with hyperedges and intermediate nodes having direct access to sources. Some differences include: I) the extension potentially constrains more on a random variable in the region of entropic vectors, since it may be involved in more than one network constraints; II) the coding function for each intermediate (hyper)edge may encode the sources with some other edge variables together; III) the decoding at sink nodes may be a function of some sources and intermediate edges as well; IV) there is only one outgoing edge for each source and it carries the source variable itself. The differences will not destroy the essence of the proofs in [15]. For the converse and achievability proof, we view the edge capacities as constant (recall that our rate vector include both source entropies and edge capacities), and then consider the converse and achievability of the associated source entropies, which becomes

essentially the proof in [15]. ■

While the analytical expression determines, in principle, the rate region of any network under network coding, it is only an implicit characterization. This is because Γ_N^* is unknown and even not polyhedral for $N \geq 4$. Thus, the direct calculation of rate regions from equation (3.14) for a network with more than 4 variables is infeasible. However, replacing Γ_N^* with polyhedral inner and outer bounds allows (3.14) to become a polyhedral computation problem which involves applying some constraints onto a polyhedron and then projecting down onto some coordinates. This inspires us to substitute Γ_N^* with its closed polyhedral outer and inner bounds $\Gamma_N^{\text{out}}, \Gamma_N^{\text{in}}$ respectively, and get an outer and inner bound on the rate region:

$$\mathcal{R}_{\text{out}}(\mathsf{A}) = \text{proj}_{\boldsymbol{r},\boldsymbol{\omega}}(\Gamma_N^{\text{out}} \cap \mathcal{L}_{\mathsf{A}}), \tag{3.15}$$

$$\mathcal{R}_{\text{in}}(\mathsf{A}) = \text{proj}_{\boldsymbol{r},\boldsymbol{\omega}}(\Gamma_N^{\text{in}} \cap \mathcal{L}_{\mathsf{A}}). \tag{3.16}$$

If $\mathcal{R}_{\text{out}}(\mathsf{A}) = \mathcal{R}_{\text{in}}(\mathsf{A})$, we know $\mathcal{R}(\mathsf{A}) = \mathcal{R}_{\text{out}}(\mathsf{A}) = \mathcal{R}_{\text{in}}(\mathsf{A})$. Otherwise, tighter bounds are necessary.

In this work, we will follow (3.15) and (3.16) to calculate the rate region. Typically the Shannon outer bound Γ_N and some inner bounds obtained from matroids, especially representable matroids, are used. We will briefly review the definition of these bounds in the next subsection, while details on the polyhedral computation methods with these bounds are available in [22, 23, 33, 42].

3.2.3 Construction of bounds on rate region

An introduction to the region of entropic vectors and the polyhedral inner and outer bounds we will utilize from it can be found in greater detail in [41]. Here we briefly review their definitions for accuracy, completeness, and convenience.

3.2.3.1 Region of entropic vectors Γ_N^*

Consider an arbitrary collection $\mathbf{X} = (X_1, \ldots, X_N)$ of N discrete random variables with joint probability mass function p_X. To each of the $2^N - 1$ non-empty subsets of the collection of random variables, $X_{\mathcal{A}} := (X_i \,|i \in \mathcal{A})$ with $\mathcal{A} \subseteq \{1, \ldots, N\} \equiv [[N]]$, there is associated a joint Shannon entropy $H(X_{\mathcal{A}})$. Stacking these subset entropies for different subsets into a $2^N - 1$ dimensional vector we form an entropy vector

$$\mathbf{h} = [H(X_{\mathcal{A}}) \,|\mathcal{A} \subseteq [[N]], \mathcal{A} \neq \emptyset] \tag{3.17}$$

By virtue of having been created in this manner, the vector $\mathbf{h}$ must live in some subset of $\mathbb{R}_+^{2^N-1}$, and is said to be *entropic* due to the existence of p_X. However, not every point in $\mathbb{R}_+^{2^N-1}$ is entropic since, for many points, there does not exist an associated valid distribution p_X. All entropic vectors form a region denoted as Γ_N^*. It is known that the closure of the region of entropic vectors $\bar{\Gamma}_N^*$ is a convex cone [15]. Elementary inequalities on Shannon entropies should form a fundamental outer bound on Γ_N^*, named the *Shannon outer bound* Γ_N.

3.2.3.2 Shannon outer bound Γ_N

We observe that elementary properties of Shannon entropies indicate that $H(X_{\mathcal{A}})$ is a non-decreasing submodular function, so that $\forall \mathcal{A} \subseteq \mathcal{B} \subseteq [[N]], \forall \mathcal{C}, \mathcal{D} \subseteq [[N]]$

$$H(X_{\mathcal{A}}) \leq H(X_{\mathcal{B}}) \tag{3.18}$$

$$H(X_{\mathcal{C}\cup\mathcal{D}}) + H(X_{\mathcal{C}\cap\mathcal{D}}) \leq H(X_{\mathcal{C}}) + H(X_{\mathcal{D}}). \tag{3.19}$$

Since they are true for any collection of subset entropies, these linear inequalities (3.18), (3.19) can be viewed as supporting halfspaces for Γ_N^*.

Thus, the intersection of all such inequalities form a polyhedral outer bound Γ_N for Γ_N^* and $\bar{\Gamma}_N^*$, where

$$\Gamma_N := \left\{ \mathbf{h} \in \mathbb{R}^{2^N-1} \;\middle|\; \begin{array}{c} h_{\mathcal{A}} \leq h_{\mathcal{B}} \quad \forall \mathcal{A} \subseteq \mathcal{B} \\ h_{\mathcal{C}\cup\mathcal{D}} + h_{\mathcal{C}\cap\mathcal{D}} \leq h_{\mathcal{C}} + h_{\mathcal{D}} \quad \forall \mathcal{C}, \mathcal{D} \end{array} \right\}.$$

This outer bound Γ_N is known as the *Shannon outer bound*, as it can be thought of as the set of all inequalities resulting from the positivity of Shannon's information measures among the random variables. While $\Gamma_2 = \Gamma_2^*$ and $\Gamma_3 = \bar{\Gamma}_3^*$, $\bar{\Gamma}_N^* \subsetneq \Gamma_N$ for all $N \geq 4$ [15], and indeed it is known [35] that $\bar{\Gamma}_N^*$ is not even polyhedral for $N \geq 4$.

The inner bounds on Γ_N^* we consider are based on representable matroids. We briefly review the basic definitions of matroids and representable matroids.

3.2.3.3 Matroid basics

Matroid theory [11] is an abstract generalization of independence in the context of linear algebra and graphs to the more general setting of set systems. There are numerous equivalent definitions of matroids, however, we will present the definition of matroids utilizing *rank functions* as this is best matched to our purposes.

Definition 11: A set function $r_M : 2^{\mathcal{M}} \to \{0, \ldots, N\}$ is a rank function of a matroid if it obeys the following axioms:

1. Cardinality: $r_M(\mathcal{A}) \leq |\mathcal{A}|$;
2. Monotonicity: if $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{M}$ then $r_M(\mathcal{A}) \leq r_M(\mathcal{B})$;
3. Submodularity: if $\mathcal{A}, \mathcal{B} \subseteq \mathcal{M}$ then $r_M(\mathcal{A} \cup \mathcal{B}) + r_M(\mathcal{A} \cap \mathcal{B}) \leq r_M(\mathcal{A}) + r_M(\mathcal{B})$.

A subset with rank function $r_M(\mathcal{A}) = |\mathcal{A}|$ is called an independent set of the matroid. Though there are many classes of matroids, we are especially interested in one of them, *representable matroids*, because they can be related to linear codes to solve network coding problems as discussed in [22,23].

3.2.3.4 Representable matroids

Representable matroids are an important class of matroids which connect the independent sets to the notion of independence in a vector space.

Definition 12: A matroid M with ground set $\mathcal{M}$ of size $|\mathcal{M}| = N$ and rank $r_M(\mathcal{M}) = r$ is representable over a field $\mathbb{F}$ if there exists a matrix $\mathbb{A} \in \mathbb{F}^{r \times N}$ such that for each set $\mathcal{A} \in \mathcal{M}$ the rank $r_M(\mathcal{A})$ equals the linear rank of the corresponding columns in $\mathbb{A}$, viewed as vectors in $\mathbb{F}^r$.

Note that, for an independent set $\mathcal{I}$, the corresponding columns in the matrix $\mathbb{A}$ are linearly independent. There has been significant effort towards characterizing the set of matroids that are representable over various field sizes, with a complete answer only available for fields of sizes two, three, and four. For example, a matroid M is binary representable (representable over a binary field) iff it does not have the matroid $U_{2,4}$ as a minor. Here, a minor is obtained by series of operations of contraction and deletion [11]. $U_{k,N}$ is the *uniform* matroid on the ground set $\mathcal{M} = \{1, \ldots, N\}$ with independent sets $\mathcal{I}$ equal to all subsets of $[[N]]$ of size at most k. For example, $U_{2,4}$ has as its independent sets

$$\mathcal{I} = \{\emptyset, 1, 2, 3, 4, \{1, 2\}, \{1, 3\}, \{1, 4\}, \{2, 3\}, \{2, 4\}, \{3, 4\}\}. \tag{3.20}$$

Another important observation is that the first non-representable matroid is the so-called *Vámos* matroid, a well known matroid on ground set of size 8. That is to say, all matroids are representable, at least in some field, for $N \leq 7$.

3.2.3.5 Inner bounds from representable matroids

Suppose a matroid M with ground set $\mathcal{M}$ of size $|\mathcal{M}| = N$ and rank $r_M(\mathcal{M}) = k$ is representable over the finite field $\mathbb{F}_q$ of size q and the representing matrix is $\mathbb{A} \in \mathbb{F}_q^{k\times N}$ such that $\forall \mathcal{B} \subseteq \mathcal{M}$ $r_M(\mathcal{B}) = \text{rank}(\mathbb{A}_{:,\mathcal{B}})$, the matrix rank of the columns of $\mathbb{A}$ indexed by $\mathcal{B}$. Let Γ_N^q be the conic hull of all rank functions of matroid with N elements and representable in $\mathbb{F}_q$. This provides an inner bound $\Gamma_N^q \subseteq \bar{\Gamma}_N^*$, because any extremal rank function $\mathbf{r}$ of Γ_N^q is by definition representable and hence is associated with a matrix representation $\mathbb{A} \in \mathbb{F}_q^{k\times N}$, from which we can create the random variables

$$(X_1, \ldots, X_N) = \mathbf{u}\mathbb{A}, \quad \mathbf{u} \sim \mathcal{U}(\mathbb{F}_q^k), \tag{3.21}$$

whose elements are $h_{\mathcal{A}} = r_M(\mathcal{A}) \log_2 \ q, \quad \forall \mathcal{A} \subseteq \mathcal{M}$. Hence, all extreme rays of Γ_N^q are entropic, and $\Gamma_N^q \subseteq \bar{\Gamma}_N^*$. Further, if a vector in the rate region of a network is (a projection of) an $\mathbb{F}_q$-representable matroid rank, the representation $\mathbb{A}$ can be used as a linear code to achieve that rate vector, and this code is denoted a *basic scalar* $\mathbb{F}_q$ *code*. For an interior point in the rate region, which is the conic hull of projections of $\mathbb{F}_q$-representable matroid ranks, the code to achieve it can be constructed by time-sharing between the basic scalar codes associated with the ranks involved in the conic combination. This code is denoted a *scalar* $\mathbb{F}_q$ *code*. Details on construction of such a code can be found in [22] and [41].

One can further generalize the relationship between representable matroids and entropic vectors established by (3.21). Suppose the ground set $\mathcal{M} = \{1', \ldots, N'\}$ and a partition mapping $\pi : \{1', \ldots, N'\} \to \{1, \ldots, N\}$ such that $\cup_{i' \in \mathcal{M}} \pi(i') = \mathcal{M}$, and $\pi(i') \cap \pi(j') = \emptyset, i', j' \in \mathcal{M}, i' \neq j'$. That is, the set $\mathcal{M}$ is partitioned into N disjoint sets. Suppose the variables associated with $\mathcal{M}$ are $X_{1'}, \ldots, X_{N'}$. Now we define for $n \in \{1, \ldots, N\}$ the new vector-valued random variables $\mathbf{Y}_n = [X_{i'} | i' \in \pi^{-1}(n)]$. The associated entropic vector will have entropies $h_{\mathcal{A}} = r_M(\cup_{n\in\mathcal{A}} \pi^{-1}(n)) \log_2 q, \mathcal{A} \subseteq \{1, \ldots, N\}$, and is thus proportional to a *projection* of the original rank vector $\mathbf{r}$ keeping only those elements corresponding to all elements in a set in the partition appearing together. Thus, such a projection of $\Gamma_{N'}^q$ forms an inner bound to $\bar{\Gamma}_N^*$, which we will refer to as a *vector representable matroid inner bound* $\Gamma_{N,N'}^q$. As $N' \to \infty$, $\Gamma_{N,\infty}^q$ is the conic hull of all ranks of subspaces on $\mathbb{F}_q$. The union over all field sizes for $\Gamma_{N,\infty}^q$ is the conic hull of the set of ranks of subspaces. Similarly, if a vector in the rate region of a network is (a projection of) a vector $\mathbb{F}_q$-representable matroid rank, the representation $\mathbb{A}$ can be used as a linear code to achieve that rate vector, and this code is denoted as a *basic vector* $\mathbb{F}_q$ *code*. The time-sharing between such basic vector codes can achieve any point inside the rate region [22, 41].

All the bounds discussed in this section could be used in equation (3.15) and (3.16) to calculate bounds on rate regions for a network A. If we substitute the Shannon outer bound Γ_N into (3.15), we get

$$\mathcal{R}_o(\mathsf{A}) = \mathrm{proj}_{\boldsymbol{r},\boldsymbol{\omega}}(\Gamma_N \cap \mathcal{L}_{\mathsf{A}}). \tag{3.22}$$

Similarly, using the representable matroid inner bound Γ_N^q, the vector representable matroid inner bound $\Gamma_{N,N'}^q$ and $\Gamma_{N,\infty}^q$ are substituted into (3.16), to obtain

$$\mathcal{R}_{s,q}(\mathsf{A}) = \mathrm{proj}_{\boldsymbol{r},\boldsymbol{\omega}}(\Gamma_N^q \cap \mathcal{L}_{\mathsf{A}}), \tag{3.23}$$

$$\mathcal{R}_q^{N'}(\mathsf{A}) = \mathrm{proj}_{\boldsymbol{r},\boldsymbol{\omega}}(\Gamma_{N,N'}^q \cap \mathcal{L}_{\mathsf{A}}), \tag{3.24}$$

$$\mathcal{R}_q(\mathsf{A}) = \mathrm{proj}_{\boldsymbol{r},\boldsymbol{\omega}}(\Gamma_{N,\infty}^q \cap \mathcal{L}_{\mathsf{A}}). \tag{3.25}$$

We will present the experimental results utilizing these bounds to calculate the capacity regions of various networks in §3.5. However, before we do this, we will first aim to generate a list of network coding problems to which we may apply our computations and thereby calculate their rate regions. In order to tackle the largest collection of networks possible in this study, in the next section we will seek to obtain a minimal problem description for each network coding problem instance, removing any redundant sources, edges, or nodes.

3.3 Minimality Reductions on Networks

Though in principle, any network coding problem as described in §3.2 forms a valid network coding problem, such a problem can include networks with nodes, edges, and sources which are completely extraneous and un-necessary from the standpoint of determining the underlying network coding problem's capacity region. To deal with this, in this section, we show how to, given any network coding problem defined in the form §3.2, form a problem of equal or lesser number of sources, edges, or nodes, which contains no obviously inessential nodes, edges, or sources, and from whose rate region it is trivial to calculate the rate region of the original network. Network coding problems without such extraneous and un-necessary components will be called minimal, and we shall proceed by first precisely defining what a minimal network coding problem, or, for short, minimal network, is and then showing, via Theorem 11, how to map a non-minimal network to a minimal network, then form the capacity region of the non-minimal network directly from the minimal one.

Definition 13: An acyclic network instance $\mathsf{A} = \{\mathcal{V}, \mathcal{E}, (\beta(t), t \in \mathcal{V})\}$ is *minimal* if it obeys the following

constraints:

Source minimality:

(C1) all sources connect with the set of intermediate nodes: $\forall s \in \mathcal{S}$, $\mathrm{Hd}(\mathrm{Out}(s)) \cap \mathcal{G} \neq \emptyset$;

(C2) sinks do not demand sources to which they are directly connected: $\forall s \in \mathcal{S}$, $t \in \mathcal{T}$, if $t \in \mathrm{Hd}(\mathrm{Out}(s))$ then $s \notin \beta(t)$;

(C3) every source is demanded by at least one sink: $\forall s \in \mathcal{S}$, $\exists\, t \in \mathcal{T}$ such that $Y_s \in \beta(t)$;

(C4) sources connected to the same intermediate node and demanded by the same set of sinks should be merged: $\nexists s, s' \in \mathcal{S}$ such that $\mathrm{Hd}(\mathrm{Out}(s)) = \mathrm{Hd}(\mathrm{Out}(s'))$ and $\gamma(s) = \gamma(s')$, where $\gamma(s) = \{t \in \mathcal{T} | s \in \beta(t)\}$;

Node minimality:

(C5) intermediate nodes with identical inputs should be merged: $\nexists\, k, l \in \mathcal{G}$ such that $\mathrm{In}(k) = \mathrm{In}(l)$;

(C6) intermediate nodes should have nonempty inputs and outputs, and sink nodes should have nonempty inputs: $\forall g \in \mathcal{G}, t \in \mathcal{T}, \mathrm{In}(g) \neq \emptyset, \mathrm{Out}(g) \neq \emptyset, \mathrm{In}(t) \neq \emptyset$;

Edge minimality:

(C7) all hyperedges must have at least one head: $\nexists e \in \mathcal{E}$ such that $\mathrm{Hd}(e) = \emptyset$;

(C8) identical edges should be merged: $\nexists e_i, e_j \in \mathcal{E}$ with $\mathrm{Tl}(e_i) = \mathrm{Tl}(e_j)$, $\mathrm{Hd}(e_i) = \mathrm{Hd}(e_j)$;

(C9) intermediate nodes with unit in and out degree, and whose in edge is not a hyperedge, should be removed: $\nexists e_i, e_j \in \mathcal{E}, g \in \mathcal{G}$ such that $\mathrm{In}(g) = e_i$, $\mathrm{Hd}(e_i) = g$, $\mathrm{Out}(g) = e_j$;

Sink minimality:

(C10) there must exist a path to a sink from every source wanted by that sink: $\forall t \in \mathcal{T}, \beta(t) \subseteq \sigma(t)$, where $\sigma(t) = \{k \in \mathcal{S} | \exists$ a path from k to $t\}$;

(C11) every pair of sinks must have a distinct set of incoming edges: $\forall t_i, t_j \in \mathcal{T}, i \neq j$, $\mathrm{In}(t_i) \neq \mathrm{In}(t_j)$;

(C12) if one sink receives a superset of inputs of a second sink, then the two sinks should have no common sources in demand: If $\mathrm{In}(t_i) \subseteq \mathrm{In}(t_j)$, then $\beta(t_i) \cap \beta(t_j) = \emptyset$;

(C13) if one sink receives a superset of inputs of a second sink, then the sink with superset input should not have direct access to the sources that demanded by the sink with subset input: If $\mathrm{In}(t_i) \subseteq \mathrm{In}(t_j)$ then $t_j \notin \mathrm{Hd}(\mathrm{Out}(s))$ for all $s \in \beta(t_i)$.

Connectivity:

(**C14**) the direct graph associated with the network A is weakly connected.

To better highlight this definition of network minimality, we presently explain the conditions involved in greater detail. The first condition (**C1**) requires that all sources must be connected with the network, for otherwise there is no way for anything to decode them at any entropy rate other than zero, and hence they would be extraneous. The condition (**C2**) holds because otherwise the demand of this sink will always be trivially satisfied, hence removing this reconstruction constraint will not alter the rate region. Note that other sources not demanded by a given sink can be directly available to that sink as side information (e.g., as in index coding problems), as long as condition (**C13**) is satisfied. The condition (**C3**) indicates that each source must be demanded by some sink nodes, for otherwise it is extraneous and can be removed. The condition (**C4**) says that no two sources have exactly the same paths and set of demanders (sinks requesting the source), because in that case the network can be simplified by combining the two sources as a super-source. The condition (**C5**) requires that no two intermediate nodes have exactly the same input, for otherwise the two nodes can be combined. The condition (**C6**) requires that no nodes have empty input except the source nodes, for otherwise these nodes are useless and extraneous from the standpoint of satisfying the network coding problem. The condition (**C7**) requires that every edge variable must be in use in the network, for otherwise it is also extraneous and can be removed. The condition (**C8**) guarantees that there is no duplication of hyperedges, for otherwise they can be merged with one another. The condition (**C9**) says that there is no trivial relay node with only one non-hyperedge input and output, for otherwise the head of the input edge can be replaced with the head of the output edge. The condition (**C10**) reflects the fact that a sink can only decode the sources to which it has at least one path of access, and any demanded source not meeting this constraint will be forced to have entropy rate of zero. The condition (**C11**) indicates the trivial requirement that no two decoders should have the same input, for otherwise these two decoders can be combined. The condition (**C12**) simply stipulates that implied capabilities of sink nodes are not to be stated, but rather inferred from the implications. In particular, if $\mathrm{In}(t_i) \subseteq \mathrm{In}(t_j)$, and $\beta(t_i) \cap \beta(t_j) \neq \emptyset$, the decoding ability of $\beta(t_i)$ is implied at t_j: pursuing minimality, we only let t_j demand extra sources, if any. The condition (**C13**) is also necessary because the availability of s is already implied by having access to $\mathrm{In}(t_i)$, hence, there is no need to have direct access to s.

As shown in Definition 13, a network may have some redundancies that we are not interested in and the network can be further reduced for easier solvability. In this section, we will show why

the minimality reductions are necessary by showing that with the redundancies the rate region can be easily derived based on the network without redundancies. Following the same order of the constraints (**C1**)–(**C14**), we give the actions on each reduction and how the rate region of network with that redundancy can be derived.

Theorem 11: Suppose a network instance $\mathsf{A} = \{\mathcal{V}, \mathcal{E}, (\beta(t), t \in \mathcal{V})\}$, which has rate region $\mathcal{R}_*(\mathsf{A})$, is a reduction from another network $\mathsf{A}' = \{\mathcal{V}', \mathcal{E}', (\beta(t), t \in \mathcal{V}')\}$, which has rate region $\mathcal{R}_*(\mathsf{A}')$, by removing one of the redundancies specified in (**C1**)–(**C14**). Then:

Source minimality:

(**D1**) if $\exists s' \in \mathcal{S}', \mathrm{Hd}(\mathrm{Out}(s')) \cap \mathcal{G}' = \emptyset$, A will be A′ with s' removed and if s' was demanded by some sink in A′, then

$$\mathcal{R}_*(\mathsf{A}') := \left\{\mathbf{R} | \mathbf{R}_{\setminus s'} \in \mathcal{R}_*(\mathsf{A}), H(Y_{s'}) = 0\right\}; \tag{3.26}$$

and if s' was not demanded by any sink in A′, then

$$\mathcal{R}(\mathsf{A}') := \left\{\mathbf{R} | \mathbf{R}_{\setminus s} \in \mathcal{R}_*(\mathsf{A}), H(Y_{s'}) \geq 0\right\}; \tag{3.27}$$

(**D2**) if $\exists s' \in \mathcal{S}', t' \in \mathcal{T}'$, such that $t' \in \mathrm{Hd}(\mathrm{Out}(s'))$ and $s' \in \beta(t')$, A will be A′ with $\mathrm{In}(t) = \mathrm{In}(t') \setminus s'$ and $\beta(t) = \beta(t') \setminus s'$. Further, $\mathcal{R}_*(\mathsf{A}') = \mathcal{R}_*(\mathsf{A})$;

(**D3**) if $\exists s' \in \mathcal{S}'$, such that $\forall\, t' \in \mathcal{T}'$, $Y_{s'} \notin \beta(t')$, A will be A′ with removal of the redundant source s' and

$$\mathcal{R}_*(\mathsf{A}') := \left\{\mathbf{R} | \mathbf{R}_{\setminus s'} \in \mathcal{R}_*(\mathsf{A}), H(Y_{s'}) = 0\right\}; \tag{3.28}$$

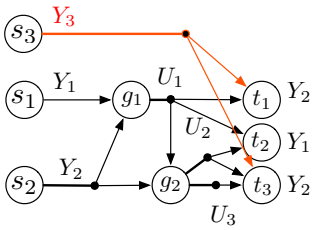
(**D4**) if $\exists s, s' \in \mathcal{S}'$ such that $\mathrm{Hd}(\mathrm{Out}(s)) = \mathrm{Hd}(\mathrm{Out}(s'))$ and $\gamma(s) = \gamma(s')$, A will be A′ with sources s, s' merged and

$$\mathcal{R}_*(\mathsf{A}') = \left\{\mathbf{R} | (\mathbf{R}^T_{\setminus\{s,s'\}}, H(Y_s) + H(Y_{s'}))^T \in \mathcal{R}_*(\mathsf{A})\right\}; \tag{3.29}$$

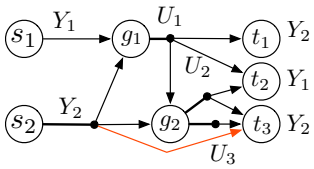
i.e. replace $H(Y_s)$ in $\mathcal{R}_*(\mathsf{A})$ with $H(Y_s) + H(Y_{s'})$ to get $\mathcal{R}_*(\mathsf{A}')$.

Node minimality:

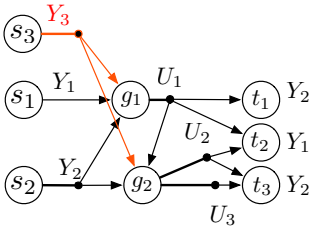
(**D5**) if $\exists\, k', l' \in \mathcal{G}$ such that $\mathrm{In}(k') = \mathrm{In}(l')$, A will be A′ with k', l' merged so that $\mathrm{In}(k) = \mathrm{In}(k') = \mathrm{In}(l'), \mathrm{Out}(k) = \mathrm{Out}(k') \cup \mathrm{Out}(l')$, and $\mathcal{G} = \mathcal{G}' \setminus l'$. Further, $\mathcal{R}_*(\mathsf{A}) = \mathcal{R}_*(\mathsf{A}')$;



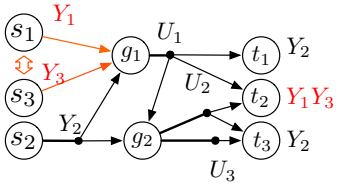
(a) C1: source node s_3 does not connected with any intermediate node, and thus is redundant.



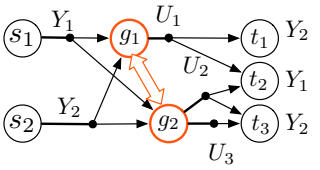
(b) C2: sink t_3 has direct access to Y_2, the demand of Y_2 is trivially satisfied and thus t_3 is redundant.



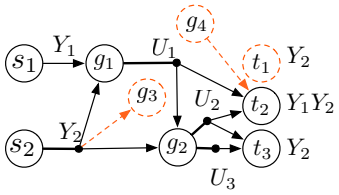
(c) C3: source variable Y_3 is not demanded by any sink, and thus is redundant.



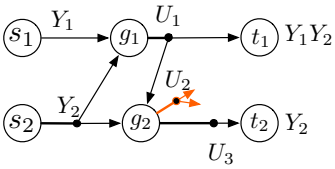
(d) C4: sources Y_1, Y_3 have exactly the same output and demanders, and thus can be combined.



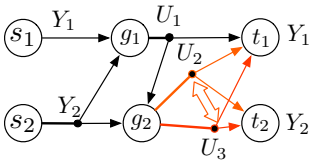
(e) C5: node g_1, g_2 have same input, and thus can be combined.



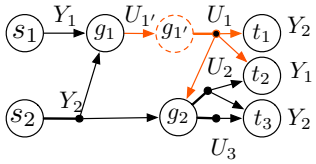
(f) C6: node g_3, g_4 and sink t_1 have empty input/ output, and thus are redundant.



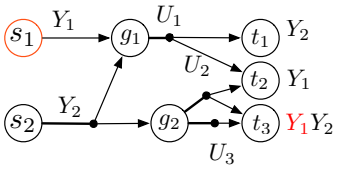
(g) C7: edge U_2 is not connected to any other nodes, and thus is redundant.



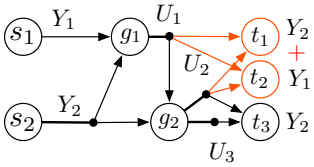
(h) C8: edges U_2, U_3 have exactly the same input and output nodes, and thus can be combined.



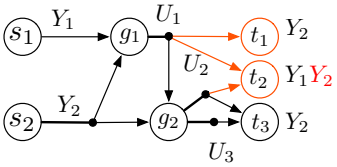
(i) C9: node g_2 has exactly one input and one output, and thus the input and output edges can be combined.



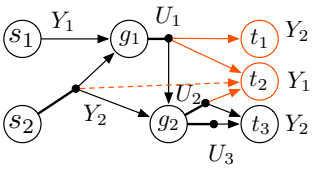
(j) C10: sink t_3 has no access to s_1 but demands Y_1, so the only way to satisfy it is s_1 is sending no information. Then s_1 is redundant.



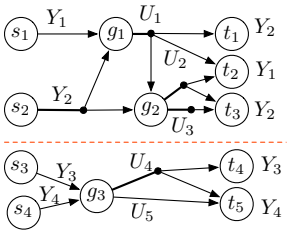
(k) C11: sinks t_1, t_2 have exactly the same input and thus can be combined into one sink node.



(l) C12: since t_2 also has access to U_1 and can decode Y_2 (implied by t_1), thus, Y_2 can be deleted from the demands of t_2.



(m) C13: since t_2 also has access to U_1 and can decode Y_2 (implied by t_1), thus, there is no need to have direct access to Y_2.



(n) C14: each connected component can be viewed as a separate network instance.

*Figure 3.5: Examples to demonstrate the minimality conditions (**C1**–**C14**).*

(D6) if $\exists g' \in \mathcal{G}$, or $t' \in \mathcal{T}$, such that $\mathrm{In}(g) \neq \emptyset$, or $\mathrm{Out}(g) \neq \emptyset$, or $\mathrm{In}(t) \neq \emptyset$, A will be A with removal of the redundant node(s) and $\mathcal{R}_*(\mathsf{A}') = \mathcal{R}_*(\mathsf{A})$;

Edge minimality:

(D7) if $\exists e' \in \mathcal{E}'$ such that $\mathsf{Hd}(e') = \emptyset$, A will be A′ with removal of edge e' and

$$\mathcal{R}_*(\mathsf{A}') = \{\mathbf{R} | \mathbf{R}_{\setminus e'} \in \mathcal{R}_*(\mathsf{A}), R_{e'} \geq 0\}; \tag{3.30}$$

(D8) if $\exists e, e' \in \mathcal{E}'$ with $\mathrm{Tl}(e) = \mathrm{Tl}(e')$, $\mathrm{Hd}(e) = \mathrm{Hd}(e')$, A will be A′ with edges e, e' merged as e and

$$\mathcal{R}_*(\mathsf{A}') = \left\{\mathbf{R} | (\mathbf{R}^T_{\setminus\{e,e'\}}, R_e + R_{e'})^T \in \mathcal{R}_*(\mathsf{A})\right\}; \tag{3.31}$$

i.e. replace R_e in $\mathcal{R}_*(\mathsf{A})$ with $R_e + R_{e'}$ to get $\mathcal{R}_*(\mathsf{A}')$.

(D9) if $\exists e, e' \in \mathcal{E}', g' \in \mathcal{G}'$ such that $\mathrm{In}(g') = e$, $\mathrm{Hd}(e) = g'$, $\mathrm{Out}(g') = e'$, then A will be A′ with contraction of node g' so that a new edge e_i will replace e, e' by directly connecting $\mathrm{Tl}(e)$ and $\mathrm{Hd}(e')$. Further,

$$\mathcal{R}_*(\mathsf{A}') = \left\{\mathbf{R} | (\mathbf{R}^T_{\setminus\{e,e'\}}, \min\{R_e, R_{e'}\})^T \in \mathcal{R}_*(\mathsf{A})\right\}; \tag{3.32}$$

i.e. replace R_e in obtaining $\mathcal{R}_*(\mathsf{A})$ with $\min\{R_e, R_{e'}\}$ to get $\mathcal{R}_*(\mathsf{A}')$.

Sink minimality:

(D10) if $\exists t' \in \mathcal{T}', s' \in \mathcal{S}'$, such that $s' \in \beta(t')$ but $s' \notin \sigma(t')$, then A will be A′ with deleting s' and further

$$\mathcal{R}_*(\mathsf{A}') = \{\mathbf{R} | \mathbf{R}_{\setminus s'} \in \mathcal{R}_*(\mathsf{A}), H(Y_{s'}) = 0\}; \tag{3.33}$$

(D11) if $\exists t, t' \in \mathcal{T}', t \neq t'$, such that $\mathrm{In}(t) = \mathrm{In}(t)$, then A will be A′ with sinks t, t' merged and $\mathcal{R}_*(\mathsf{A}') = \mathcal{R}_*(\mathsf{A})$;

(D12) if $\exists t, t'$ such that $\mathrm{In}(t) \subseteq \mathrm{In}(t')$ and $\beta(t) \cap \beta(t') \neq \emptyset$, then A will be A′ with removal of $\beta(t) \cap \beta(t')$ from $\beta(t')$ and $\mathcal{R}_*(\mathsf{A}') = \mathcal{R}_*(\mathsf{A})$;

(D13) if $\exists t, t, s' \in \beta(t)$ such that If $\mathrm{In}(t) \subseteq \mathrm{In}(t)$ and $t' \in \mathrm{Hd}(\mathrm{Out}(s'))$, then A will be A′ with removal of s' from $\mathrm{In}(t')$ and $\mathcal{R}_*(\mathsf{A}') = \mathcal{R}_*(\mathsf{A})$;

Connectivity:

(D14) if A' is not weakly connected and $\mathsf{A}_1, \mathsf{A}_2$ are two weakly disconnected components, then $\mathcal{R}_*(\mathsf{A}') = \mathcal{R}_*(\mathsf{A}_1) \cup \mathcal{R}_*(\mathsf{A}_2)$.

Proof: (**D1**) is easy to see since s' is not transmitting information to the network, though it might be connected with some sink nodes. When it is demanded by some sink, the only constraint on $H(Y_{s'})$ is trivially $H(Y_{s'}) \geq 0$. Otherwise, its rate is zero, i.e., $H(Y_{s'}) = 0$.

(**D2**) holds because the demand of s' at sink t' is trivially satisfied if it has direct access to s'. The constraint has no impact on the rate region of the network.

If a source is not demanded by anyone, it is not of interest for the network. Therefore, any use of network transmission on this source is a waste of resource and it should be eliminated. Therefore, (**D3**) holds.

When two sources have exactly the same connections and are demanded by same sinks, they can be simply viewed as a combined source. Since the source entropies are variables in the rate region expression, it is equivalent to make s as the combined source. Thus, (**D4**) holds.

Since a node is allowed to have different output hyperedges, for pursuing minimality of representation of a network, two nodes have same input can be represented as one node. Thus, (**D5**) is necessary and the merge of nodes with same input does not impact the coding on edges and rate region.

(**D6**) is trivial because if the input or output of a node, or sink, is empty, it is redundant to the network and removal of it does not impact the rate region of the network.

(**D7**) is trivial since the edge with no outgoing connection does not carry any transferrable information and hence its capacity is zero. It should be removed.

(**D8**) can be shown as follows. If $\mathcal{R}_*(\mathsf{A})$ is known and when edge e in A is represented as two parallel edges e, e' so that the network becomes A', then the constraint on e, e' in A' is simply to make sure the total capacity $R_e + R_{e'}$ can allow the information to be transmitted from the tail node to head nodes. Time sharing among the two edges will achieve this. Therefore, replace the R_e in $\mathcal{R}_*(\mathsf{A})$ with $R_e + R_{e'}$ will obtain the rate region $\mathcal{R}_*(\mathsf{A}')$. Therefore, (**D8**) holds.

(**D9**) is not difficult to understand since the capacities on the two edges in A' is lower bounded by $\min(R_e, R_{e'})$. In A, edge e will represent the bottle network edge between e, e' in A. Hence, when e in A is represented as e, e' and it becomes A', in obtaining $\mathcal{R}_*(\mathsf{A})$, we simply replace R_e with $\min(R_e, R_{e'}$ to get $\mathcal{R}_*(\mathsf{A}')$. Thus, (**D9**) holds.

If a sink demands a source that it does not have access to, the only way to satisfy this network constraint is the source entropy is 0. Hence, (**D10**) holds. The removal of this redundant source

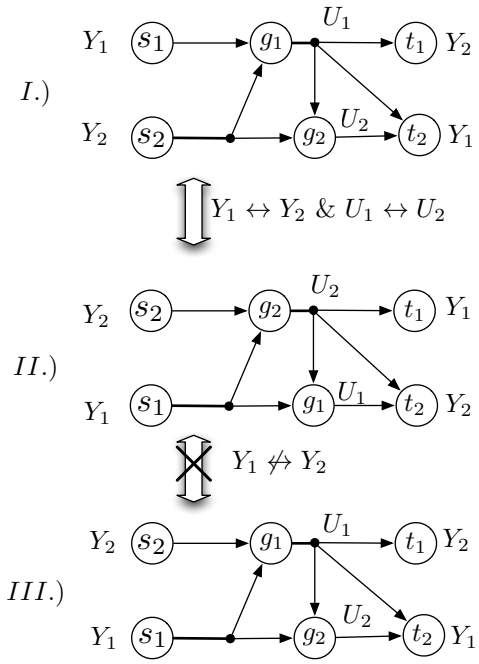


Figure 3.6: A demonstration of the equivalence between network coding problems via isomorphism: networks I and II are equivalent because II can be obtained by permuting Y_1, Y_2 and U_1, U_2 simultaneously. However, network III is not equivalent to I or II, because the demands at the sinks do not reflect the same permutation of Y_1, Y_2 as is necessary on the source side.

does not impact the rate region of the network with remaining variables.

Similar as (**D3**), the two sink nodes with same input does not impact the rate region. Thus, (**D11**) holds. For pursuing minimality, they should be merged.

(**D12**) is easy to understand because the decoding ability of $\beta(t_{i'})$ at sink node $t_{j'}$ is implied by sink $t_{i'}$. The non-necessary repeated decoding constraints will not affect the rate region for this network. Thus, (**D12**) holds.

Similar as (**D12**), since the decoding ability of $\beta(t_{i'})$ has been implied, the direct access to any of the sources in $\beta(t_{i'})$ is not necessary and with the direct access does not affect the rate region. Therefore, (**D13**) holds.

(**D14**) is obviously true since each connected component is a small network. ■

As a demonstration of the reductions on networks, example are given in Fig. 3.5. After showing how to reduce a non-minimal network to minimal one, we discuss the enumeration of minimal networks of a particular size.

3.4 Enumeration of Non-isomorphic Minimal Networks

Even though the notion of network minimality (§3.3) boils down the set of network coding problem instances by removing parts of a network coding problem which are inessential, much more needs to be done to group network coding problem instances into appropriate equivalence classes. Although we have to use label sets to describe the edges and sources in order to specify a network coding problem instance (identifying a certain source as source number one, another as source number two, and so on), it is clear that the essence of the underlying network coding problem should be (and is) insensitive to these labels. For instance, it is intuitively clear that the first two problems depicted in Fig. 3.6 should be equivalent even though their labeled descriptions differ, while the the third problem should not be considered equivalent to the first two.

In a certain sense, having to label network coding problems in order to completely specify them obstructs our ability to work effectively with a class of problems. This is because one unlabeled network coding problem equivalence class typically consists of many labeled network coding problems. In principle, we could go about investigating the unlabeled problems by exhaustively listing labeled network coding problem obeying the minimality constraints, testing for equivalence under relabeling of the source and node or edge indices, and grouping them together into equivalence classes. A naïve such network coding problem enumeration algorithm is described in Appendix B.

However, listing networks by generating all variants of the labeled encoding as done in Appendix B becomes infeasible rapidly as the problem grows because of the large number of labeled networks in each equivalence class. As a more feasible alternative, it is desirable to find a method for directly cataloguing all (unlabeled) network coding problem equivalence classes by generating exactly (i.e., exclusively) one representative from each equivalence class directly, without isomorphism (equivalence) testing.

In order to develop such a method, and to explain the connection between its solution and other isomorphism-free exhaustive generation problems of a similar ilk, in this section we must first formalize a precise method for encoding a network coding problem instance in §3.4.1. With this encoding in hand, in §3.4.2, the notion of equivalence classes for network coding problem instances can be made precise as orbits in this labeled problem space under an appropriate group action. The generic algorithm *Leiterspiel* [17, 46], for computing orbits within the power set of subsets of some set $\mathcal{X}$ on which a group $\mathbf{G}$ acts, can then be applied, together with some other standard orbit computation techniques in computational group theory [47,48], and some small further modification, in order to provide the desired non-isomorphic network coding problem list generation method in

3.4.3.

3.4.1 Encoding a Network Coding Problem

One way to concretely specify a network coding problem with K sources and $|\mathcal{E}_U|$ edges that can obey the minimality conditions (**C1-C14**) is as an ordered pair $(\mathcal{Q}, \mathcal{W})$ consisting of a set $\mathcal{Q}$ of edge definitions $\mathcal{Q} \subseteq \{(i, \mathcal{A}) | i \in \{K+1, \ldots, K+|\mathcal{E}_U|\},\ \mathcal{A} \subseteq \{1, \ldots, K+|\mathcal{E}_U|\} \setminus \{i\},\ |\mathcal{A}| > 0\}$, and a set $\mathcal{W}$ of sink definitions $\mathcal{W} \subseteq \{(i, \mathcal{A}) | i \in \{1, \ldots, K\},\ \mathcal{A} \subseteq \{1, \ldots, K+|\mathcal{E}_U|\} \setminus \{i\}\}$. Here, the sources are associated with labels $\{1, ..., K\}$ and the edges are associated with labels $\{K+1, \ldots, K+|\mathcal{E}_U|\}$. Each $(i, \mathcal{A}) \in \mathcal{Q}$ indicates that the edge $i \in \mathcal{E}_U$ is encoded exclusively from the sources and edges in $\mathcal{A}$, and hence represents the information that $\mathcal{A} = \mathrm{In}(\mathrm{Tl}(i))$. Furthermore, each sink definition $(i, \mathcal{A}) \in \mathcal{W}$ represents the information that there is a sink node whose inputs are $\mathcal{A}$ and which decodes source i as its output. Note that there are $|\mathcal{E}_U|$ edges in the network, each of which must have some input according to condition (**C6**). We additionally have the requirement that $|\mathcal{Q}| = |\mathcal{E}_U|$, and, to ensure that no edge is multiply defined, we must have that if $(i, \mathcal{A})$ and $(i', \mathcal{A}')$ are two different elements in $\mathcal{Q}$, then $i \neq i'$. As the same source may be decoded at multiple sinks, there is no such requirement for $\mathcal{W}$.

As is illustrated in Figures 3.7(a) and 3.7(b), this edge-based definition of the directed hyper graph included in a network coding problem instance can provide a more parsimonious representation than a node-based representation, and as every edge in the network for a network coding problem is associated with a random variable, this representation maps more easily to the entropic constraints than the node representation of the directed acyclic hypergraph does. Additionally it is beneficial because it is guaranteed to obey several of the key minimality constraints. In particular, the representation ensures that there are no redundant nodes (**C5**),(**C11**), since the intermediate nodes are associated directly the elements of the set $\{\mathcal{A} |\ \exists i,\ (i, \mathcal{A}) \in \mathcal{Q}\}$ and the sink nodes are associated directly with $\{\mathcal{A} |\ \exists i,\ (i, \mathcal{A}) \in \mathcal{W}\}$. Representing $\mathcal{Q}$ as a set (rather than a multi-set) also ensures that (**C8**) is always obeyed, since such a parallel edge would be a repeated element in $\mathcal{Q}$.

3.4.2 Expressing Network Equivalence with a Group Action

Another benefit of the representation of the network coding problem as the ordered pair $(\mathcal{Q}, \mathcal{W})$ is that it enables the notion of network isomorphism to be appropriately defined. In particular, let $\mathbf{G} := S_{\{1,2,\ldots,K\}} \times S_{\{K+1,\ldots,K+|\mathcal{E}_U|\}}$ be the direct product of the symmetric group of all permutations of the set $\{1, 2, \ldots, K\}$ of source indices and the symmetric group of all permutations of the set $\{K+1, \ldots, K+|\mathcal{E}_U|\}$ of edge indices. The group $\mathbf{G}$ acts in a natural manner on the elements of

the sets $\mathcal{Q}, \mathcal{W}$ of edge and sink definitions. In particular, let $\pi \in \mathbf{G}$ be a permutation in $\mathbf{G}$, then the group action maps

$$\pi((i, \mathcal{A})) \mapsto (\pi(i), \pi(\mathcal{A})) \tag{3.34}$$

with the usual interpretation that $\pi(\mathcal{A}) = \{\pi(j) | j \in \mathcal{A}\}$. This action extends to an action on the sets $\mathcal{Q}$ and $\mathcal{W}$ in the natural manner

$$\pi(\mathcal{Q}) \mapsto \{\pi((i, \mathcal{A})) | (i, \mathcal{A}) \in \mathcal{Q}\} \tag{3.35}$$

This action then extends further still to an action on the network $(\mathcal{Q}, \mathcal{W})$ via

$$\pi((\mathcal{Q}, \mathcal{W})) = (\pi(\mathcal{Q}), \pi(\mathcal{W})) \tag{3.36}$$

Two networks $(\mathcal{Q}_1, \mathcal{W}_1)$ and $(\mathcal{Q}_2, \mathcal{W}_2)$ are said to be *isomorphic*, or in the same equivalence class, if there is some permutation of $\pi \in \mathbf{G}$ such that $\pi((\mathcal{Q}_1, \mathcal{W}_1)) = (\mathcal{Q}_2, \mathcal{W}_2)$. In the language of group actions, two such pairs are isomorphic if they are in the same orbit under the group action, i.e. if $(\mathcal{Q}_2, \mathcal{W}_2) \in \{\pi((\mathcal{Q}_1, \mathcal{Q}_2)) \,|\pi \in \mathbf{G}\} =: \mathcal{O}_{(\mathcal{Q}_1, \mathcal{W}_1)}$. In other words, the equivalence classes of networks are identified with the orbits in the set of all valid minimal problem description pairs $(\mathcal{Q}, \mathcal{W})$ under the action of $\mathbf{G}$.

We elect to represent each equivalence class with its *canonical network*, which is the element in each orbit that is least in a lexicographic sense. Note that this lexicographic (i.e., dictionary) order is well-defined, as we can compare two subsets $\mathcal{A}$ and $\mathcal{A}'$ by viewing their members in increasing order (under the usual ordering of the integers $\{1, \ldots, |\mathcal{E}_U| + K\}$) and lexicographically comparing them. This then implies that we can lexicographically order the ordered pairs $(i, \mathcal{A})$ according to $(i, \mathcal{A}) > (j, \mathcal{A}')$ if $j < i$ or $i = j$ and $\mathcal{A}' < \mathcal{A}$ under this lexicographic ordering. Since the elements of $\mathcal{Q}$ and $\mathcal{W}$ are of the form $(i, \mathcal{A})$, this in turn means that they can be ordered in increasing order, and then also lexicographically compared, enabling comparison of two edge definition sets $\mathcal{Q}$ and $\mathcal{Q}'$ or two sink definition sets $\mathcal{W}$ and $\mathcal{W}'$. Finally, one can then use these orderings to define the lexicographic order on the network ordered pairs $(\mathcal{Q}, \mathcal{W})$. The element in an orbit $\mathcal{O}_{(\mathcal{Q}, \mathcal{W})}$ which is minimal under this lexicographic ordering will be the canonical representative for the orbit.

A key basic result in the theory of group actions, the *Orbit Stabilizer Theorem*, states that the number of elements in an orbit, which in our problem is the number of networks that are isomorphic to a given network, is equivalent to the ratio of the size of the acting group $\mathbf{G}$ and its stabilizer

subgroup $\mathbf{G}_{(\mathcal{Q},\mathcal{W})}$ of any element selected from the orbit:

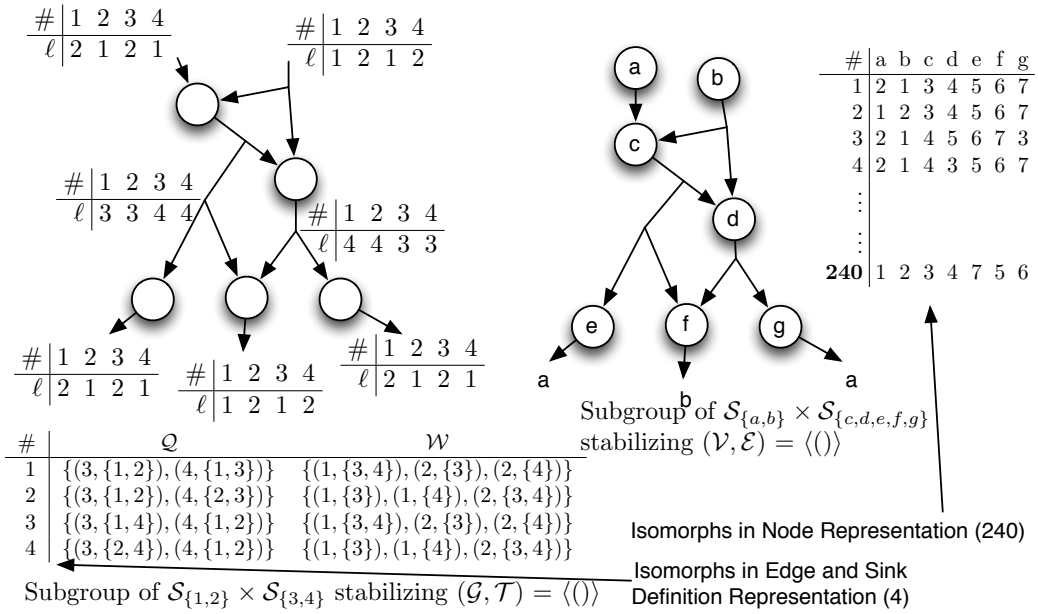
$$|\{\pi((\mathcal{Q},\mathcal{W}))\,|\pi \in \mathbf{G}\}| = \left|\mathcal{O}_{(\mathcal{Q},\mathcal{W})}\right| = \frac{|\mathbf{G}|}{\left|\mathbf{G}_{(\mathcal{Q},\mathcal{W})}\right|}, \quad \mathbf{G}_{(\mathcal{Q},\mathcal{W})} := \{\pi \in \mathbf{G}\,|\pi((\mathcal{Q},\mathcal{W})) = (\mathcal{Q},\mathcal{W})\} \quad (3.37)$$

Note that, because it leaves the sets of edges, decoder demands, and topology constraints set-wise invariant, the elements of the stabilizer subgroup $\mathbf{G}_{(\mathcal{Q},\mathcal{W})}$ also leave the set of rate region constraints (3.10), (3.11), (3.12), (3.13) invariant. Such a group of permutations on sources and edges is called the *network symmetry group*, and is the subject of a separate investigation [42, 43]. This network symmetry group plays a role in the present study because, as depicted in Figures 3.7(a) and 3.7(b), by the orbit stabilizer theorem mentioned above, it determines the number of networks equivalent to a given canonical network (the representative we will select from the orbit).

In particular, Fig. 3.7(a) shows the orbit of a $(2,2)$ network $(\mathcal{Q},\mathcal{W})$ whose stabilizer subgroup (i.e., network symmetry group) is simply the identity, and hence has only one element. In this instance, the number of isomorphic labeled network coding problems in this equivalence class is then $|\mathbf{G}| = \left|\mathbf{S}_{\{1,2\}} \times \mathbf{S}_{\{3,4\}}\right| = 4$ in the edge representation, as shown at the right. Even this tiny example demonstrates well the benefits of using the encoding of a network coding problem into the more parsimonious representation $(\mathcal{Q},\mathcal{W})$ as an alternative to the encoding via the node representation hypergraph $(\mathcal{V},\mathcal{E})$ and the sink demands $\beta(\cdot)$ because the size of the group acting on the node representation is $|\mathbf{S}_{\{a,b\}} \times \mathbf{S}_{\{c,d,e,f,g\}}| = 240$, and, as the stabilizer subgroup in the node representation has the same order (1), the number of isomorphic networks represented in the node based representation is 240.

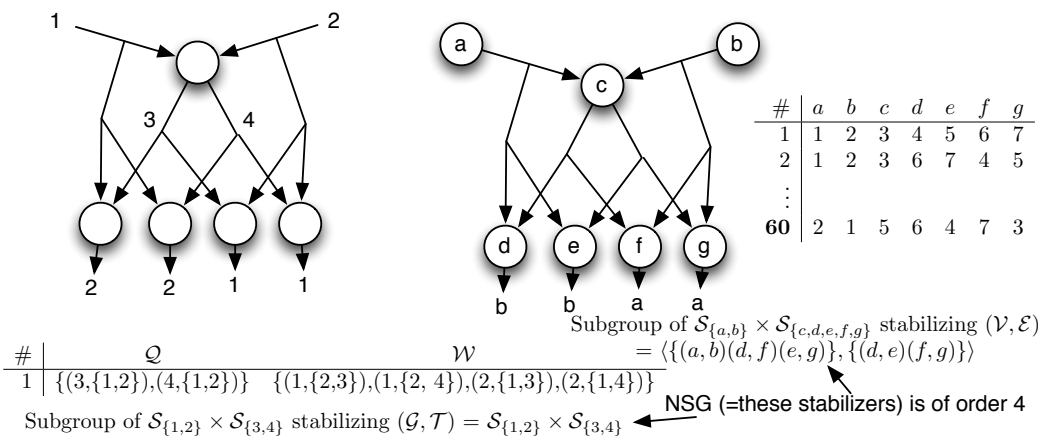
By contrast, Fig. 3.7(b) shows the orbit of a $(2,2)$ network $(\mathcal{Q},\mathcal{W})$ whose stabilizer subgroup i.e., network symmetry group) is the largest possible among $(2,2)$ networks, and has order 4. In this instance, the number of isomorphic labeled network coding problems in this equivalence class is $\frac{|\mathbf{G}|}{|\mathbf{G}_{(\mathcal{Q},\mathcal{W})}|} = 1$ in the edge representation. The stabilizer subgroup in the node representation is $\langle\{(a,b)(d,f)(e,g)\},\{(d,e)(f,g)\}\rangle$, which has the same order of 4, and hence there are $\frac{240}{4} = 60$ isomorphic network coding problems to this one in the node representation.

With a firm understanding of posing equivalence classes as orbits under group actions on appropriate parameterizations of the set of minimal network coding problems, as well as an appropriate notion of ordering and canonical representatives from these orbits, we are now ready to describe a computational group theory based method of enumerating non-isomorphic network coding problem instances.



#	$\mathcal{Q}$	$\mathcal{W}$
1	$\{(3,\{1,2\}),(4,\{1,3\})\}$	$\{(1,\{3,4\}),(2,\{3\}),(2,\{4\})\}$
2	$\{(3,\{1,2\}),(4,\{2,3\})\}$	$\{(1,\{3\}),(1,\{4\}),(2,\{3,4\})\}$
3	$\{(3,\{1,4\}),(4,\{1,2\})\}$	$\{(1,\{3,4\}),(2,\{3\}),(2,\{4\})\}$
4	$\{(3,\{2,4\}),(4,\{1,2\})\}$	$\{(1,\{3\}),(1,\{4\}),(2,\{3,4\})\}$

(a) All isomorphisms of a $(2,2)$ network with trivial symmetry group



#	$\mathcal{Q}$	$\mathcal{W}$
1	$\{(3,\{1,2\}),(4,\{1,2\})\}$	$\{(1,\{2,3\}),(1,\{2,4\}),(2,\{1,3\}),(2,\{1,4\})\}$

(b) All isomorphisms of a $(2,2)$ network with full symmetry group

Figure 3.7: Examples of $(2,2)$ network with all edge isomorphisms and node isomorphisms. The instance indices are marked by # and the labels are marked by l.

3.4.3 Network Enumeration/Listing Algorithm

Formalizing the notion of a canonical network via group actions on the set of minimal $(\mathcal{Q}, \mathcal{W})$ pairs enables one to partly develop a method for directly listing canonical networks based on techniques from computational group theory.

To solve this problem we can harness the algorithm *Leiterspiel*, loosely translated *snakes and ladders* [17, 46], which, given an orbit computation algorithm for computing orbits on some finite set $\mathcal{X}$ under a group $\mathbf{G}$ and its subgroups, provides a method for computing the orbits on the set $\mathcal{P}_i(\mathcal{X}) = \{\mathcal{Y} \subseteq \mathcal{X} | \, |\mathcal{Y}| = i\}$ of subsets from $\mathcal{X}$ of cardinality i, incrementally in i. In fact, the algorithm can also list directly only those canonical representatives of orbits for which some test function f returns 1, provided that the test function has the property that any subset of a set with $f = 1$ also has $f = 1$. This test function is useful for only listing those subsets in $\mathcal{P}_i(\mathcal{X})$ with a desired set of properties, provided these properties are inherited by subsets of a superset with that property.

The idea behind this method is that computational group theory packages such as GAP [48] or PERMLIB [47], come with methods that are easily capable of computing orbits of group actions $\mathbf{G}$ on sets $\mathcal{X}$, and that, while these methods do not scale well to the induced group actions on subsets $\mathcal{P}_i$, owing to the size of the set of all subsets, they can be integrated with Leiterspiel to provide orbits in the power set.

The pseudocode for Leiterspiel is depicted in Alg. 4. The notation represents orbits, for instance of the group $\mathbf{G}$ acting on the set $\mathcal{X}$, with a data structure containing three elements: *i*) a transversal T consisting of a set with one representative, the canonical one, from each orbit, *ii*) a stabilizer map σ which, for every element of the transversal $x \in T$, returns the associated stabilizer subgroup $\sigma(x) = \mathbf{G}_x$, and *iii*) a transporter map ϕ, which for every element in the set $x \in \mathcal{X}$ in which orbits are being computed, returns a group member $\pi \in \mathbf{G}_x$ such that $\pi(x)$ is the canonical representative of $\mathcal{O}_x$, the orbits of x. Additionally, $\mathbf{G}_{R,x}$ will denote the subgroup $\mathbf{G}$ which stabilizes the set R and the point x. The algorithm only builds sets in the transversals T_i associated with the orbits in $\mathcal{P}_i(\mathcal{X})$ incrementally in the cardinality i, taking care to indicate to the future when a possible extension has already been incorporated in the transversal via the use of a function ψ. See [17] pp. 709–710 for some more details and proofs about the proper operation of the algorithm.

To see how to apply and modify Leiterspiel for network coding problem enumeration, let $\mathcal{X}$ be

```
Input: orbit(G, P_i^f(X)) = (T_i, σ_i, φ_i)
Output: orbit(G, P_{i+1}^f(X)) = (T_{i+1}, σ_{i+1}, φ_{i+1})
for R ∈ T_i do
    compute orbit(G_R, X \ R) := (T_R, σ_R, φ_R);
end
T_{i+1} = ∅;
for R ∈ T_i (in increasing order) do
    for x ∈ T_R (in increasing order) with f(R ∪ {x}) = 1 and for which ψ_R(x) has not yet
    been defined do
        G_{R,x} := σ_R(x);
        H := G_{R,x};
        for all r ∈ R which are least in their H-orbit do
            t := φ_i((R \ {r}) ∪ {x}); S := ((R \ {r}) ∪ {x})t;
            h := φ_S(rt); y := rth;
            (now (R ∪ {x})th = S ∪ {y}, S ∈ T_i, y ∈ T_S);
            if S = R and y = x then
                H := ⟨H, th⟩;
            end
            else
                ψ_S(y) := (th)^{-1};
            end
        end
        append R ∪ {x} to T_{i+1};
        σ_{i+1}(R ∪ {x}) := H(= G_{R∪{x}});
    end
end
return (T_{i+1}, σ_{i+1}, φ_{i+1});
where the function φ_{i+1} is defined as:
Function φ_{i+1}(F);
z := max F, Z := F \ {z};
t := φ_I(Z); S := Zt; h := φ_S(zt), y := zth;
if ψ_S(y) has been defined then
    return thψ_S(y);
end
else
    return th;
end
```

Algorithm 4: Leiterspiel algorithm to incrementally obtain orbits on subets.

the set of possible edge definitions

$$\mathcal{X} := \{(i, \mathcal{A}) \,|\, i \in \{K+1, \ldots, K+|\mathcal{E}_U|\},\ \mathcal{A} \subseteq \{1, \ldots, K+|\mathcal{E}_U|\} \setminus \{i\}\} \tag{3.38}$$

For small to moderately sized networks, the orbits in $\mathcal{X}$ from $\mathbf{G}$ and its subgroups can be readily computed with modern computational group theory packages such as GAP [48] or PERMLIB [47]. Leiterspiel can be applied to first calculate the non-isomorphic candidates for the edge definition set $\mathcal{Q}$, as it is a subset of $\mathcal{X}$ with cardinality $|\mathcal{E}_U|$ obeying certain conditions associated with the definition of a network coding problem and its minimality (c.f. **C1**–**C14**). Next, for each non-isomorphic edge-definition $\mathcal{Q}$, a list of non-isomorphic sink-definitions $\mathcal{A}$, also constrained to obey problem definition and minimality conditions (**C1**–**C14**), can be created with a second application of Leiterspiel. The pseudo-code for the resulting generation/enumeration is provided in Alg. 5

Input: number of sources K, number of edges $|\mathcal{E}_U|$
Output: All non-isomorphic network instances $\mathcal{M}$
Initialization: $\mathcal{M} = \emptyset$;
Let $\mathcal{X} := \{(i, \mathcal{A}) \,|\, i \in \{K+1, \ldots, K+|\mathcal{E}_U|\},\ \mathcal{A} \subseteq \{1, \ldots, K+|\mathcal{E}_U|\} \setminus \{i\}\}$;
Let f_1 be the condition $\forall \mathcal{B} \subseteq \mathcal{X}$, $\nexists (i, \mathcal{A}), (i', \mathcal{A}')$ such that $i = i'$;
Let f_2 be the condition $\forall \mathcal{B} \subseteq \mathcal{X}$, $\mathcal{B}$ is acyclic;
Let acting group $G := \mathbf{S}_{\{1,\ldots,K\}} \times \mathbf{S}_{\{K+1,\ldots,K+|\mathcal{E}_U|\}}$;
Call Leiterspiel algorithm to incrementally get all candidate transversals up to $|\mathcal{E}_U|$;
$T_{|\mathcal{E}_U|} = \mathit{Leiterspiel}(G, \mathcal{P}^{f_1,f_2}_{|\mathcal{E}_U|}(\mathcal{X}))$;
$i \in \{1, \ldots, K\}, \mathcal{A} \subseteq \{1, \ldots, K+|\mathcal{E}_U|\} \setminus \{i\}$;
for *each* $\mathcal{Q} \in T_{|\mathcal{E}_U|}$ **do**
  if $\mathcal{Q}$ *obeys* ***(C1)*** **then**
    Let $\mathcal{Y} := \{(i, \mathcal{A}) \,|\, \exists$ a directed path in $\mathcal{Q}$ from i to at least one edge in $\mathcal{A}\}$;
    Let f_1 be the condition (**C12**): $\forall \mathcal{B} \subseteq \mathcal{Y}$, $\nexists (i, \mathcal{A}), (i, \mathcal{A}')$ such that $\mathcal{A} \subset \mathcal{A}'$;
    Let f_2 be the condition (**C13**): $\forall \mathcal{B} \subseteq \mathcal{Y}$, if $(i, \mathcal{A}), (i', \mathcal{A}') \in \mathcal{B}$ and $\mathcal{A} \subset \mathcal{A}'$, then $(i', \mathcal{A}'') \notin \mathcal{B}$, where $i \in \mathcal{A}''$;
    Let acting group $G := \mathbf{S}_{\{1,\ldots,K\}} \times \mathbf{S}_{\{K+1,\ldots,K+|\mathcal{E}_U|\}}$;
    Call Leiterspiel algorithm to incrementally get all candidate transversals up to no new element can be added obeying (**C12**,**C13**);
    $T_K = \mathit{Leiterspiel}(G, \mathcal{P}^{f_1,f_2}_K(\mathcal{Y}))$;
    for *each* $\mathcal{W} \in T_K$ **do**
      if $(\mathcal{Q}, \mathcal{W})$ *obeys* ***(C3–C7)*** *and* ***(C14)*** **then**
        $\mathcal{M} = \mathcal{M} \cup (\mathcal{Q}, \mathcal{W})$;
      end
    end
  end
end

Algorithm 5: Enumerate all non-isomorphic $(K, |\mathcal{E}_U|)$ networks using Leiterspiel algorithm.

As outlined above, in the first stage of the enumeration/generation algorithm, Leiterspiel is applied to grow subsets from $\mathcal{X}$ of size i incrementally in i until $i = |\mathcal{E}_U|$. Some of the network

conditions have the appropriate inheritance properties, and hence can be incorporated as constraints into the constraint function f in the Leiterspiel process. These include

- **no repeated edge definitions**: If $\mathcal{B} \subseteq \mathcal{C} \subseteq \mathcal{X}$ and $\mathcal{C}$ has the property that no two of its edge definitions $(i, \mathcal{A})$ and $(i', \mathcal{A}')$ have $i = i'$, then so does $\mathcal{B}$. Hence, the constraint function f in the first application of Leiterspiel incorporates checks to ensure that no two edge definitions in the candidate subset define the same edge.

- **acyclicity**: If $\mathcal{B} \subseteq \mathcal{C} \subseteq \mathcal{X}$ and $\mathcal{C}$ is associated with an acyclic hyper graph, then so is $\mathcal{B}$. Hence, the constraint function f in the first application of Leiterspiel checks to determine if the subset in question is acyclic.

At the end of this first Leiterspiel process, some more canonical edge definition sets $\mathcal{Q}$ can be ruled as non-minimal owing to (**C1**), requiring that each source appears in the definition of at least one edge variable.

For each member of the resulting narrowed list of canonical edge definition sets $\mathcal{Q}$, we must then build a list of canonical representative sink definitions $\mathcal{W}$. This is done by first creating the ($\mathcal{Q}$-dependent) set of *valid* sink definitions

$$\mathcal{Y} := \{(i, \mathcal{A}) \,|\exists \text{ a directed path in } \mathcal{Q} \text{ from } i \text{ to at least one edge in } \mathcal{A}\} \tag{3.39}$$

which are crafted to obey the minimality conditions (**C10**) that the created sink (defined by its input which is the set of sources and edges in $\mathcal{A}$ in the sink definition $(i, \mathcal{A})$) must have at least one path in the hyper graph defined by $\mathcal{Q}$ to the source i it is demanding, and (**C2**) that is can not have a direct connection to the source it is demanding.

Leiterspiel is then applied to determine canonical (lexicographically minimal) representatives of sink definition sets $\mathcal{W}$, utilizing the associated stabilizer of the canonical edge definition set $\mathcal{Q}$ being extended as the group, with the test function f handling the following minimality conditions during the iterations:

- (**C12**): No two sink definitions $(i, \mathcal{A})$ and $(i', \mathcal{A}')$ with $i = i'$ can have $\mathcal{A} \subset \mathcal{A}'$. Clearly, this property is inherited by a subset of a sink definition $\mathcal{W}$ for a minimal network, and hence it can be incorporated in the constraint function f in the Leiterspiel generation.

- (**C13**): No sink may have direct access to a source which it would be capable of decoding due its containment within its in-edges of the in-edges of a second sink. This property is

also inherited over subsets, and hence can be incorporated in the constraint function f in Leiterspiel.

This second application of Leiterspiel to determine the list of canonical sink definition sets $\mathcal{W}$ for each canonical edge definition set $\mathcal{Q}$ does not have a definite cap on the cardinality of each of the canonical sink definition sets $\mathcal{W}$. Rather, subsets of all sizes are determined incrementally until there is no longer any canonical subset that can obey the constraint function associated with (**C12**) and (**C13**). Each of the candidate canonical sink definition sets $\mathcal{W}$ (of all different cardinalities) are then tested together with $\mathcal{Q}$ with the remaining conditions below, which do have the inheritance property necessary for incorporation as constraints earlier in the two stages of Leiterspiel processing.

- (**C3**): each source is demanded by at least one sink.
- (**C4**): no two sources are available to exactly the same edges and sinks and demanded by exactly the same sinks.
- (**C7**): Every edge variable has at least one other edge variable or sink that is dependent on it.
- (**C8**): No parallel edges.
- (**C9**): No redundant nodes with exactly one directed (non-hyper) edge in and one edge out.
- (**C14**): The final network coding problem (including both the sink and edge definitions) must yield a weakly connected graph.

Any pair of canonical $(\mathcal{Q}, \mathcal{W})$ surviving each of these checks is then added to the list of canonical minimal non-isomorphic network coding problem instances.

An additional pleasant side effect of the enumeration is that the stabilizer subgroups, i.e., the network symmetry groups [43], are directly provided by the second Leiterspiel. Harnessing these network symmetry groups provides a powerful technique to reduce the complex process of calculating the rate region for a network coding problem instance [42].

Although this method directly generates the canonical representatives from the network coding problem equivalence classes without ever listing other isomorphs within these classes, one can also use the stabilizer subgroups provided by Leiterspiel to directly enumerate the sizes of these equivalence classes of $(\mathcal{Q}, \mathcal{W})$ pairs, as described above via the orbit stabilizer theorem. Experiments summarized in Table 3.1 show that the number of isomorphic cases is substantially larger than the number of canonical representives/equivalence classes, and hence the extra effort to directly list only canonical

Table 3.1: Number of network coding problems of different sizes: $|\hat{\mathcal{M}}|$ represents the number of isomorphic networks and $|\mathcal{M}|$ represents the number of non-isomorphic networks.

$(K, \lvert\mathcal{E}_U\rvert)$	(1,2)	(1,3)	(2,1)	(2,2)	(2,3)	(3,1)	(3,2)	Total
$\lvert\mathcal{M}\rvert$	4	132	1	333	485 890	9	239 187	725 556
$\lvert\hat{\mathcal{M}}\rvert$	7	749	1	1 270	5 787 074	31	2 829 932	8 619 064
$\lvert\hat{\mathcal{M}}_n\rvert$	39	18,401	6	163,800	$\geq 2.2 \times 10^{12}$	582	$\geq 0.176 \times 10^{12}$	$\geq 2.3 \times 10^{12}$

networks is worthwhile. It is also worth noting that a node representation, utilizing a node based encoding of the hyper edges, would yield a substantially higher number of isomorphs.

3.4.4 Modification to Other Problem Types

A final point worth noting is that this algorithm is readily modified to handle listing canonical representatives of special network coding problem families contained within our general model, as described in §3.2.1. For instance, IDSC problems can be enumerated by simply defining $\mathcal{Q}$ to have each edge access all of the sources and no other edges, then continuing with the subsequent sink enumeration process. It is also easily adapted to enumerate only directed edges and match the more restrictive constraints described in the original Yan, Yeung, and Zhang [15] rate region papers.

3.4.5 Enumeration Results for Networks with Different Sizes

By using our enumeration tool with an implementation of the algorithms above, we obtained the list of canonical minimal network instances for different network coding problem sizes with $K = 1, 2, 3$ and $|\mathcal{E}_U| = 1, 2, 3$. While the whole list is available [49], we give the numbers of network problem instances in Table 3.1, where $|\hat{\mathcal{M}}|, |\mathcal{M}|, |\mathcal{M}_n|$ represent the number of canonical network coding problems (i.e., the number of equivalence classes), the number of edge descriptions of network coding problems including symmetries/equivalences, and the number of node descriptions of network coding problems including the symmetries/equivalences, respectively. As we can see from the table, the number of possibilities in the node representation of the network coding problems explodes very quickly, with the more than 2 trillion labeled node network coding problems covered by the study only necessitating a list of consisting of roughly 750,000 equivalence classes of network coding problems. That said, it is also important to note that the number of non-isomorphic network instances increases exponentially fast as network size grows. For instance, the number of non-isomorphic general network instances grows from 333 to $485,890$ (roughly, an increase of about 1500 times), when the network size grows from $(2, 2)$ to $(2, 3)$. To provide an illustration of the variety of networks that are encountered, Fig. 3.8 depicts all 46 of the 333 canonical minimal network coding problems of size $(2, 2)$ obeying the extra constraint that no sink has direct access to a source.

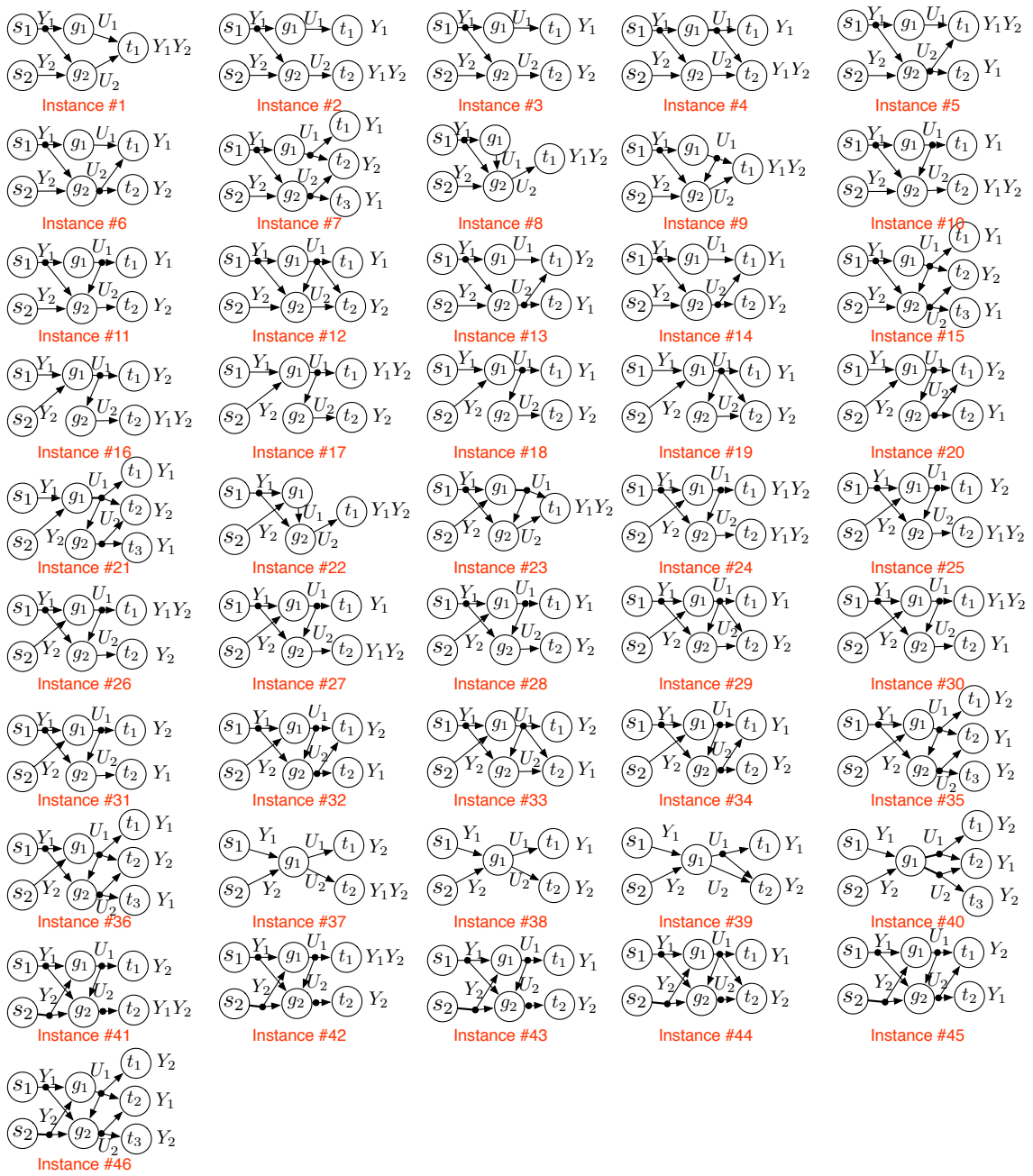


Figure 3.8: All 46 non-isomorphic network instances of $(2, 2)$ networks with the constraint that sinks do not have direct access to sources.

Table 3.2: List of numbers of IDSC configurations. $|\text{Sper}(\mathcal{E})|$ *represents the number of all Sperner families of* $\mathcal{E}$*,* $|\mathcal{M}'_n|$ *is the number of configurations in the node representation including isomorphs,* $|\mathcal{M}'|$ *is the number of configurations in the edge representation including isomorphs, and* $|\mathcal{M}|$ *is the number of all non-isomorphic configurations.*

$(K, \lvert\mathcal{E}\rvert)$	2				3			
	$\lvert\text{Sper}(\mathcal{E})\rvert$	$\lvert\mathcal{M}'_n\rvert$	$\lvert\mathcal{M}'\rvert$	$\lvert\mathcal{M}\rvert$	$\lvert\text{Sper}(\mathcal{E})\rvert$	$\lvert\mathcal{M}'_n\rvert$	$\lvert\mathcal{M}'\rvert$	$\lvert\mathcal{M}\rvert$
2	4	54	12	4	18	4970	234	33
3	4	234	24	3	18	443130	4752	179

As a special class of hyperedge multi-source network coding problems, it is easier to enumerate IDSC networks. Since we assume that all encoders in IDSC have access to all sources, we only need to consider the configurations at the decoders, which additionally are only afforded access to edges from intermediate nodes. These extra constraints are easily incorporated into Algorithms 8 and 5 by removing the edge definitions, restricting to the unique one associated with the IDSC problems, and enumerating exclusively the sink definitions.

We give the enumeration results for $K = 2, 3$ and $|\mathcal{E}_U| = 2, 3$ in Table 3.2, while the full list is available in [50]. From the table we see that, even for this special type of network, the number of non-isomorphic instances grows very quickly due to the fast growth of the number of Sperner families in the network size. For instance, the number of non-isomorphic IDSC instances grows from 33 to 179 (roughly, a factor of 6 increase), when the network size grows from $(2, 3)$ to $(3, 3)$.

3.5 Rate Region Results for Small Networks

After enumeration of non-isomorphic networks, one can generate a database of rate regions for networks. Here, we will show the database of calculated rate regions for networks with sizes $K \leq 3, |\mathcal{E}_U| \leq 3$.

3.5.1 Database of Rate Regions for all networks of size $K \leq 3, |\mathcal{E}_U| \leq 3$

We first show the experimental results on general hyperedge network instances. We investigated rate regions for more than $725,556$ non-isomorphic network instances, representing more than $8,619,064$ isomorphic ones. These include the cases when $K \leq 3$ and $|\mathcal{E}_U| \leq 3$, except the case when both $K = 3$ and $|\mathcal{E}_U| = 3$. For each non-isomorphic network instance, we calculated several bounds on its rate region: the Shannon outer bound $\mathcal{R}_o$, the scalar binary representable matroid inner bound $\mathcal{R}_{s,2}$, and the vector binary representable matroid inner bounds $\mathcal{R}_2^{N+1}$ and $\mathcal{R}_2^{N+2}$, where $N = K + |\mathcal{E}_U|$. If the outer bound on the rate region matches with an inner bound, we not only obtain the exact rate region, but also know the codes that suffice to achieve any point in it.

Table 3.3: Sufficiency of codes for network instances: Columns 3–7 show the number of instances that the rate region inner bounds match with the Shannon outer bound.

$(K, \lvert\mathcal{E}\rvert)$	$\lvert\mathcal{M}\rvert$	$\mathcal{R}_{s,2}(\mathsf{A})$	$\mathcal{R}_2^{N+1}(\mathsf{A})$	$\mathcal{R}_2^{N+2}(\mathsf{A})$	$\mathcal{R}_2^{N+3}(\mathsf{A})$	$\mathcal{R}_2^{N+4}(\mathsf{A})$
$(1,2)$	4	4	4	4	4	4
$(1,3)$	132	122	132	132	132	132
$(2,1)$	1	1	1	1	1	1
$(2,2)$	333	301	319	323	323	333
$(2,3)$	485890	341406	403883	432872	434545	–
$(3,1)$	9	4	4	9	9	9
$(3,2)$	239187	118133	168761	202130	211417	–

Though it is infeasible to list each of the more than $725,556$ rate regions in this chapter, a summary of results on the matches of various bounds is shown in Table 3.3. The full list of rate region bounds can be obtained at [51] and can be re-derived using [37].

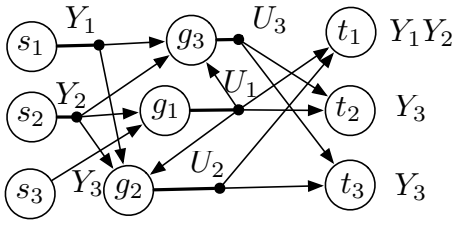
For the more than $725,556$ non-isomorphic network instances we considered, the Shannon outer bound is proved to be tight for most of them, excepting possibly some unknown instances where the rate region is not established using our techniques (i.e., the inner and outer bounds are not identical). For those remaining unknown, or unproven, instances, we obtain various inner bounds achievable by simple linear codes, e.g., binary or ternary codes. For the simple $(1,2)$ networks, scalar binary codes suffice. However, this is not true in general even when there are only two edge variables. For example, there are some instances in $(1,3)$ and $(2,2)$ networks for which scalar binary codes do not suffice. Even more instances of $(2,3)$ and $(3,3)$ networks need tighter inner bounds. As we can see from the table, as the vector binary inner bounds get tighter and tighter, the exact rate region can be proved for more and more instances. That is, with tighter and tighter binary inner bounds, more and more instances are found for which binary codes suffice.

We now use an example to show the tightness of various inner bounds on the rate region of a general $(3,3)$ problem. Equivalently, we have computer aided proofs for both achievability part and converse part.

Example 8: A 3-source 3-encoder hyperedge network instance A with block diagram and rate region $\mathcal{R}_*(\mathsf{A})$ shown in Fig. 3.9.

First, scalar binary codes do not suffice for this network. The scalar binary coding rate region is

$$\mathcal{R}_{s,2} = \mathcal{R}_*(\mathsf{A}) \cap \left\{ \begin{array}{l} R_1 + R_2 + R_3 \geq H(Y_1) + 2H(Y_2) + H(Y_3) \\ R_1 + R_2 + R_3 \geq H(Y_1) + H(Y_2) + 2H(Y_3) \end{array} \right\}. \tag{3.40}$$



$$
\begin{aligned}
R_1 &\geq H(Y_3) \\
R_2 &\geq H(Y_1) \\
R_1 + R_2 &\geq H(Y_1) + H(Y_2) + H(Y_3) \\
R_2 + R_3 &\geq H(Y_1) + H(Y_3) \\
R_1 + R_2 + 2R_3 &\geq H(Y_1) + H(Y_2) + 2H(Y_3)
\end{aligned}
$$

Figure 3.9: Block diagram and rate region $\mathcal{R}_*(\mathsf{A})$ for the $(3,3)$ network instance A in Example 8.

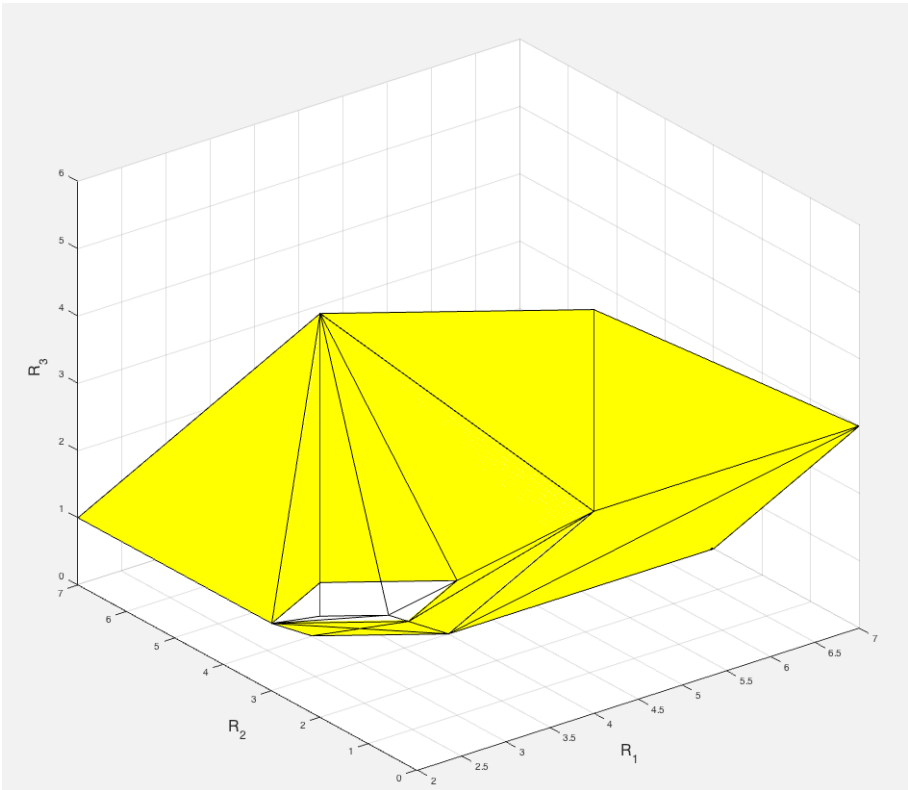


Figure 3.10: Comparison of rate regions $\mathcal{R}_*(\mathsf{A})$ (which equals to $\mathcal{R}_o(\mathsf{A})$) and $\mathcal{R}_2^7(\mathsf{A})$ for the $(3,3)$ network instance A in Example 8, when source entropies are $(H(Y_1), H(Y_2), H(Y_3)) = (1,2,1)$ and the cone is capped by $R_1 + R_2 + R + 3 \leq 10$: the white part is the portion that superposition coding cannot achieve. The ratio of $\mathcal{R}_2^7(\mathsf{A})$ over $\mathcal{R}_*(\mathsf{A})$ is about 99.57%.

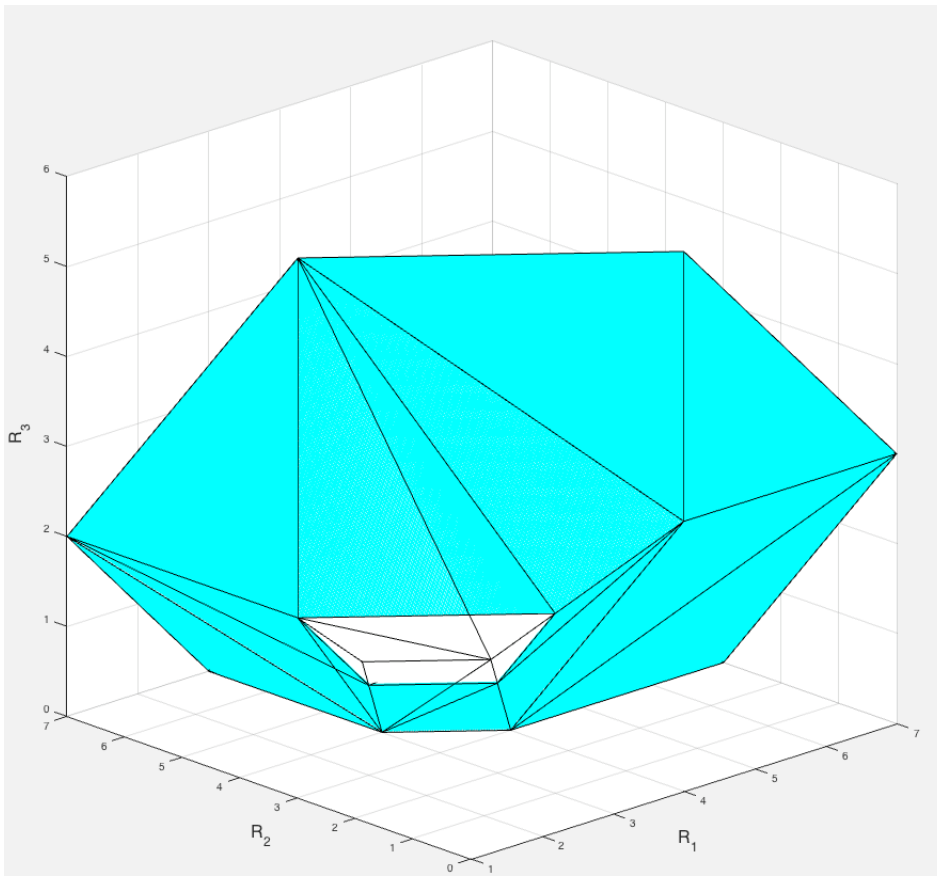


Figure 3.11: Comparison of rate regions $\mathcal{R}_2^7(\mathsf{A})$ and $\mathcal{R}_{s,2}(\mathsf{A})$ for the $(3,3)$ network instance A in Example 8 when source entropies are $(H(Y_1), H(Y_2), H(Y_3)) = (1,1,2)$ and the cone is capped by $R_1 + R_2 + R + 3 \leq 10$: the white part is the portion that scalar binary codes cannot achieve. The ratio of $\mathcal{R}_{s,2}(\mathsf{A})$ over $\mathcal{R}_2^7(\mathsf{A})$ is about 99.41%.

Table 3.4: Sufficiency of codes for IDSC instances: Columns 3 and 4 show the number of instances that the rate region inner bounds match with the Shannon outer bound.

$(K, \lvert\mathcal{E}\rvert)$	$\lvert\mathcal{M}\rvert$	$\mathcal{R}_{s,2}(\mathsf{A})$	$\mathcal{R}_2^{N+1}(\mathsf{A})$
$(2,2)$	4	4	4
$(2,3)$	33	26	33
$(3,2)$	3	3	3
$(3,3)$	179	143	179

One of the extreme rays in the Shannon outer bound on rate region is $(R_1, R_2, R_3, H(Y_1), H(Y_2), H(Y_3)) = (1, 1, 1, 0, 0, 2)$. This extreme ray cannot be achieved by scalar binary codes because no scalar code can encode a source with entropy of 2 into variables with entropy of 1. Fig. 3.10 illustrates the gap between $\mathcal{R}_*(\mathsf{A})$ and $\mathcal{R}_{s,2}(\mathsf{A})$ with a particular source entropy assignment. When source entropies are $(H(Y_1), H(Y_2), H(Y_3)) = (1, 2, 1)$ and the cone is capped by $R_1 + R_2 + R + 3 \leq 10$, there is a clear gap between the two polytopes, though the inner bound takes more than 99% space of the exact rate region.

Secondly, vector binary codes from 7 bits do not suffice for this network, either. The vector binary coding rate region is

$$\mathcal{R}_2^7 = \mathcal{R}_*(\mathsf{A}) \cap \{R_1 + R_2 + R_3 \geq H(Y_1) + 2H(Y_2) + H(Y_3)\}. \tag{3.41}$$

One of the extreme rays in the Shannon outer bound on rate region is $(R_1, R_2, R_3, H(Y_1), H(Y_2), H(Y_3)) = (2, 1, 1, 1, 2, 0)$. This extreme ray cannot be achieved by binary codes from 7 bits because 8 bits are necessary, including the empty source Y_3. Though this inner bound is still loose in the sense of matching with the exact rate region, it is tighter than the scalar binary inner bound $\mathcal{R}_{s,2}(\mathsf{A})$. This is illustrated in Fig. 3.11. When source entropies are $(H(Y_1), H(Y_2), H(Y_3)) = (1, 1, 2)$ and the cone is capped by $R_1 + R_2 + R + 3 \leq 10$, there is a clear gap between the two polytopes, though the scalar inner bound takes more than 99% space of the tighter vector binary inner bound.

However, vector binary codes from 8 bits suffice for this network and thus $\mathcal{R}_2^8(\mathsf{A}) = \mathcal{R}_*(\mathsf{A})$. One can construct vector binary codes to achieve all extreme rays in the Shannon outer bound on the rate region. For instance, the extreme ray $(R_1, R_2, R_3, H(X), H(Y), H(Z)) = (2, 1, 1, 1, 2, 0)$ can be achieved by the vector binary code as follows: $U_1 = [Y_1 + Y_2^{(2)},\ Y_2^{(1)}], U_2 = Y_2^{(1)} + Y_2^{(2)}, U_3 = Y_2^{(2)}$, where $Y_2^{(1)}, Y_2^{(2)}$ are the two bits in source Y_2.

3.5.2 Database of Rate Regions for small IDSC instances

Here, experimental results on thousands of IDSC instances are presented separately. We investigated rate regions for 219 non-isomorphic minimal IDSC instances representing 5130 isomorphic ones.

These include the cases when $(K, |\mathcal{E}|) = (2,2), (2,3), (3,2), (3,3)$. Similarly, for the rate region of each non-isomorphic IDSC instance, we calculated its Shannon outer bound $\mathcal{R}_o$, scalar binary inner bound $\mathcal{R}_{s,2}$, and the vector binary inner bounds $\mathcal{R}_2^{N+1}$, where $N = K + |\mathcal{E}|$.

A summary of results on the number of instances for which the various bounds agree is shown in Table 3.4. The exact rate regions, their converses, and the codes that achieve them for all 219 non-isomorphic cases can be obtained at [50] and can be re-derived using [37]. For the non-isomorphic IDSC instances we considered, the Shannon outer bound is always tight on the rate regions, and the exact rate regions are obtained. Scalar binary codes also only suffice for the instances with $|\mathcal{E}| = 2$ but not for all instances with $|\mathcal{E}| = 3$. However, vector binary codes from binary matroids on $N+1$ variables suffice for all the 219 instances. Thus, for the IDSC problems up to $K \leq 3, |\mathcal{E}| \leq 3$, vector binary codes suffice.

After obtaining a large database of rate regions for small networks, our next question is how to use them to solve more (larger) networks. For this purpose, we would like to define some operations to relate the networks of different sizes and show how to obtain rate regions form one another, and how the important properties, e.g., the tightness of outer/inner bounds, are preserved in the process.

3.6 Network Embedding Operations

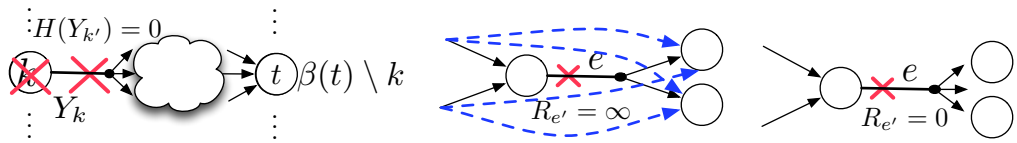
In this section, we propose a series of operations, including embedding and combination operations, relating smaller networks with larger networks in a manner such that one can obtain the rate region of the networks after operation directly. In addition, the insufficiency/sufficiency of a class of linear network codes is preserved. We first define some embedding operations on a larger network so that the rate region of the *embedded* network can be easily derived from the original larger network. Furthermore, a class of linear codes will be sufficient for the embedded networks (after operation), if the codes are sufficient for the larger network. We give the definitions first.

3.6.1 Definition of embedding operations

The first operation is source deletion. When a source is deleted, the decoders that demand it will no longer demand it after deletion. In addition, the nodes and edges that only depend on this source will also be deleted.

Definition 14 (Source Deletion $(\mathsf{A}\backslash k)$)**:** Suppose a network $\mathsf{A} = (\mathcal{S}, \mathcal{G}, \mathcal{T}, \mathcal{E}, (\beta(t), t \in \mathcal{T}))$. When a source $k \in \mathcal{S}$ is deleted, in the new network $\mathsf{A}' = (\mathcal{S}' = \mathcal{S} \setminus k, \mathcal{G}', \mathcal{T}', \mathcal{E}', (\beta'(t), t \in \mathcal{T}'))$, we will have:

1. $\mathcal{G}' = \mathcal{G} \setminus \{i | \text{no path between } i, j, \forall j \in \mathcal{S} \setminus k\}$;



(a) Source deletion: when source k is deleted, it sends nothing to the network. Decoders that previously required k will no longer require it.

(b) Edge contraction: when e is contracted, the head nodes will directly have access to all input of $\mathrm{Tl}(e)$.

(c) Edge deletion: when e is deleted, its head nodes will no longer receive information from e.

Figure 3.12: Definitions of embedding operations on a network

2. $\mathcal{T}' = \mathcal{T} \setminus \{t | \exists t' \in \mathcal{T}, t' \neq t, \beta(t) = \beta(t') \setminus k, \mathrm{In}_{\mathsf{A}}(t') \subseteq \mathrm{In}_{\mathsf{A}}(t)\} \setminus \{t | \mathrm{Tl}(\mathrm{In}_{\mathsf{A}}(t)) \cap \mathcal{G}' = \emptyset$ or $\beta(t) = k\}$;

3. $\mathcal{E}' = \{(\mathrm{Tl}(e) \cap \{\mathcal{S}' \cup \mathcal{G}'\}, \mathrm{Hd}(e) \cap \{\mathcal{G}' \cup \mathcal{T}'\}), \forall e \in \mathcal{E}\}$.

4. $\forall t \in \mathcal{T}'$, $\beta'(t) = \beta(t) \setminus k$;

Fig. 3.12(a) demonstrates the deletion of a source. When source k is deleted, t will no longer require k. The other edges and nodes that only depend on k, if any, will also be deleted. A particular example is shown in Fig. 3.13(a).

Next, we consider the operation of contracting an edge. When an edge is contracted, the head node will directly have access to all the inputs of the tail node. This yields the same effective network as if the contracted edge had been replaced with a super edge with infinite capacity so that all the inputs of tail node could be directly sent to the head node.

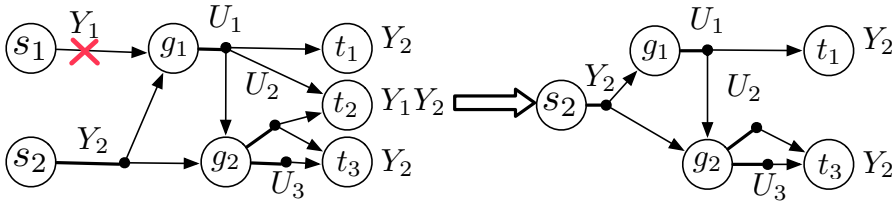
Definition 15 (Edge Contraction (A/e))**:** Suppose a network $\mathsf{A} = (\mathcal{S}, \mathcal{G}, \mathcal{T}, \mathcal{E}, (\beta(t), t \in \mathcal{T}))$. A smaller network $\mathsf{A}' = (\mathcal{S}, \mathcal{G}, \mathcal{T}, \mathcal{E}', (\beta(t), t \in \mathcal{T}))$ is obtained by contracting $e, e \in \mathcal{E}$, denoted by A/e, if $\mathcal{E}' = (\mathcal{E} \setminus \{e\}) \cup \{e' = (\mathrm{Tl}(e), \mathrm{Hd}(e)) | R_{e'} = \infty\}$.

Fig. 3.12(b) demonstrates the contraction of an edge. As it shows, when edge e is contracted, the two nodes it connects will be connected with infinity capacity so that the head node of e will directly have all the input of tail node of e. A particular example is shown in Fig. 3.13(b).

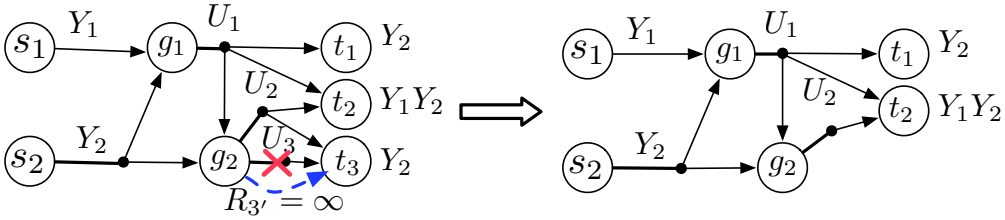
Next, we define edge deletion.

Definition 16 (Edge Deletion ($\mathsf{A} \backslash e$))**:** Suppose a network $\mathsf{A} = (\mathcal{S}, \mathcal{G}, \mathcal{T}, \mathcal{E}, (\beta(t), t \in \mathcal{T}))$. A smaller network instance $\mathsf{A}' = (\mathcal{S}, \mathcal{G}, \mathcal{T}, \mathcal{E}', (\beta(t), t \in \mathcal{T}))$ is obtained by deleting $e, e \in \mathcal{E}$, denoted by $\mathsf{A} \backslash e$, if $\mathcal{E}' = \mathcal{E} \setminus e$.

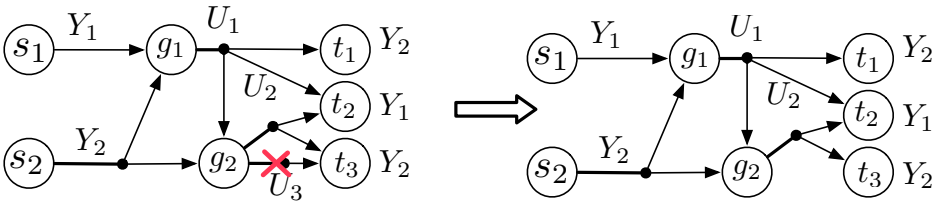
The essence of this definition is to keep the dependence relationship between input and output of its head node when an edge is deleted. In other words, when an edge is deleted, the head node should function as before deletion. Fig. 3.12(c) demonstrates the deletion of an edge. When edge e is deleted, head node of e no longer has access to U_e. A particular example is shown in Fig. 3.13(c).



(a) Source deletion example: when Y_1 is deleted, the network is not available to it and thus t_2 will only demand Y_2, which has been implied by t_1. Thus, t_2 is removed as well.



(b) Demonstration of edge contraction on a network: when e_3 is contracted, input of g_2 will be directly available to t_3, which indicates that the demand of t_3 will be trivially satisfied and thus t_3 is removed.



(c) Demonstration of edge deletion on a network: when e_3 is deleted, t_3 has no access to U_3, it can only decode Y_2 from U_2.

Figure 3.13: Examples to show the embedding operations on a network

Remark 1: After the embedding operations on a network, redundancies may be introduced so that the network should be further reduced to its minimal representation by actions introduced in §3.3.

Next, we consider the order of different operations. It is not difficult to see that if a collection of sources are deleted, it does not matter which source is deleted first. Similarly, if a collection of edges are contracted or deleted, it does not matter which edge is operated first. Next we would like to show that different orders of operations give equivalent results.

Theorem 12: Let $\mathcal{O}_1, \mathcal{O}_2$ be two different operations on different elements, among the three operations defined in Definition 14– Definition 16. Applying $\mathcal{O}_1$ firstly and $\mathcal{O}_2$ secondly is equivalent to apply $\mathcal{O}_2$ firstly and $\mathcal{O}_1$ secondly.

Proof: We need to consider the 3 combinations.

Edge deletion and edge contraction: we need to show $(\mathsf{A}\backslash e_1)/e_2 = (\mathsf{A}/e_2)\backslash e_1$. We only need to consider the case when $e_1 \in \mathrm{In}(\mathrm{Tl}(e_2))$. If e_1 is deleted at first, when e_2 is contracted, the head node of e_2 will have access to tails of all the other edges that go into tail of e_2 except e_1. If e_2 is contracted first, e_1 will still not be available for head of e_2 since it is deleted.

Edge deletion and source deletion: we need to show $(\mathsf{A}\backslash e_1)\backslash s = (\mathsf{A}\backslash s)\backslash e_1$. We need to consider the case when e_1 only depends on s. In this case, no matter which operation is done first, e_1 and all other edges (nodes) that only depend on s will be deleted.

Edge contraction and source deletion: we need to show $(\mathsf{A}/e_1)\backslash s = (\mathsf{A}\backslash s)/e_1$. We need to consider the case that e_1 only depends on s. In this case, $\mathrm{In}(\mathrm{Tl}(e_1))$ will also only depend on s. No matter which operation is done first, e_1, together with $\mathrm{In}(\mathrm{Tl}(e_1))$ will be deleted. The remaining network is the same. ■

Based on these operations and Theorem 12, we can define an *embedded* network.

Definition 17 (Embedded Network)**:** A network A' is said *embedded* in another network A or a *minor* of A, denoted as $\mathsf{A}' \prec \mathsf{A}$, if A' can be obtained by a series of operations of source deletion, edge deletion/ contraction on A. Reversely, we say that A is an extension of A', $\mathsf{A} \succ \mathsf{A}'$.

3.6.2 Preservation properties of embedding operations

Here we show that the property of the insufficiency of a class of linear network coding is inherited in the larger network from the smaller network under the embedding operations presented in the previous section.

Theorem 13: Suppose a network $\mathsf{A}' = (\mathcal{S} \setminus k, \mathcal{G}', \mathcal{T}', \mathcal{E}', (\beta'(t), t \in \mathcal{T}'))$ is obtained by deleting k from

another network $\mathsf{A} = (\mathcal{S}, \mathcal{G}, \mathcal{T}, \mathcal{E}, (\beta(t), t \in \mathcal{T}))$, then

$$\mathcal{R}(\mathsf{A}') = \mathrm{Proj}_{\mathbf{X}_{\setminus k}, \mathbf{R}_{\mathcal{E}'}} \left(\{ \mathbf{R} \in \mathcal{R}(\mathsf{A}) \,|H(X_k) = 0 \} \right), \tag{3.42}$$

$$\mathcal{R}_q(\mathsf{A}') = \mathrm{Proj}_{\mathbf{X}_{\setminus k}, \mathbf{R}_{\mathcal{E}'}} \left(\{ \mathbf{R} \in \mathcal{R}_q(\mathsf{A}) \,|H(X_k) = 0 \} \right), \tag{3.43}$$

$$\mathcal{R}_{s,q}(\mathsf{A}') = \mathrm{Proj}_{\mathbf{X}_{\setminus k}, \mathbf{R}_{\mathcal{E}'}} \left(\{ \mathbf{R} \in \mathcal{R}_{s,q}(\mathsf{A}) \,|H(X_k) = 0 \} \right). \tag{3.44}$$

Proof: Select any point $\mathbf{R}' \in \mathcal{R}(\mathsf{A}')$, then there exists a sequence of random variables $\{\mathbf{X}_{\setminus k}^{(j)}, U_i^{(j)}, i \in \mathcal{E}'\}$ such that their entropies satisfy all the constraints determined by A', as $j \to \infty$. Define $X_k^{(j)}$ to be the empty sources, $H(X_k^{(j)}) = 0$. Then the entropies of random variables $\{\mathbf{X}_{\setminus k}^{(j)}, U_i^{(j)}, i \in \mathcal{E}'\} \cup X_k^{(j)}$ will satisfy the constraints in A with $H(X_k^{(j)}) = 0$. Hence, as $j \to \infty$, the associated rate point $\mathbf{R} \in \{\mathbf{R} \in \mathcal{R}(\mathsf{A}) \,|H(X_k) = 0\}$. Thus, we have $\mathcal{R}(\mathsf{A}') \subseteq \mathrm{Proj}_{\mathbf{X}_{\setminus k}, \mathbf{R}_{\mathcal{E}'}}(\{\mathbf{R} \in \mathcal{R}(\mathsf{A}) | H(X_k) = 0\})$. If $\mathbf{R}'$ is achievable by $\mathbb{F}_q$ codes, since letting X_k does not affect the other sources and codes, the same $\mathbb{F}_q$ code will also achieve the point $\mathbf{R}$ with $H(X_k) = 0$. Thus, $\mathcal{R}_q(\mathsf{A}') \subseteq \mathrm{Proj}_{\mathbf{X}_{\setminus s}, \mathbf{R}_{\mathcal{E}'}}(\{\mathbf{R} \in \mathcal{R}_q(\mathsf{A}) | H(X_k) = 0\})$ follows.

On the other hand, if we select any point $\mathbf{R} \in \{\mathbf{R} \in \mathcal{R}(\mathsf{A}) | H(X_k) = 0\}$, we can see that $\mathbf{R}' = \mathrm{Proj}_{\mathbf{X}_{\setminus k}, \mathbf{R}_{\mathcal{E}'}}(\mathbf{R}) \in \mathcal{R}(\mathsf{A}')$ because $\mathbf{R}'$ the entropies of $\{\mathbf{X}_{\setminus k}, U_i, i \in \mathcal{E}'\}$ satisfy all constraints determined by A' and the entropic vector projecting out X_k is still entropic. Thus, we have $\mathrm{Proj}_{\mathbf{X}_{\setminus k}, \mathbf{R}_{\mathcal{E}'}}(\{\mathbf{R} \in \mathcal{R}(\mathsf{A}) | H(X_k) = 0\}) \subseteq \mathcal{R}(\mathsf{A}')$. If $\mathbf{R}$ is achievable by $\mathbb{F}_q$ code $\mathbb{C}$, then the code to achieve $\mathbf{R}'$ could be the code $\mathbb{C}$ with deletion of rows associated with source X_k, i.e., $\mathbb{C}' = \mathbb{C}_{\setminus X_k, :}$. Thus, $\mathrm{Proj}_{\mathbf{X}_{\setminus k}, \mathbf{R}_{\mathcal{E}'}}(\{\mathbf{R} \in \mathcal{R}_q(\mathsf{A}) | H(X_k) = 0\}) \subseteq \mathcal{R}_q(\mathsf{A}')$. ■

Theorem 14: Suppose a network $\mathsf{A}' = (\mathcal{S}, \mathcal{G}', \mathcal{T}, \mathcal{E}', (\beta(t), t \in \mathcal{T}))$ is obtained by contracting e from another network $\mathsf{A} = (\mathcal{S}, \mathcal{G}, \mathcal{T}, \mathcal{E}, (\beta(t), t \in \mathcal{T}))$, i.e., $\mathsf{A}' = \mathsf{A} \setminus e$, then

$$\mathcal{R}(\mathsf{A}') = \mathrm{Proj}_{H(X_k), k \in \mathcal{S}, \mathbf{R}_{\mathcal{E}'}} \mathcal{R}(\mathsf{A}), \tag{3.45}$$

$$\mathcal{R}_q(\mathsf{A}') = \mathrm{Proj}_{H(X_k), k \in \mathcal{S}, \mathbf{R}_{\mathcal{E}'}} \mathcal{R}_q(\mathsf{A}), \tag{3.46}$$

$$\mathcal{R}_{s,q}(\mathsf{A}') \supseteq \mathrm{Proj}_{H(X_k), k \in \mathcal{S}, \mathbf{R}_{\mathcal{E}'}} \mathcal{R}_{s,q}(\mathsf{A}), \tag{3.47}$$

Proof: Select any point $\mathbf{R}' \in \mathcal{R}(\mathsf{A}')$, then there exists a sequence of random variables $\{\mathbf{X}_{\mathcal{S}}^{(j)}, U_i^{(j)}, i \in \mathcal{E}'\}$ such that their entropies satisfy all the constraints determined by A', as $j \to \infty$. In the network A, define $U_e^{(j)}$ to be the concatenation of all input of tail node of e, $U_e^{(j)} = \mathbf{U}_{\mathrm{In}(\mathrm{Tl}(e))}^{(j)}$. Then the entropies of random variables $\{\mathbf{X}_{\mathcal{S}}^{(j)}, U_i^{(j)}, i \in \mathcal{E}'\} \cup U_e^{(j)}$ will satisfy the constraints in A, and additionally obey $H(U_e^{(j)}) = H(\mathbf{U}_{\mathrm{In}(\mathrm{Tl}(e))}^{(j)})$. Hence, as $j \to \infty$, the associated rate point $\mathbf{R} \in \left\{\mathbf{R} \in \mathcal{R}(\mathsf{A}) \,\middle|\, H(U_e) \geq H(\mathbf{U}_{\mathrm{In}(\mathrm{Tl}(e))}) \right\}$. Thus, we have

$$\begin{aligned}
&\mathcal{R}(\mathsf{A}') \subseteq \\
&\mathrm{Proj}_{H(X_{\mathcal{S}}),\mathbf{R}_{\mathcal{E}'}}(\{\mathbf{R} \in \mathcal{R}(\mathsf{A}) | H(U_e) \geq H(\mathbf{U}_{\mathrm{In}(\mathrm{Tl}(e))})\}) \\
&\subseteq \mathrm{Proj}_{H(X_{\mathcal{S}}),\mathbf{R}_{\mathcal{E}'}} \mathcal{R}(\mathsf{A}).
\end{aligned} \tag{3.48}$$

If $\mathbf{R}'$ is achievable by general $\mathbb{F}_q$ codes, since concatenation of all sources is a valid $\mathbb{F}_q$ code, we have

$$\begin{aligned}
&\mathcal{R}_q(\mathsf{A}') \subseteq \\
&\mathrm{Proj}_{H(X_{\mathcal{S}}),\mathbf{R}_{\mathcal{E}'}}(\{\mathbf{R} \in \mathcal{R}_q(\mathsf{A}) | H(U_e) \geq H(\mathbf{U}_{\mathrm{In}(\mathrm{Tl}(e))})\}) \\
&\subseteq \mathrm{Proj}_{H(X_{\mathcal{S}},\mathbf{R}_{\mathcal{E}'}} \mathcal{R}_q(\mathsf{A}).
\end{aligned} \tag{3.49}$$

However, we cannot establish same relationship when scalar $\mathbb{F}_q$ codes are considered, because for the point $\mathbf{R}'$, the associated $\mathbf{R}$ with $H(U_e)$ may not be scalar $\mathbb{F}_q$ achievable.

On the other hand, if we select any point $\mathbf{R} \in \{\mathbf{R} \in \mathcal{R}(\mathsf{A}) | H(X_k) = 0\}$, we can see that $\mathbf{R}' = \mathrm{Proj}_{\mathbf{X}_{\setminus k},\mathbf{R}_{\mathcal{E}'}}(\mathbf{R}) \in \mathcal{R}(\mathsf{A}')$ because the entropies of $\{\mathbf{X}, U_i, i \in \mathcal{E}'\}$ satisfy all constraints determined by A' (since they are a subset of the constraints from A) and the entropic vector projecting out U_e is still entropic. Thus, we have

$$\mathrm{Proj}_{H(X_{\mathcal{S}}),\mathbf{R}_{\mathcal{E}'}} \mathcal{R}(\mathsf{A}) \subseteq \mathcal{R}(\mathsf{A}'). \tag{3.50}$$

If $\mathbf{R} \in \mathcal{R}(\mathsf{A})$ is achievable by $\mathbb{F}_q$ code $\mathbb{C}$, either scalar or vector, then the code to achieve $\mathbf{R}' = \mathrm{Proj}_{H(X_{\mathcal{S}}),\mathbf{R}_{\mathcal{E}'}} \mathbf{R} \in \mathcal{R}(\mathsf{A}')$ could be the code $\mathbb{C}$ with deletion of columns associated with encoder E_e, i.e., $\mathbb{C}' = \mathbb{C}_{:,\setminus E_e}$. Thus, we have

$$\mathrm{Proj}_{H(X_{\mathcal{S}}),\mathbf{R}_{\mathcal{E}'}} \mathcal{R}_q(\mathsf{A}) \subseteq \mathcal{R}_q(\mathsf{A}'), \tag{3.51}$$

$$\mathrm{Proj}_{H(X_{\mathcal{S}}),\mathbf{R}_{\mathcal{E}'}} \mathcal{R}_{s,q}(\mathsf{A}) \subseteq \mathcal{R}_{s,q}(\mathsf{A}'). \tag{3.52}$$

■

Theorem 15: Suppose a network instance $\mathsf{A}' = (\mathcal{S}, \mathcal{G}, \mathcal{T}, \mathcal{E}', (\beta(t), t \in \mathcal{T}))$ is obtained by deleting e from another network instance $\mathsf{A} = (\mathcal{S}, \mathcal{G}, \mathcal{T}, \mathcal{E}, (\beta(t), t \in \mathcal{T}))$, then

$$\mathcal{R}(\mathsf{A}') = \mathrm{Proj}_{H(X_{\mathcal{S}}),\mathbf{R}_{\mathcal{E}'}}(\{\mathbf{R} \in \mathcal{R}(\mathsf{A}) | R_e = 0\}), \tag{3.53}$$

$$\mathcal{R}_q(\mathsf{A}') = \mathrm{Proj}_{H(X_{\mathcal{S}}),\mathbf{R}_{\mathcal{E}'}}(\{\mathbf{R} \in \mathcal{R}_q(\mathsf{A}) | R_e = 0\}), \tag{3.54}$$

$$\mathcal{R}_{s,q}(\mathsf{A}') = \mathrm{Proj}_{H(X_{\mathcal{S}}),\mathbf{R}_{\mathcal{E}'}}(\{\mathbf{R} \in \mathcal{R}_{s,q}(\mathsf{A}) | R_e = 0\}). \tag{3.55}$$

Proof: Select any point $\mathbf{R}' \in \mathcal{R}(\mathsf{A}')$, then there exists a sequence of random variables $\{\mathbf{X}_{\mathcal{S}}^{(j)}, U_i^{(j)}, i \in \mathcal{E}'\}$ such that their entropies satisfy all the constraints determined by A', as $j \to \infty$. Let $U_e^{(j)}$ be empty set or encode all input with the all-zero vector, $U_e^{(j)} = \emptyset$. Then the entropies of random variables $\{\mathbf{X}_{\mathcal{S}}^{(j)}, U_i^{(j)}, i \in \mathcal{E}'\} \cup U_e^{(j)}$ will satisfy the constraints in A, and additionally obey $H(U_e^{(j)}) \leq R_e = 0$, as $j \to \infty$. Hence, the associated rate point $\mathbf{R} \in \mathcal{R}(\mathsf{A}) \cap \{\mathbf{R} \in \mathcal{R}(\mathsf{A}) \,|R_e = 0\}$. Thus, we have

$$\mathcal{R}(\mathsf{A}') \subseteq \mathrm{Proj}_{H(X_{\mathcal{S}}),\mathbf{R}_{\mathcal{E}'}}(\{\mathbf{R} \in \mathcal{R}(\mathsf{A})|R_e = 0\}). \tag{3.56}$$

If $\mathbf{R}'$ is achievable by general $\mathbb{F}_q$ linear vector or scalar codes, since all-zero code is a valid $\mathbb{F}_q$ vector and linear code, we have

$$\mathcal{R}_q(\mathsf{A}') \subseteq \mathrm{Proj}_{H(X_{\mathcal{S}}),\mathbf{R}_{\mathcal{E}'}}(\{\mathbf{R} \in \mathcal{R}_q(\mathsf{A})|R_e = 0\}), \tag{3.57}$$

$$\mathcal{R}_{s,q}(\mathsf{A}') \subseteq \mathrm{Proj}_{H(X_{\mathcal{S}}),\mathbf{R}_{\mathcal{E}'}}(\{\mathbf{R} \in \mathcal{R}_{s,q}(\mathsf{A})|R_e = 0\}). \tag{3.58}$$

On the other hand, if we select any point $\mathbf{R} \in \{\mathbf{R} \in \mathcal{R}(\mathsf{A})|R_e = 0\}$, we can see that $\mathbf{R}' = \mathrm{Proj}_{H(X_{\mathcal{S}}),\mathbf{R}_{\mathcal{E}'}}(\mathbf{R}) \in \mathcal{R}(\mathsf{A}')$ because the entropies of $\{\mathbf{X}, U_i, i \in \mathcal{E}'\}$ satisfy all constraints determined by A' (since the picked point $\mathbf{R}$ has the property of $R_e = 0$) and the entropic vector projecting out U_e is still entropic. Thus, we have

$$\mathrm{Proj}_{H(X_{\mathcal{S}}),\mathbf{R}_{\mathcal{E}'}}(\{\mathbf{R} \in \mathcal{R}_q(\mathsf{A})|R_e = 0\}) \subseteq \mathcal{R}(\mathsf{A}'). \tag{3.59}$$

If $\mathbf{R}$ is achievable by $\mathbb{F}_q$ code $\mathbb{C}$, no matter $\mathbb{C}$ is vector or scalar code, then the code to achieve $\mathbf{R}'$ could be the code $\mathbb{C}$ with deletion of columns associated with U_e, i.e., $\mathbb{C}' = \mathbb{C}_{:,\setminus U_e}$, because U_e is sending nothing. Thus, we have

$$\mathrm{Proj}_{H(X_{\mathcal{S}}),\mathbf{R}_{\mathcal{E}'}}(\{\mathbf{R} \in \mathcal{R}_q(\mathsf{A})|R_e = 0\}) \subseteq \mathcal{R}_q(\mathsf{A}'), \tag{3.60}$$

$$\mathrm{Proj}_{H(X_{\mathcal{S}}),\mathbf{R}_{\mathcal{E}'}}(\{\mathbf{R} \in \mathcal{R}_q(\mathsf{A})|R_e = 0\}) \subseteq \mathcal{R}_{s,q}(\mathsf{A}'). \tag{3.61}$$

■

Corollary 5: Given two networks A, A' such that $\mathsf{A}' \prec \mathsf{A}$. If $\mathbb{F}_q$ linear codes suffice for A, then $\mathbb{F}_q$ linear codes suffice for A'. Equivalently, if $\mathbb{F}_q$ linear codes do not suffice for A', then $\mathbb{F}_q$ linear codes do not suffice for A. Equivalently, if $\mathcal{R}_q(\mathsf{A}) = \mathcal{R}(\mathcal{A})$, then $\mathcal{R}_q(\mathcal{A}') = \mathcal{R}(\mathcal{A}')$.

Proof: From Definition 17 we know that A' is obtained by a series of operations of source deletion, edge deletion, edge contraction. Theorem 12 indicates that the order of the operations

does not matter. Thus, it suffices to show that the statement holds when A' can be obtained by one of the operations on A.

Suppose $\mathbb{F}_q$ linear codes suffice to achieve every point in $\mathcal{R}(\mathsf{A})$, i.e., $\mathcal{R}_q(\mathsf{A}) = \mathcal{R}(\mathsf{A})$. If A' is obtained by contracting one edge in A, (3.45) and (3.46) in Theorem 14 indicate $\mathcal{R}_q(\mathsf{A}') = \mathcal{R}(\mathsf{A})$. Similarly, same conclusion is obtained from (3.42) and (3.43) in Theorem 13, (3.53) and (3.54) in Theorem 15, when A' is obtained by source deletion or deletion deletion. ■

Note that scalar codes are a spacial class of general linear codes. If we only consider scalar linear codes, we will have the following similar corollary.

Corollary 6: Given two network instances A, A' such that $\mathsf{A}' \prec \mathsf{A}$. If $\mathbb{F}_q$ scalar linear codes suffice for A, then $\mathbb{F}_q$ scalar linear codes suffice for A'. Equivalently, if $\mathbb{F}_q$ scalar linear codes do not suffice for A', then $\mathbb{F}_q$ scalar linear codes do not suffice for A. Equivalently, if $\mathcal{R}_{s,q}(\mathsf{A}) = \mathcal{R}(\mathcal{A})$, then $\mathcal{R}_{s,q}(\mathcal{A}') = \mathcal{R}(\mathcal{A}')$.

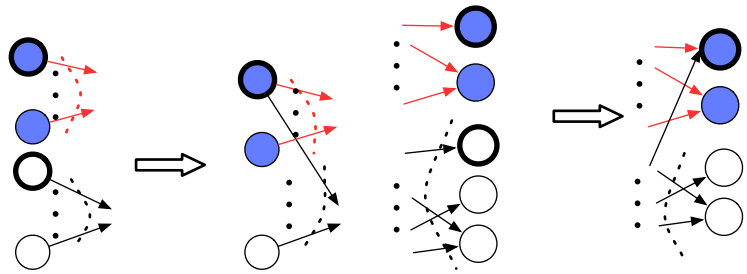
Proof: It follows a similar proof for Corollary 5. If $\mathcal{R}_{s,q}(\mathsf{A}) = \mathcal{R}(\mathcal{A})$, we can get $\mathcal{R}_{s,q}(\mathcal{A}') = \mathcal{R}(\mathcal{A}')$ from (3.53) and (3.55) in Theorem 15 or (3.42) and (3.44) in Theorem 13 if A' is obtained by deleting an encoder or source from A.

Now we consider the case that A' is obtained by contracting an edge from A. If $\mathcal{R}_{s,q}(\mathsf{A}) = \mathcal{R}(\mathcal{A})$, the projections $\mathrm{Proj}_{H(X_{\mathcal{S}}),\mathbf{R}_{\mathcal{E}'}}\mathcal{R}_{s,q}(\mathsf{A}) = \mathrm{Proj}_{H(X_{\mathcal{S}}),\mathbf{R}_{\mathcal{E}'}}\mathcal{R}(\mathcal{A})$. Together with (3.45) and (3.47), we will have $\mathcal{R}_{s,q}(\mathcal{A}') \supseteq \mathcal{R}(\mathcal{A}')$. It is trivial that $\mathcal{R}_{s,q}(\mathcal{A}') \subseteq \mathcal{R}(\mathcal{A}')$ because $\Gamma_N^q \subseteq \bar{\Gamma}_N^*$. Thus, we have $\mathcal{R}_{s,q}(\mathcal{A}') = \mathcal{R}(\mathcal{A}')$. ■

After introducing the embedding operations which give smaller networks from larger networks, we would like to introduce some combination operations to get larger networks from smaller ones.

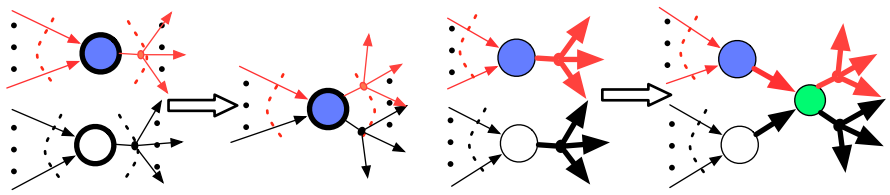
3.7 Network Combination Operations

In this section, we propose a series of combination operations relating smaller networks with larger networks in a manner such that the rate region of the larger network can be easily derived from those of the smaller ones. In addition, the sufficiency of a class of linear network codes is inherited in the larger network from the smaller one. Throughout the following, the network $\mathsf{A} = (\mathcal{S}, \mathcal{G}, \mathcal{T}, \mathcal{E}, (\beta(t), t \in \mathcal{T}))$ is a combination of two *disjoint* networks $\mathsf{A}_i = (\mathcal{S}_i, \mathcal{G}_i, \mathcal{T}_i, \mathcal{E}_i, (\beta_i(t), t \in \mathcal{T}_i))$, $i \in \{1, 2\}$, meaning $\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$, $\mathcal{G}_1 \cap \mathcal{G}_2 = \emptyset$, $\mathcal{T}_1 \cap \mathcal{T}_2 = \emptyset$, $\mathcal{E}_1 \cap \mathcal{E}_2 = \emptyset$, and $\beta_1(t_1) \cap \beta_2(t_2) = \emptyset, \forall t_1 \in \mathcal{T}_1, t_2 \in \mathcal{T}_2$.



(a) Sources merge: the merged source will serve for the new larger network.

(b) Sinks merge: input and output of the sinks are unioned, respectively.



(c) Intermediate nodes merge: input and output of the nodes are unioned, respectively.

(d) Edges merge: one extra node and four associated edges are added to replace the two edges.

Figure 3.14: Combination operations on two smaller networks to form a larger network. Thickly lined nodes (edges) are merged.

3.7.1 Definition of Combination Operations

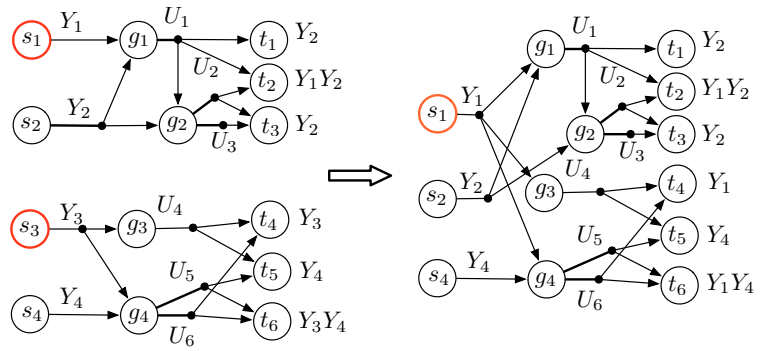
The operations we will define will merge network elements, e.g., sources, intermediate nodes, sink nodes, edges, etc, and are depicted in Fig. 3.14. Since each merge will combine one or several pairs of elements, with each pair containing one element from A_1 and the other from A_2, each merge definition will involve a bijection π indicating which element from the appropriate set of A_2 is paired with its argument in A_1.

We first consider the sources merge operation, in which the merged sources will function as identical sources for both sub-networks, as shown in Fig. 3.14(a).

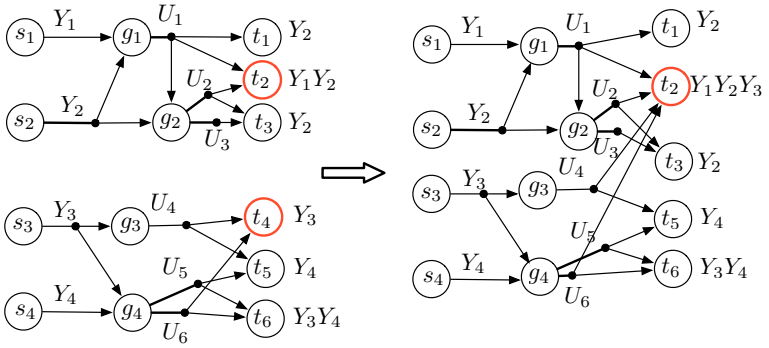
Definition 18 (Source Merge ($\mathsf{A}_1.\hat{\mathcal{S}} = \mathsf{A}_2.\pi(\hat{\mathcal{S}})$) – Fig. 3.14(a))**:** The result A of merging the sources $\hat{\mathcal{S}} \subseteq \mathcal{S}_1$ from network A_1 with the sources $\pi(\hat{\mathcal{S}}) \subseteq \mathcal{S}_2$ from a disjoint network A_2, will have: $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2 \setminus \pi(\hat{\mathcal{S}})$; $\mathcal{G} = \mathcal{G}_1 \cup \mathcal{G}_2$; $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$; $\mathcal{E} = (\mathcal{E}_1 \cup \mathcal{E}_2 \setminus \{e \in \mathcal{E}_1 \cup \mathcal{E}_2 | \mathsf{Tl}(e) \in \hat{\mathcal{S}} \cup \pi(\hat{\mathcal{S}})\}) \cup \{(s, \mathcal{F}_1 \cup \mathcal{F}_2) | s \in \hat{\mathcal{S}}, (s, \mathcal{F}_1) \in \mathcal{E}_1, (\pi(s), \mathcal{F}_2) \in \mathcal{E}_2\}$; and

$$\beta(t) = \begin{cases} \beta_1(t) & t \in \mathcal{T}_1 \\ \left(\beta_2(t) \setminus \pi(\hat{S})\right) \cup \pi^{-1}\left(\pi(\hat{S}) \cap \beta_2(t)\right) & t \in \mathcal{T}_2 \end{cases}.$$

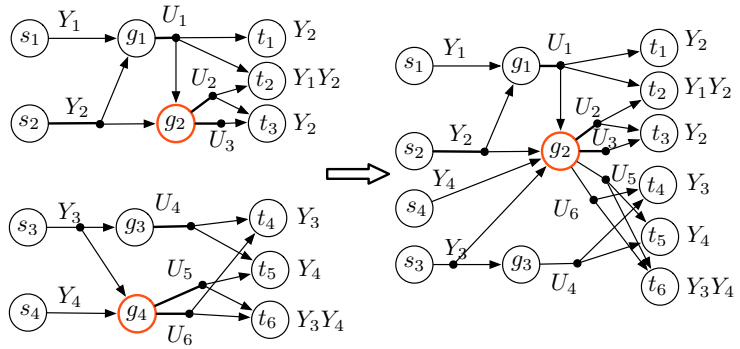
To clearly demonstrate the source merge, a particular example is given in Fig. 3.15(a).



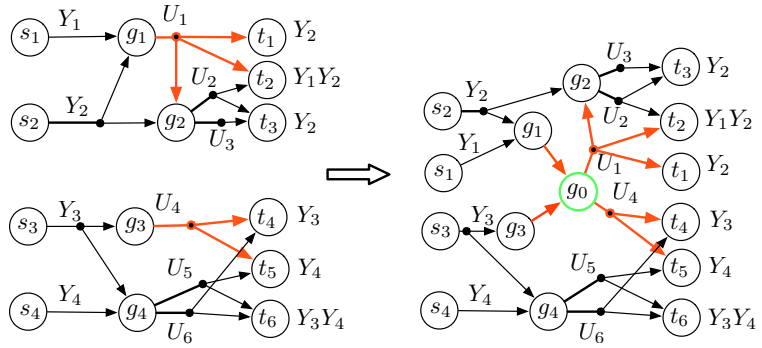
(a) Demonstration of source merge on two networks: s_1, s_2 are merged to s_1, so s_1 will send information to both sub-networks.



(b) Demonstration of sink merge on two networks: sink t_2, t_4 are merged, so input demands are combined.



(c) Demonstration of node merge on two networks: node g_2 and g_4 are merged at node g_2, their input and output are also combined.



(d) Demonstration of edge merge on two networks: when U_1, U_4 get merged, one extra node and four edges are added to replace U_1, U_4 in the two networks, respectively.

Figure 3.15: Example to demonstrate combinations of two networks.

Definition 19 (Sink Merge ($\mathsf{A}_1.\hat{\mathcal{T}} + \mathsf{A}_2.\pi(\hat{\mathcal{T}})$) – Fig. 3.14(b).)**:** The result A of merging the sinks $\hat{\mathcal{T}} \subseteq \mathcal{T}_1$ from network A_1 with the sinks $\pi(\hat{\mathcal{T}}) \subseteq \mathcal{T}_2$ from the disjoint network A_2 will be: $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2$; $\mathcal{G} = \mathcal{G}_1 \cup \mathcal{G}_2$; $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2 \setminus \pi(\hat{\mathcal{T}})$; $\mathcal{E} = \mathcal{E}_1 \cup \mathcal{E}_2 \cup \{(g_2, \mathcal{F}_1 \cup \mathcal{F}_2) | g_2 \in \mathcal{G}_2, \mathcal{F}_1 \subseteq \hat{\mathcal{T}}, \mathcal{F}_2 \subseteq \mathcal{T}_2, (g_2, \pi(\mathcal{F}_1) \cup \mathcal{F}_2) \in \mathcal{E}_2\} \setminus \{(g_2, \mathcal{F}_2) \in \mathcal{E}_2 | \mathcal{F}_2 \cap \pi(\hat{\mathcal{T}}) \neq \emptyset\}$; and

$$\beta(t) = \begin{cases} \beta_i(t) & t \in \mathcal{T}_i \setminus \hat{\mathcal{T}}, i \in \{1,2\} \\ \beta_1(t) \cup \beta_2(\pi(t)) & t \in \hat{\mathcal{T}} \end{cases} . \tag{3.62}$$

To clearly demonstrate the sink merge, a particular example is given in Fig. 3.15(b). Next, we define intermediate nodes merge.

Definition 20 (Intermediate Node Merge ($\mathsf{A}_1.g + \mathsf{A}_2.\pi(g)$)– Fig. 3.14(c))**:** The result A of merging the intermediate node $g \in \mathcal{G}_1$ from network A_1 with the intermediate node $\pi(g) \in \mathcal{G}_2$ from the disjoint network A_2 will be: $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2$; $\mathcal{G} = \mathcal{G}_1 \cup \mathcal{G}_2 \setminus \pi(g)$; $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$; $\mathcal{E} = \mathcal{E}_1 \cup \mathcal{E}_2 \cup \{(g_2, \mathcal{F}_2 \setminus \pi(g) \cup g) | g_2 \in \mathcal{G}_2, (g_2, \mathcal{F}_2 \cup \pi(g)) \in \mathcal{E}_2\} \cup \{(g, \mathcal{F}_2) | (\pi(g), \mathcal{F}_2) \in \mathcal{E}_2\} \setminus \{e \in \mathcal{E}_2 | \mathsf{Tl}(e) = \pi(g)\} \setminus \{e \in \mathcal{E}_2 | \pi(g) \in \mathsf{Hd}(e)\}$; and

$$\beta(t) = \begin{cases} \beta_1(t) & t \in \mathcal{T}_1 \\ \beta_2(t) & t \in \mathcal{T}_2 \end{cases} \tag{3.63}$$

To clearly demonstrate the node merge, a particular example is given in Fig. 3.15(c).

Definition 21 (Edge Merge ($\mathsf{A}_1.e + \mathsf{A}_2.\pi(e)$)–Fig. 3.14(d))**:** The result A of merging edge $e \in \mathcal{E}_1$ from network A_1 with edge $\pi(e) \in \mathcal{E}_2$ from disjoint network A_2 will be: $\mathcal{S} = \mathcal{S}_1 \cup \mathcal{S}_2$; $\mathcal{G} = \mathcal{G}_1 \cup \mathcal{G}_2 \cup g_0$, where $g_0 \notin \mathcal{G}_1, g_0 \notin \mathcal{G}_2$; $\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2$; $\mathcal{E} = (\mathcal{E}_1 \setminus e) \cup (\mathcal{E}_2 \setminus \pi(e)) \cup \{(\mathsf{Tl}(e), g_0), (\mathsf{Tl}(\pi(e)), g_0), (g_0, \mathsf{Hd}(e)), (g_0, \mathsf{Hd}(\pi(e)))\}$; and (3.63).

It is not difficult to see that this edge merge operation can be thought of as a special node merge operation. Suppose the edges being merged are $\mathsf{A}_1.e$, $\mathsf{A}_2.\pi(e)$. If two virtual nodes g_1, g_2 are added on $e, \pi(e)$, respectively, splitting them each into two edges, so that $e, \pi(e)$ go into and flow out g_1, g_2, respectively, then, the merge of g_1, g_2 gives the same network as merging $e, \pi(e)$. To clearly demonstrate the edge merge, a particular example is given in Fig. 3.15(d).

3.7.2 Preservation Properties of Combination Operations

Here we prove that the combination operations enable the rate regions of the small networks to be combined to get the rate region of the resulting large network, and also preserve sufficiency of classes of codes and tightness of other bounds.

Theorem 16: Suppose a network A is obtained by merging $\mathsf{A}_1.\hat{\mathcal{S}} = \mathsf{A}_2.\pi(\hat{\mathcal{S}})$, then for each $i \in \{*, q, (s,q), o\}$

$$\mathcal{R}_i(\mathsf{A}) = \mathrm{Proj}((\mathcal{R}_i(\mathsf{A}_1) \times \mathcal{R}_i(\mathsf{A}_2)) \cap \mathcal{L}_0), \tag{3.64}$$

with $\mathcal{L}_0 = \left\{H(X_s) = H(X_{\pi(s)}), \forall s \in \hat{\mathcal{S}}\right\}$, and the dimensions kept in the projection are $H(X_s), s \in \mathcal{S}, R_e, e \in \mathcal{E}$.

Remark 2: The inequality description of the polyhedral cone $\mathrm{Proj}((\mathcal{P}_1 \times \mathcal{P}_2) \cap \mathcal{L}_0)$ for two polyhedral cones $\mathcal{P}_j, j \in \{1,2\}$ can be created by concatenating the inequality descriptions for $\mathcal{P}_1$ and $\mathcal{P}_2$, then replacing the variable $H(X_{\pi(s)})$ with the $H(X_s)$ for each $s \in \hat{\mathcal{S}}$.

Proof: Select any point $\mathbf{R} \in \mathcal{R}_*(\mathsf{A})$, then there exist a sequence of random variables $\{X_s^{(j)}, s \in \mathcal{S}, U_i^{(j)}, i \in \mathcal{E}, j = 1, 2, \ldots\}$ whose entropies as $j \to \infty$ satisfy all the constraints determined by A. When decomposing A into $\mathsf{A}_1, \mathsf{A}_2$, let i.i.d. copies of variables $X_s^{(j)}, s \in \hat{\mathcal{S}}$ work as sources $\pi(\hat{\mathcal{S}}) \subseteq \mathcal{S}_2$. The associated edges connecting $\hat{\mathcal{S}}$ and nodes in $\mathcal{G}_2$ will then connect $\pi(\hat{\mathcal{S}})$ and nodes in $\mathcal{G}_2$. Then the random variables $\{X_s^{(j)}, s \in \mathcal{S}_1, U_i^{(j)}, i \in \mathcal{E}_1\}, \{X_s^{(j)}, s \in \mathcal{S}_2, U_i^{(j)}, i \in \mathcal{E}_2\}$ will satisfy the network constraints determined by $\mathsf{A}_1, \mathsf{A}_2$, and also $\mathcal{L}_0$, as $j \to \infty$. Thus, $\mathbf{R} \in \mathrm{Proj}((\mathcal{R}_*(\mathsf{A}_1) \times \mathcal{R}_*(\mathsf{A}_2)) \cap \mathcal{L}_0)$. Similarly, if $\mathbf{R}$ is achievable by $\mathbb{F}_q$ codes, vector or scalar, the same code applied to the part of A that is $\mathsf{A}_1, \mathsf{A}_2$ will achieve $\mathbf{R}_1, \mathbf{R}_2$, respectively. Putting these, together, we have $\mathcal{R}_i(\mathsf{A}) \subseteq \mathrm{Proj}((\mathcal{R}_i(\mathsf{A}_1) \times \mathcal{R}_i(\mathsf{A}_2)) \cap \mathcal{L}_0), i \in \{*, q, (s,q)\}$.

Next, if we select two points $\mathbf{R}_1 \in \mathcal{R}_*(\mathsf{A}_1)$, $\mathbf{R}_2 \in \mathcal{R}_*(\mathsf{A}_2)$ such that $H(X_s) = H(X_{\pi(s)}), \forall s \in \hat{\mathcal{S}}$, then there exist sequence of network codes for A_1 and A_2 achieving these points. By using the same source bits as the source inputs for s in A_1 and $\pi(s)$ in A_2 for each $s \in \hat{S}$, we have the same effect as using these source bits as the inputs for s in the source merged A and achieving the associated rate vector $\mathbf{R}$, implying $\mathbf{R} \in \mathcal{R}_*(\mathsf{A})$, and hence $\mathcal{R}_*(\mathsf{A}) \supseteq \mathrm{Proj}((\mathcal{R}_*(\mathsf{A}_1) \times \mathcal{R}_*(\mathsf{A}_2)) \cap \mathcal{L}_0)$. The same argument about using the same inputs for the merged sources of the two codes in A_1 and A_2 holds for $\mathcal{R}_{s,q}$ and $\mathcal{R}_q$ and implies $\mathcal{R}_{s,q}(\mathsf{A}) \supseteq \mathrm{Proj}((\mathcal{R}_{s,q}(\mathsf{A}_1) \times \mathcal{R}_{s,q}(\mathsf{A}_2)) \cap \mathcal{L}_0)$ and $\mathcal{R}_q(\mathsf{A}) \supseteq \mathrm{Proj}((\mathcal{R}_q(\mathsf{A}_1) \times \mathcal{R}_q(\mathsf{A}_2)) \cap \mathcal{L}_0)$. Together with statement above, these prove (3.64) for $i \in \{*, q, (s,q)\}$.

Furthermore, by (3.15), any point $\mathbf{R} \in \mathcal{R}_o(\mathsf{A})$, is the projection of some point $\mathbf{h} \in \Gamma_N^o \cap \mathcal{L}(\mathsf{A})$, and because the Shannon inequalities and network constraints in $\Gamma_N^o \cap \mathcal{L}(\mathsf{A})$ form a superset (i.e. include all of) of the network constraints in $\Gamma_N^o \cap \mathcal{L}(\mathsf{A}_i)$, the subvectors $\mathbf{h}_i$ of $\mathbf{h}$ associated only with the variables in A_i (with $X_{\pi(s)}$ being recognized as X_s for all $s \in \hat{S}$) are in $\Gamma_{N_i}^o \cap \mathcal{L}(\mathsf{A}_i)$ and obey $\mathcal{L}_0$, implying $\mathbf{R} \in \mathrm{Proj}((\mathcal{R}_o(\mathsf{A}_1) \times \mathcal{R}_o(\mathsf{A}_2)) \cap \mathcal{L}_0)$, and hence $\mathcal{R}_o(\mathsf{A}) \subseteq \mathrm{Proj}((\mathcal{R}_o(\mathsf{A}_1) \times \mathcal{R}_o(\mathsf{A}_2)) \cap \mathcal{L}_0)$.

Next, if we select select two points $\mathbf{R}_1 \in \mathcal{R}_o(\mathsf{A}_1)$, $\mathbf{R}_2 \in \mathcal{R}_o(\mathsf{A}_2)$ such that $H(X_s) = H(X_{\pi(s)}), \forall s \in$

$\hat{\mathcal{S}}$, then there exists $\mathbf{h}^i \in \Gamma^o_{N_i} \cap \mathcal{L}(\mathsf{A}_i)$ such that $\mathbf{R}_i = \mathrm{Proj}_{\boldsymbol{r},\boldsymbol{\omega}}\mathbf{h}^i$, $i \in \{1,2\}$ with $h^1_{X_s} = h^2_{X_{\pi(s)}}$ for all $s \in \hat{S}$. Define $\mathbf{h}$ whose element associated with the subset $\mathcal{A}$ of $\mathcal{N} = \mathcal{S} \cup \mathcal{E}$ is $h_{\mathcal{A}} = h^1_{\mathcal{A}\cap\mathcal{N}_1} + h^2_{\mathcal{A}\cap\mathcal{N}_2} - h^2_{\mathcal{A}\cap\pi(\hat{\mathcal{S}})}$ where $\mathcal{N}_i = \mathcal{S}_i \cup \mathcal{E}_i$, $i \in \{1,2\}$. By virtue of its creation this way, this function is submodular and $\mathbf{h} \in \Gamma^o_N$. Since the two networks are disjoint, the list of equalities in $\mathcal{L}_3(\mathsf{A})$ is simply the concatenation of the lists in $\mathcal{L}_3(\mathsf{A}_1)$ and $\mathcal{L}_3(\mathsf{A}_2)$, each of which involved inequalities in disjoint variables $\mathcal{N}_1$ and $\mathcal{N}_2$, and the same thing holds for $\mathcal{L}_{4'}$. Furthermore, since $\mathbf{h}^i \in \mathcal{L}_2(\mathsf{A}_i)$ and $h^1_{X_s} = h^2_{X_{\pi(s)}}, s \in \hat{\mathcal{S}}$, $\mathbf{h}$ obeys $\mathcal{L}_2(\mathsf{A})$. The definition of $\mathbf{h}$, together with $\mathbf{h}^i \in \mathcal{L}_1(\mathsf{A}_i)$, $i \in \{1,2\}$ and $h^1_{X_s} = h^2_{X_{\pi(s)}}$, $s \in \hat{\mathcal{S}}$, implies that $\mathbf{h} \in \mathcal{L}_1(\mathsf{A})$. Finally $\mathbf{h}^1 \in \mathcal{L}(\mathsf{A}_1)$ and $\mathbf{h}^2 \in \mathcal{L}(\mathsf{A}_2)$ imply $\mathbf{h} \in \mathcal{L}_5(\mathsf{A})$. Putting these facts together we observe that $\mathbf{h} \in \Gamma^o_N \cap \mathcal{L}(\mathsf{A})$, so $\mathbf{R} \in \mathcal{R}_o(\mathsf{A})$, implying $\mathcal{R}_o(\mathsf{A}) \supseteq \mathrm{Proj}((\mathcal{R}_o(\mathsf{A}_1) \times \mathcal{R}_o(\mathsf{A}_2)) \cap \mathcal{L}_0)$, completing (3.64) for $i = o$. ■

Theorem 17: Suppose a network A is the sink nodes merge $(\mathsf{A}_1.\hat{\mathcal{T}} + \mathsf{A}_2.\pi(\hat{\mathcal{T}}))$, then

$$\mathcal{R}_i(\mathsf{A}) = \mathcal{R}_i(\mathsf{A}_1) \times \mathcal{R}_i(\mathsf{A}_2),\ i \in \{*, q, (s,q), o\} \tag{3.65}$$

with the index on the dimensions mapping from $\{e \in \mathcal{E}_2 | \mathsf{Hd}(e) \in \pi(\hat{\mathcal{T}})\}$ to $\{e \in \mathcal{E} | \mathsf{Hd}(e) \in \hat{\mathcal{T}}, \mathsf{Tl}(e) \in \mathcal{G}_2\}$.

Proof: Consider a point $\mathbf{R} \in \mathcal{R}_k(\mathsf{A})$ for any $k \in \{*, q, (s,q), o\}$, and (sequence of) random variables associated with the associated (sequence of) codes. Due to the independence of sources in networks $\mathsf{A}_1, \mathsf{A}_2$, and the fact that their sources and intermediate nodes are disjoint, the variables arriving at a merged sink node from A_1 will be independent of the sources in A_2 and the variables arriving at a merged sink node from A_2 will be independent of the sources in A_1. In particular, Shannon type inequalities imply the Markov chains $H(\mathbf{Y}_{\mathcal{S}_1}|\mathbf{U}_{\mathrm{In}(t)\cap\mathcal{E}_1}, \mathbf{U}_{\mathrm{In}(t)\cap\mathcal{E}_2}) = H(\mathbf{Y}_{\mathcal{S}_1}|\mathbf{U}_{\mathrm{In}(t)\cap\mathcal{E}_1})$ and $H(\mathbf{Y}_{\mathcal{S}_2}|\mathbf{U}_{\mathrm{In}(t)\cap\mathcal{E}_1}, \mathbf{U}_{\mathrm{In}(t)\cap\mathcal{E}_2}) = H(\mathbf{Y}_{\mathcal{S}_2}|\mathbf{U}_{\mathrm{In}(t)\cap\mathcal{E}_2})$ for all $t \in \mathcal{T}$ (even if the associated "entropies" are only in Γ^o_N and not necessarily $\bar{\Gamma}^*_N$). This then implies, together with the independence of the sources, that $H(\mathbf{Y}_{\beta(t)}|\mathbf{U}_{\mathrm{In}(t)}) = H(\mathbf{Y}_{\beta(t)\cap\mathcal{S}_1}|\mathbf{U}_{\mathrm{In}(t)\cap\mathcal{E}_1}) + H(\mathbf{Y}_{\beta(t)\cap\mathcal{S}_2}|\mathbf{U}_{\mathrm{In}(t)\cap\mathcal{E}_2})$, showing that the constraints in $\mathcal{L}_5(\mathsf{A})$ imply the constraints in $\mathcal{L}_5(\mathsf{A}_1)$ and $\mathcal{L}_5(\mathsf{A}_2)$. Furthermore, given the disjoint nature of A_1 and A_2, the constraints in $\mathcal{L}_i(\mathsf{A})$, are simply the concatenation of the constraints in $\mathcal{L}_i(\mathsf{A}_1)$ and $\mathcal{L}_i(\mathsf{A}_2)$, for $i \in \{2, 3, 4'\}$. Furthermore, the joint independence of all of $\mathbf{Y}_{\mathcal{S}_1}, \mathbf{Y}_{\mathcal{S}_2}$ imply the marginal independence of the collections of variables $\mathbf{Y}_{\mathcal{S}_1}$ and $\mathbf{Y}_{\mathcal{S}_2}$, so that $\mathcal{L}_1(\mathsf{A})$ implies $\mathcal{L}_1(\mathsf{A}_i), i \in \{1,2\}$. This shows that $\mathbf{R} \in \mathcal{R}_k(\mathsf{A}_1) \times \mathcal{R}_k(\mathsf{A}_2)$, and hence $\mathcal{R}_k(\mathsf{A}) \subseteq \mathcal{R}_k(\mathsf{A}_1) \times \mathcal{R}_k(\mathsf{A}_2), k \in \{*, q, (s,q), o\}$.

Next, consider two points $\mathbf{R}_i \in \mathcal{R}_k(\mathsf{A}_i)$, $i \in \{1,2\}$ for any $k \in \{q, (s,q), o\}$. By definition these are projections of $\mathbf{h}^i \in \Gamma^k_{N_i} \cap \mathcal{L}(\mathsf{A}_i)$, $i \in \{1,2\}$. Define $\mathbf{h}$ with value associated with subset $\mathcal{A} \subseteq \mathcal{N}$ of

$h_{\mathcal{A}} = h^1_{\mathcal{A}\cap\mathcal{N}_1} + h^2_{\mathcal{A}\cap\mathcal{N}_2}$, then it is easily verified that the resulting $\mathbf{h} \in \Gamma_N^k \cap \mathcal{L}(\mathsf{A})$ (simply use the same codes from A_1 and A_2 on the corresponding parts of A). Since $\mathbf{R} = \mathrm{Proj}_{\boldsymbol{\omega},\boldsymbol{r}}\mathbf{h}$, we have proven $\mathbf{R} \in \mathcal{R}_k(\mathsf{A})$, and hence that $\mathcal{R}_k(\mathsf{A}) \supseteq \mathcal{R}_k(\mathsf{A}_1) \times \mathcal{R}_k(\mathsf{A}_2)$. Further, for two points $\mathbf{R}_i \in \mathcal{R}_*(\mathsf{A}_i),\ i \in \{1,2\}$, there exist a sequence of random variables $\{X_s^{(j)}, s \in \mathcal{S}_1, U_i^{(j)}, i \in \mathcal{E}_1\}, \{X_s^{(j)}, s \in \mathcal{S}_2, U_i^{(j)}, i \in \mathcal{E}_2\}$ satisfy the network constraints determined by $\mathsf{A}_1, \mathsf{A}_2$, as $j \to \infty$. Due to the independence of sources and disjoint of edge variables, the sequence of the union of variables in $\mathsf{A}_1, \mathsf{A}_2$ will satisfy the network constraints in the merged A, as $j \to \infty$. Thus, $\mathcal{R}_*(\mathsf{A}) \supseteq \mathcal{R}_*(\mathsf{A}_1) \times \mathcal{R}_*(\mathsf{A}_2)$. ■

Theorem 18: Suppose a network A is obtained by intermediate nodes merge $(\mathsf{A}_1.g + \mathsf{A}_2.\pi(g))$, then

$$\mathcal{R}_i(\mathsf{A}) = \mathcal{R}_i(\mathsf{A}_1) \times \mathcal{R}_i(\mathsf{A}_2),\ i \in \{*, q, (s,q), o\} \tag{3.66}$$

with dimensions indices mapping from $\{e \in \mathcal{E}_2 | \mathsf{Hd}(e) = \pi(g)\}$ to $\{e \in \mathcal{E} | \mathsf{Hd}(e) = g, \mathsf{Tl}(e) \in \mathcal{G}_2\}$ and from $\{e \in \mathcal{E}_2 | \mathsf{Tl}(e) = \pi(g)\}$ to $\{e \in \mathcal{E} | \mathsf{Tl}(e) = g, \mathsf{Hd}(e) \in \mathcal{G}_2\}$.

Proof: Consider a point $\mathbf{R} \in \mathcal{R}_k(\mathsf{A})$ for any $k \in \{*, q, (s,q)\}$ and all associated random variables (or sequence of variables when $\mathbf{R} \in= \mathcal{R}_*$) that satisfy $\mathcal{L}_i(\mathsf{A}), i = 1,2,3,4',5$. Partition the incoming edges of the merged node g in A, $\mathrm{In}(g)$, up into $\mathrm{In}_1(g) = \mathrm{In}(g) \cap \mathcal{E}_1$ the edges from A_1, and $\mathrm{In}_2(g) = \mathrm{In}(g) \setminus \mathrm{In}_1(g)$, the new incoming edges resulting from the merge. Similarly, partition the outgoing edges $\mathrm{Out}(g)$ up into $\mathrm{Out}_1(g) = \mathrm{Out}(g) \cap \mathcal{E}_1$ and $\mathrm{Out}_2(g) = \mathrm{Out}(g) \setminus \mathrm{Out}_1(g)$. The $\mathcal{L}_3$ constraints dictate that there exist functions f_e such that for each $e \in \mathrm{Out}(g)$, $U_e = f_e(U_{\mathrm{In}_1(g)}, U_{\mathrm{In}_2(g)})$. Define the new functions f'_e via

$$f'_e(U_{\mathrm{In}_1(g)}, U_{\mathrm{In}_2(g)}) = \begin{cases} f_e(U_{\mathrm{In}_1(g)}, \mathbf{0}) & e \in \mathrm{Out}_1(g) \\ f_e(\mathbf{0}, U_{\mathrm{In}_2(g)}) & e \in \mathrm{Out}_2(g) \end{cases} \tag{3.67}$$

i.e., set the possible value for the incoming edges from the other part of the network (possibly erroneously) to a particular constant value among their possible values – let's label it $\mathbf{0}$. The network code using these new functions f'_e will utilize the same rates as before. The constraints and the topology of the merged network further dictated that $U_{\mathrm{In}_i(g)}$ were expressible as a function of $\mathcal{S}_i$, $i \in \{1,2\}$. In the remainder of the network (moving toward the sink nodes) after the merged nodes, at no other point is any information from the sources in the other part of the network encountered, and the decoders at the sink nodes in $\mathcal{T}_2$ need to work equally well decoding subsets of $\mathcal{S}_2$, regardless of the value of $\mathcal{S}_1$. Since the erroneous value for the $U_{\mathrm{In}_1(g)}$ used for $f'_e, e \in \mathrm{Out}_2(g)$ was still a valid possibility for some (possible other) value(s) of the sources in $\mathcal{S}_1$, the sinks must still produce

the correct values for their subsets of $\mathcal{S}_2$. A parallel argument for $\mathcal{T}_1$ shows that they still correctly decode their sources, which were subsets of $\mathcal{S}_1$, even though the f_es were changed to f'_es. Note further that (3.67) will still be scalar/vector linear if the original f_es were as well.

However, since the $f_{e'}$s no longer depend on the other half of the network, the resulting code can be used as separate codes for A_1 and A_2, given the associated rate points $\mathbf{R}_i$ by keeping the elements in $\mathbf{R}$ associated with A_i, $i \in \{1,2\}$ in the natural way, implying that $\mathbf{R} \in \mathcal{R}_k(\mathsf{A}_1) \times \mathcal{R}_k(\mathsf{A}_2)$. This then implies that $\mathcal{R}_k(\mathsf{A}) \subseteq \mathcal{R}_k(\mathsf{A}_1) \times \mathcal{R}_k(\mathsf{A}_2)$ for all $k \in \{*, q, (s,q)\}$. The opposite containment is obvious, since any codes for the two networks can be utilized in the trivial manner for the merged network. This proves (3.66) for $k \in \{*, q, (s,q)\}$.

Next, consider any pair $\mathbf{R}_i \in \mathcal{R}_o(\mathsf{A}_i)$ $i \in \{1,2\}$, which are, by definition, projections of some $\mathbf{h}^i \in \Gamma^o_{N_i} \cap \mathcal{L}(\mathsf{A}_i)$, $i \in \{1,2\}$. Defining $\mathbf{h}$ whose element associated with the subset $\mathcal{A} \subset \mathcal{N}$ is $h_{\mathcal{A}} = h^1_{\mathcal{A}\cap\mathcal{N}_1} + h^2_{\mathcal{A}\cap\mathcal{N}_2}$, where the intersection respects the remapping of edges under the intermediate node merge, we observe that $\mathbf{h} \in \Gamma^o_N \cap \mathcal{L}(\mathsf{A})$, and hence its projection $\mathbf{R} \in \mathcal{R}_o(\mathsf{A})$, proving $\mathcal{R}_o(\mathsf{A}) \supseteq \mathcal{R}_o(\mathsf{A}_1) \times \mathcal{R}_o(\mathsf{A}_2)$.

Finally, consider a $\mathbf{R} \in \mathcal{R}_o(\mathsf{A})$, which is a projection of some $\mathbf{h} \in \Gamma^o_N \cap \mathcal{L}(\mathsf{A})$. For every $\mathcal{A} \subseteq \mathcal{N}_i$, define $h^i_{\mathcal{A}} = h_{\mathcal{A}\cup\mathcal{S}_{3-i}} - h_{\mathcal{S}_{3-i}}$, and define $\mathbf{h}'$ with $h'_{\mathcal{A}} = h^1_{\mathcal{A}\cap\mathcal{N}_1} + h^2_{\mathcal{A}\cap\mathcal{N}_2}$ with $\mathbf{R}' = \mathrm{proj}_{\boldsymbol{\omega},\boldsymbol{r}}\mathbf{h}'$. $\mathbf{h}^i \in \mathcal{L}(\mathsf{A}_i), i \in \{1,2\}$ because conditioning reduces entropy and is non-negative, but all of the conditional entropies at nodes other than g were already zero, while at g, the conditioning on the sources from the other network will enable the same conditional entropy of zero since the incoming edges from the other network were functions of them. This shows that $\mathbf{R}' \in \mathcal{R}_o(\mathsf{A}_1) \times \mathcal{R}_o(\mathsf{A}_2)$. Owing to the independence of the sources $\mathrm{proj}_{\boldsymbol{\omega}}\mathbf{h} = \mathrm{proj}_{\boldsymbol{\omega}}\mathbf{h}'$, while $\mathrm{proj}_{\boldsymbol{r}}\mathbf{h} \geq \mathrm{proj}_{\boldsymbol{r}}\mathbf{h}'$ due to the fact that conditioning reduces entropy. The coordinate convex nature then implies that $\mathbf{R} \in \mathcal{R}_o(\mathsf{A}_1) \times \mathcal{R}_o(\mathsf{A}_2)$ showing that $\mathcal{R}_o(\mathsf{A}) \subseteq \mathcal{R}_o(\mathsf{A}_1) \times \mathcal{R}_o(\mathsf{A}_2)$ and completing the proof. ■

Theorem 19: Suppose a network A is obtained by edge merge $(\mathsf{A}_1.e + \mathsf{A}_2.\pi(e))$, then

$$\mathcal{R}_i(\mathsf{A}) = \mathrm{Proj}_{\setminus\{e,\pi(e)\}}((\mathcal{R}_i(\mathsf{A}_1) \times \mathcal{R}_i(\mathsf{A}_2)) \cap \mathcal{L}'_0), \tag{3.68}$$

$\mathcal{L}'_0 = \{R_{(\mathrm{Tl}(j),g_0)} \geq R_j, R_{(g_0,\mathrm{Hd}(j))} \geq R_j, j \in \{e,\pi(e)\}\}$, $i \in \{*, q, (s,q), o\}$, $\mathcal{R}(\mathsf{A}_q) \times \mathcal{R}(\mathsf{A}_q)$ and $\mathcal{L}'_0$ are viewed in the dimension of $|\mathcal{N}_1| + |\mathcal{N}_2| + 4$ with assumption that all dimensions not shown are unconstrained.

Proof: As observed after the definition of edge merge, one can think of edge merge as the concatenation of two operations, one which splits e in A_1 and $\pi(e)$ in A_2 each up into two edges with a

new intermediate node (g and $\pi(g)$, respectively) in between them, forming A_1' and A_2', respectively, followed by intermediate node merge of $\mathsf{A}_1'.g + \mathsf{A}_2'.\pi(g)$. It is clear that $\mathcal{L}_0'$ describes the operation that must happen to the rate region of A_i, $i \in \{1, 2\}$ to get the rate region of A_i', because the contents of the old edge e or $\pi(e)$ must now be carried by both new edges after the introduction of the new intermediate node. Applying Thm. 18 to A_1' and A_2' yields (3.68). ■

Corollary 7: Let network A be a combination of networks $\mathsf{A}_1, \mathsf{A}_2$ via one of the operations defined in §3.6. If $\mathbb{F}_q$ vector (scalar) linear codes suffice or the Shannon outer bound is tight for both $\mathsf{A}_1, \mathsf{A}_2$, then the same will be true for A. Equivalently, if $\mathcal{R}_k(\mathsf{A}_i) = \mathcal{R}_*(\mathcal{A}_i), i \in \{1, 2\}$ for some $k \in \{o, q, (s, q)\}$ then also $\mathcal{R}_k(\mathsf{A}) = \mathcal{R}_*(\mathsf{A})$.

Now we have defined both embedding and combination operators, and have demonstrated methods to obtain the rate regions of networks after applying the operators. Next, we would like to discuss how to use them to do network analysis and solve large networks.

3.8 Results with Operators

In this section, we demonstrate the use of the operators defined in §3.6 and §3.7. We will first discuss the use of network embedding operations (§3.6) to obtain the forbidden network minors and to predict code sufficiency for a larger network given the sufficiency is known for the smaller embedded network. Then, we discuss the use of combination operations (§3.7) to solve large networks. Finally, we discuss how to use both combination and embedding operations together to obtain even more solvable networks.

3.8.1 Use Embedding Operations to Obtain Network Forbidden Minors

One natural use of embedding operations is to obtain rate regions for smaller embedded networks given the rate region for a larger network, as shown in §3.6, since the rate regions of embedded networks are projections of the rate region of the larger network with some constraints.

We can use the embedding operators in a reverse manner. From Corollaries 5 and 6, we observe that if a class of linear codes suffice for a larger network, then it will suffice for the smaller networks embedded in it as well. Equivalently, the insufficiency of a class of codes for a network is inherited by the larger networks that have this network embedded inside. This is similar to the forbidden minor property in matroid theory, which states that if a matroid is not $\mathbb{F}_q$ representable, then neither are its extensions. Therefore, if we know a small network has the property that a class of linear codes does not suffice, then we can predict that all networks containing it as a *network minor* through

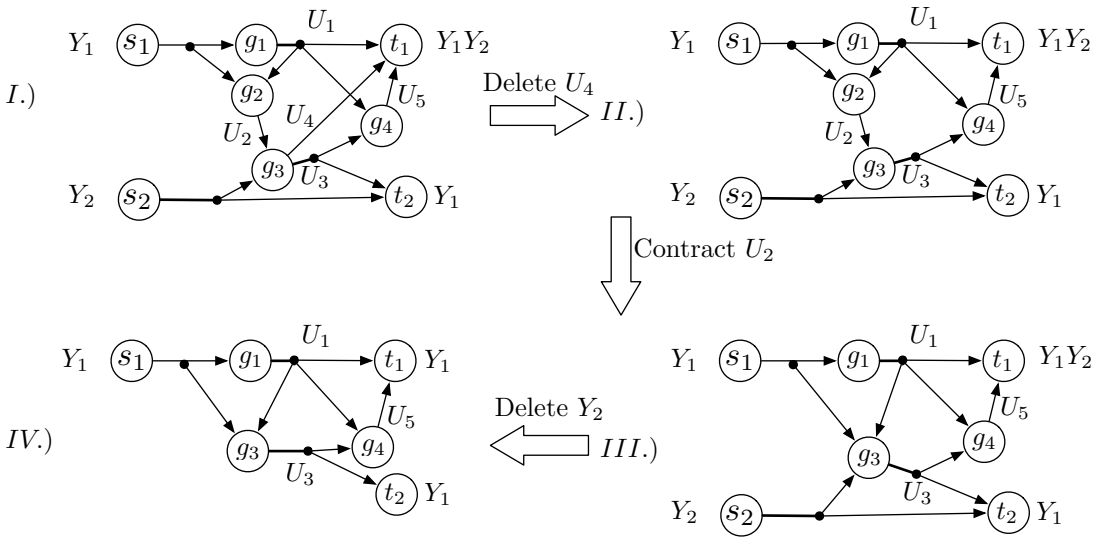


Figure 3.16: Using embedding operations to predict the insufficiency of scalar binary codes for a large network: since the large network I and intermediate stage networks II, III contain the small network IV as a minor, the insufficiency of scalar binary codes for network IV, which is easy to see, predicts the same property for networks III, II and I.

some embedding operations, will have the same property, without any calculations. For instance, in Fig. 3.16, the small network IV is a $(1,3)$ network for which scalar binary codes do not suffice, because when $H(Y_1) = 2, H(U_1) = H(U_3) = H(U_5) = 1$, there is no scalar solution but there is a vector solution. Since networks I, II, III contain it as a minor (through the operations in the figure), we know that scalar binary codes will not suffice for them, either. Actually this is verified by our computations in §3.5.

Similar as in Matroid theory, we may have a collection of small networks that we know they should be "forbidden" as a minor in larger networks, regarding the sufficiency of a class of linear codes. For instance, in [41], it was shown that for the thousands of MDCS networks that scalar binary codes do not suffice, there are actually only 12 forbidden network minors. For general hyperedge MSNC problems, we also built similar relationships as shown in Fig. 3.17. As Table 3.3 shows, the numbers of instances that scalar binary codes do not suffice for $(1,3), (3,1), (2,2), (3,2), (2,3)$ networks are $5, 10, 32, 121064, 144484$, respectively. Those insufficient instances should not be a minor of any larger network that scalar binary codes suffice, and hence are forbidden minors. To obtain a minimal list of network forbidden minors, we investigate their inherent relations. As Fig. 3.17 shows, there are 37430 out of 121064 insufficient $(3,2)$ networks actually containing smaller forbidden network minors, of which 14867 can be related by source or edge deletion and 22563 can be related by edge contraction. Similarly, there are 37739 out of 144484 insufficient $(2,3)$ networks actually containing smaller forbidden network minors, of which 22444 can be related by source or edge deletion and

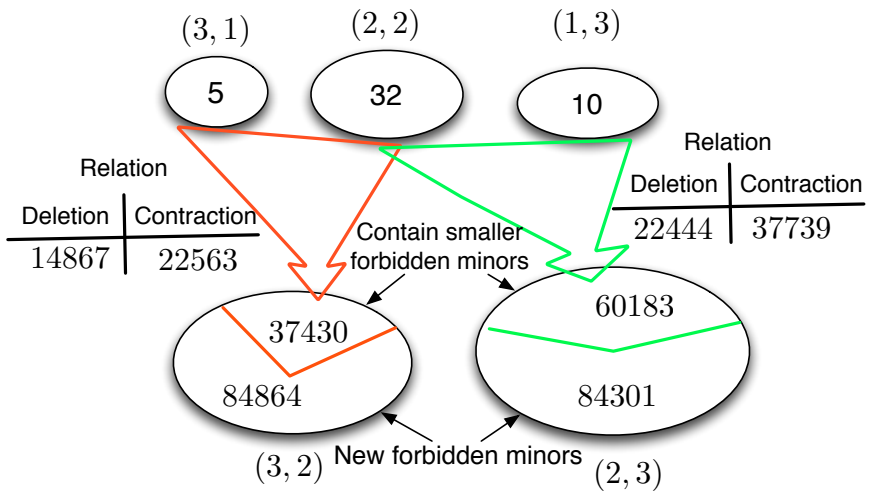


Figure 3.17: Relations between networks of different sizes that scalar binary codes do not suffice.

37739 can be related by edge contraction. Those networks that do not have smaller forbidden network minors, are new ones.

Next, we will discuss the use of combination operations.

3.8.2 Use Combination Operations to Solve Large Networks

As shown in §3.7, the rate region of a combined network can be directly obtained from the rate regions of the networks involved in the combination. We show that the sufficiency of a class of linear codes are preserved after combination. Actually, if we know one of the networks involved in the combination has the property that a class of linear codes does not suffice, the combined network will have the same property, since the small network is embedded in the combined network and they have minor relationship.

To demonstrate the idea of solving large networks obtained by combination operations, we give an example in Fig. 3.18. It shows that the rate region of the large combined network can be obtained directly from the rate regions of smaller network. In addition, sufficiency of linear codes is preserved.

Example 9: A $(6, 15)$ network instance A can be obtained by combining five smaller networks $\mathsf{A}_1, \ldots, \mathsf{A}_5$, of which the representations are shown in Fig. 3.18. The combination process is I) $\mathsf{A}_{12} = \mathsf{A}_1.\{t_1, t_2\} + \mathsf{A}_2.\{t_{1'}, t_{2'}\}$; II) $\mathsf{A}_{123} = \mathsf{A}_{12}.e_4 + \mathsf{A}_3.e_7$ with extra node g_0 and edges $e_{4'}, e_{7'}$; III) $\mathsf{A}_{45} = \mathsf{A}_4.g_{10} + \mathsf{A}_5.g_{10'}$; IV) $\mathsf{A} = \mathsf{A}_{123}.\{X_1, \ldots, X_6\} = \mathsf{A}_{45}.\{X_{1'}, \ldots, X_{6'}\}$. From the software calculations and analysis [37, 49], one obtains the rate regions below the 5 small networks. According to the theorems in §3.7, the rate region $\mathcal{R}_*(\mathsf{A})$ for A obtained from $\mathcal{R}_*(\mathsf{A}_1), \ldots, \mathcal{R}_*(\mathsf{A}_5)$, is depicted next to it. Additionally, since cal-

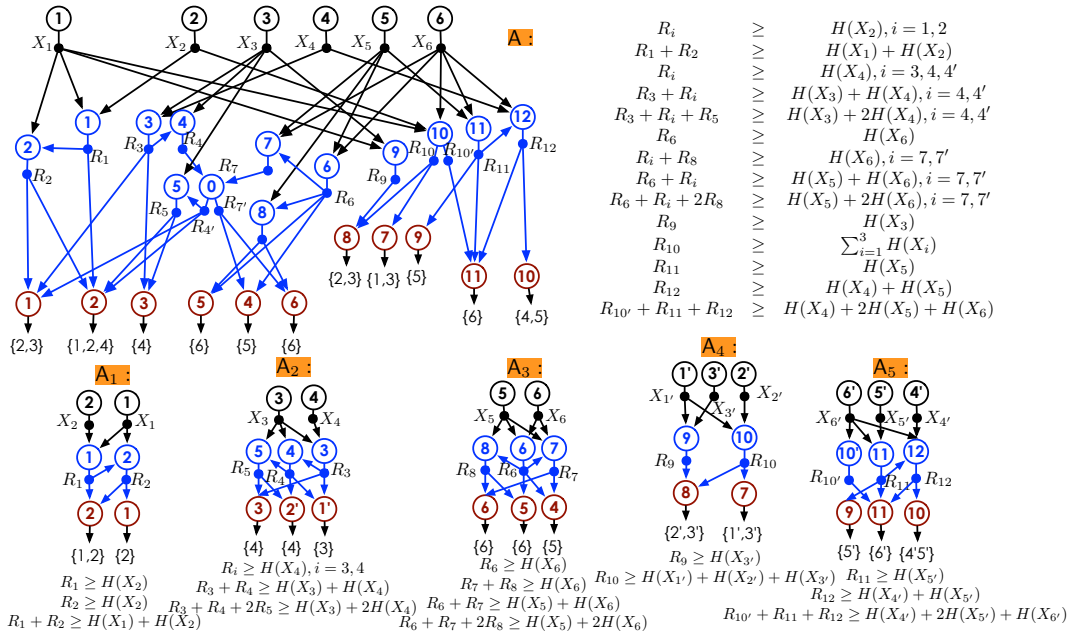


Figure 3.18: A large network and its capacity region created with the operations in this chapter from the 5 networks below it.

culations showed binary codes and the Shannon outer bound suffice for A_i, $i \in \{1, \ldots, 5\}$, Corollary 7 dictates the same for A.

3.8.3 Use combination and embedding operations to generate solvable networks

The combination operators provide a method for building large networks directly from smaller networks in such a way that the rate region of the large network can be directly obtained from those of the small networks. The embedding operators provide a method for obtaining a small network from a large network in such a way that the rate region of the small network can be expressed in terms of the rate region of the large network. These facts indicate that we can obtain networks by integrating combination and embedding operations. If we start with a list of solved networks, all networks obtained in the following combination and/or embedding process will be solvable. Note that one can apply these operations, especially the combination operations, infinite times to obtain infinite number of networks. For demonstration purpose, we would like to limit the size of networks involved in the process. We limit the worst case network sizes. As §3.7 shows, it is not difficult to predict the worst case network size after combination. Since the combined network may have redundancies, the worst case here means there is no redundancy after combination so that the network size is easy to predict. For instance, after merging k sources of a $(K, |\mathcal{E}_U|)$ with k sources of a $K', |\mathcal{E}'_U|$ network, the network size after merging will be $K + K' - k + |\mathcal{E}_U| + |\mathcal{E}'_U|$ in the worst case.

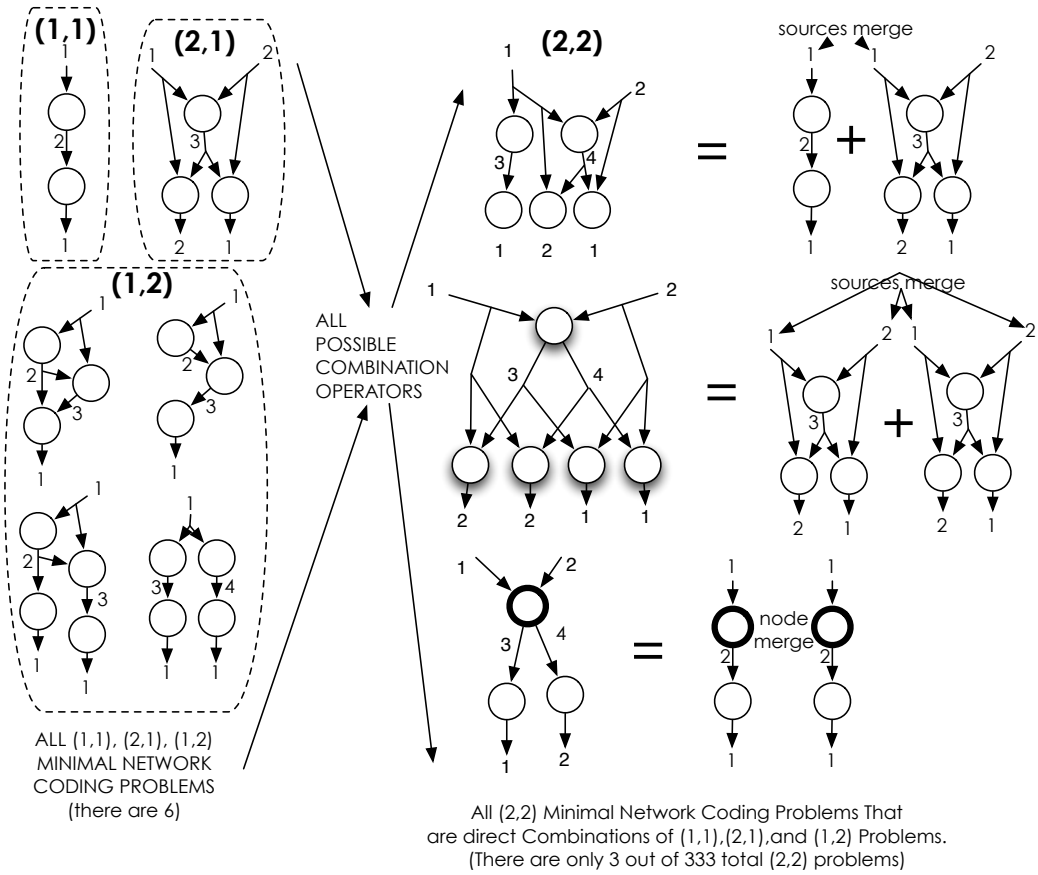


Figure 3.19: There are a total of 3 *minimal* $(2,2)$ *network coding problems directly resulting from combinations of the* 6 *small network coding problems with sizes* $(1,1)$, $(1,2)$, *and* $(2,1)$. *However, as shown in Fig. 3.20, by utilizing* both *combinations* and *embeddings operators, far more* $(2,2)$ *cases can be reached by iteratively combining and embedding the pool of networks starting from these 6* $(1,1)$, $(1,2)$, *and* $(2,1)$ *networks via Algorithm 6.*

```
Input: Seed list of networks seedList, size limits on number of sources and edges
Output: All network instances generated by combination and/or embedding operations on
        the networks in seed list

Initialization: network list for previous round prevList = ∅, new networks from previous
round prevAdd = seedList, current list of networks curList = ∅, new networks generated in
current round curAdd = ∅;
while size(prevAdd) > 0 do
    for every pair I × J ∈ prevAdd × prevAdd do
        if prediction of network size after merge does not exceed size limits then
            consider source, sink, node, edge merge on I, J;
            convert the new network to its canonical form newNet ;
            if newNet ∉ curList then
                curAdd = curAdd ∪ newNet;
            end
        end
    end
    for every pair I × J ∈ prevAdd × prevAdd do
        if prediction of network size after merge does not exceed size limits then
            consider source, sink, node, edge merge on I, J;
            convert the new network to its canonical form newNet ;
            if newNet ∉ curList then
                curAdd = curAdd ∪ newNet;
            end
        end
    end
    for every I ∈ prevAdd do
        consider source deletion, edge deletion and contraction on I;
        convert the new network to its canonical form newNet ;
        if newNet ∉ curList then
            curAdd = curAdd ∪ newNet;
        end
    end
    prevAdd = curAdd;
    prevList = curList;
    curList = curList ∪ curAdd;
end
```

Algorithm 6: Generate all networks from a seed list of small networks using combination and embedding operations.

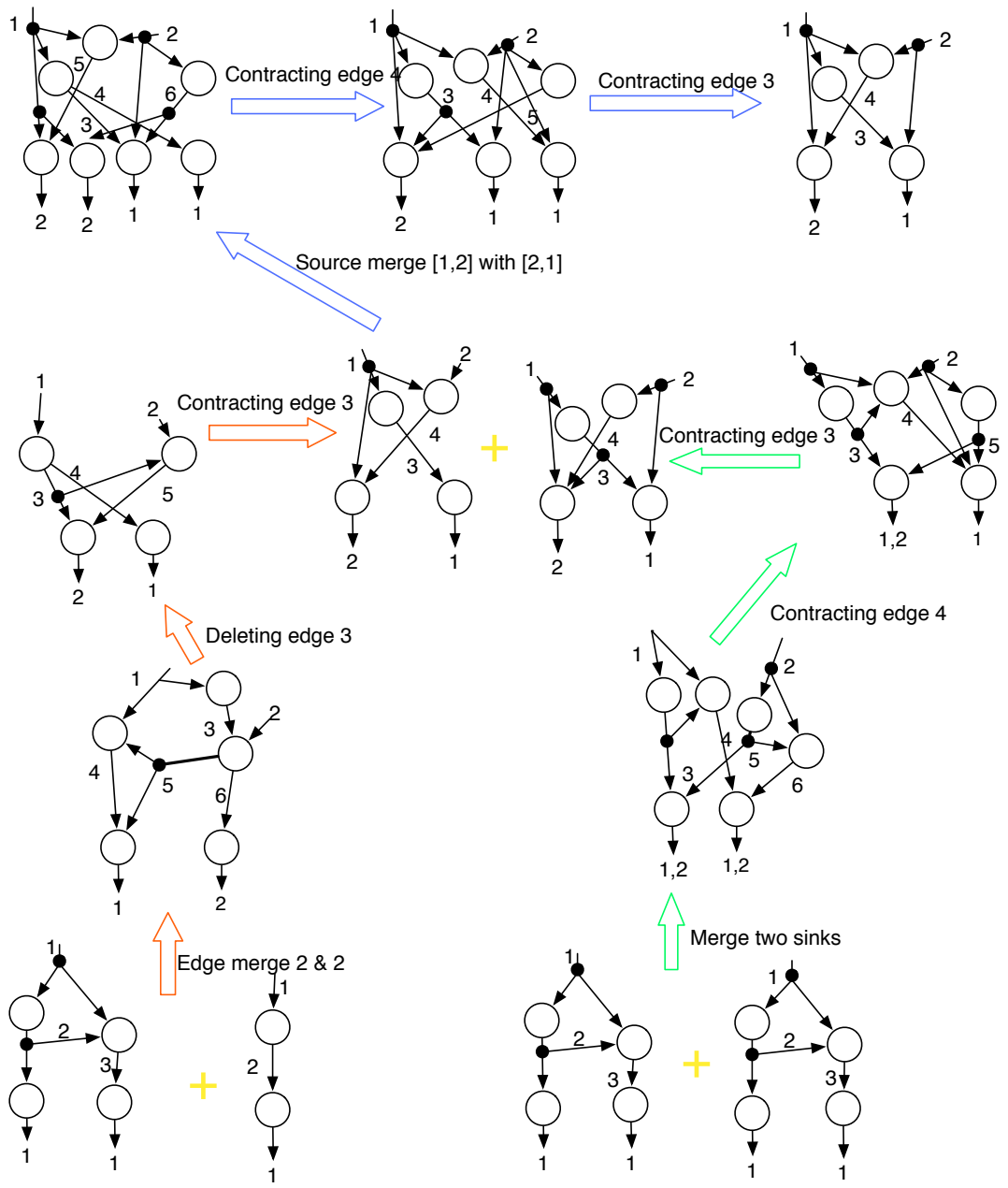


Figure 3.20: The path of operations on a seed list of small networks to get a $(2, 2)$ network that cannot be directly obtained by simple combination. The size limits on networks involved in the operation process is $K \leq 3, |\mathcal{E}_U| \leq 4$.

Table 3.5: The number of new canonical minimal network coding problems that can be generated from the 6 smallest canonical minimal network coding problems (the single $(1,1)$ *network, the single* $(2,1)$ *network, and the four* $(1,2)$*), by using combination operators (left), and both combination and embedding operators (right), in a partial closure operation where the largest network involved in a chain of operations never exceeds the "cap" (different columns). Note that these six networks can generate a very large number of larger networks (see the bottom row of the table), there is a gigantic benefit from using both combination embedding operators – moving in a nonmonotonic direction in network size, and the number of networks generated grows, even for small target network sizes, rapidly as the cap on the largest network size is increased.*

	combination operators only			embedding and combinations		
size\cap	(3,3)	(3,4)	(4,4)	(3,3)	(3,4)	(4,4)
(1,3)	4	4	4	4	4	4
(1,4)	0	10	10	0	10	10
(2,2)	3	3	3	8	15	16
(2,3)	13	16	16	30	131	155
(2,4)	0	97	101	0	516	648
(3,2)	2	3	2	4	10	11
(3,3)	24	24	24	42	353	833
(3,4)	0	135	135	0	2361	5481
(4,2)	0	0	3	0	0	3
(4,3)	0	0	17	0	0	44
(4,4)	0	0	253	0	0	4430
all	46	292	568	88	3400	11635

We define the *worst case partial closure of networks* as the networks obtained in the combination and embedding process that no network involved exceeds the size limit. An algorithm to generate the worst case partial closure of networks is shown in Algorithm 6. As it shows, as long as the predicted network does not exceed the preset size limit, it will be counted as a new network and will be used as a seed, as long as it is not isomorphic to the existing ones. We start with a seed list of networks, then through combinations and embeddings of these networks, a list of new networks will be generated, which will, in turn, be used as seeds again. The process stops when there is no new network that would not exceed the size limitations (aka. size cap) found, in the sense of closed under these operations. This quite powerful tool is able to generate a very large number of network from even small seed lists. For instance, if we use as a seed list the single $(1,1)$ and single $(1,2)$, together with the four $(1,2)$ networks, and set the size limit for intermediate networks to $K \leq 4, |\mathcal{E}_U| \leq 4$, there will be 11636 new networks generated. Even when embedding operations are not allowed, there will be 568 new networks. The details on number of networks generated for different size limits is shown in Table 3.5.

It is important to note that, when trying to calculate the rate regions of larger networks from a list of rate regions for smaller networks, *both combinations* <u>*and*</u> *embeddings* are useful. As Table 3.5 shows, when no embedding operations are allowed in the generation process, the number of reachable networks are much less than those with embedding operations. To further see a demonstration of this fact, consider all 6 canonical minimal network coding problems of dimensions $(1,1)$, $(1,2)$, $(2,1)$

as depicted in Fig. 3.19. There are only 3 out of 333 networks with size $K = 2, |\mathcal{E}_u| = 2$ can be reached by combination of smaller networks using only combination operations. The three networks and their smaller components are shown in Fig. 3.19. However, if we are allowed to use both combination and embedding operation together, we will reach more networks that are not reachable by merely combinations. As Fig. 3.20 shows, though we still use the same network pool as in Fig. 3.19, another $(2, 2)$ network is obtained by first combining smaller networks to a larger size and then using embedding operations to decrease the network size. Several steps of combination and embedding operations are necessary to get this network, and Fig. 3.19 shows the path through the operations from the initial seed list and the intermediately obtained network to reach it. As table 3.5 shows, there are at least 12 $(2, 2)$ networks which can be obtained in this manner, and the other rows show that the number of networks of a given size grows rapidly as the size cap for the partial operator closure increases.

3.9 Conclusions and Future Work

This chapter investigated the enumeration, rate region computation, and hierarchy of general multi-source multi-sink hyperedge networks. The network model includes several special ones such as independent distributed storage systems and index coding problems. This definition is further refined to a notion of minimal networks, containing no redundant sources, edges, or nodes, whose rate regions directly determine the rate regions of non-minimal networks. Furthermore, since networks related to one another through a permutation of the edge labels are derivable from one another, a notion of network equivalence or isomorphism under group action is defined. By harnessing the Leiterspiel algorithm, which calculates the orbits of subsets incrementally in subset size, an efficient enumeration algorithm is presented for enumerating non-isomorphic networks, i.e. canonical representatives of the equivalence classes under this isomorphism, directly. Using this algorithm, millions of non-isomorphic networks are obtained that represent trillions of network coding problems. Then by applying computation tools, exact rate regions of most of them are obtained, leaving the rest with outer bound and simple code achieving inner bounds. Only binary codes are considered here, and binary codes are shown to suffice for most of the networks under consideration. In order to better understand and analyze the huge repository of rate regions, a notion of network hierarchy through embedding and combination operators is created. These operations are defined in a manner such that rate region of the network after each operation can be derived from the rate region of each of the networks involved in the operation. The embedding operations enable us to obtain a list of forbidden network minors for the sufficiency of a class of linear codes. It is shown that for many networks that

scalar binary codes do not suffice, they contain a smaller network, for which scalar binary codes also do not suffice, as a minor under the embedding operations. The combination operations enable us to solve large networks that can be obtained by combining some solvable networks. The integration of both embedding and combination operations is able to generate rate regions for even more networks than we can can be solved directly with combination alone. These operations open a door to many new avenues of network coding research. Some of the pressing future problems for investigation include: I) assessing the coverage of the operators in the space of all problems; II) if necessary, the creation of more powerful combination operations, such as node & edge merge, source & sink merge, etc; III) a notion of forbidden minors which can harness both combination and embedding operators.

Chapter 4: Concluding Remarks and Future Work

This dissertation investigated the enumeration, rate region computation, and hierarchy of multi-source multi-sink hyperedge networks. The network model under consideration includes several special ones such as independent distributed storage systems, multi-level diversity coding systems, and index coding problems. We first considered the relatively simple class of networks, multi-level diversity coding systems in §2.

In §2, we first defined the notion of non-isomorphic MDCS instances, and then developed an enumeration algorithm based on list of Sperner families to obtain thousands of non-isomorphic MDCS instances. Specifically, we investigated the rate regions of 7360 non-isomorphic MDCS instances, which represent 135043 isomorphic instances. Part of our contribution is the explicit construction of superposition, scalar, and vector codes over various fields using the polyhedral inner bounds. Inspired by the notion of a forbidden matroid minor, we introduced several MDCS contraction, deletion, and unification operations and demonstrated that these operations can be used to identify a set of forbidden MDCS instances for the achievability of various classes of codes, in the sense that any extension of such a forbidden MDCS instance will likewise have a rate region for which that class of codes is insufficient. Finally, we demonstrated that we can automate the creation of "human readable proofs" for the rate regions for these 7360 non-isomorphic MDCS instances by posing a suitable optimization problem asking for the smallest cardinality set of Shannon-type inequalities required to step-by-step construct the rate region "by hand".

With lots of MDCS problems solved, we move further on the general multi-source hyperedge networks. We firstly defined a notion of minimal networks, containing no redundant sources, edges, or nodes, whose rate regions directly determine the rate regions of non-minimal networks. Furthermore, since networks related to one another through a permutation of the edge labels are derivable from one another, a notion of network equivalence or isomorphism under group action is defined. By harnessing the Leiterspiel algorithm, which calculates the orbits of subsets incrementally in subset size, an efficient enumeration algorithm is presented for enumerating non-isomorphic networks, i.e. canonical representatives of the equivalence classes under this isomorphism, directly. This is a

more efficient algorithm than the algorithm (§B) extending the one in §2. Using this new developed algorithm, millions of non-isomorphic networks are obtained that represent trillions of network coding problems. Then by applying computation tools, exact rate regions of most of them are obtained, leaving the rest with outer bound and simple code achieving inner bounds. Only binary codes are considered here, and binary codes are shown to suffice for most of the networks under consideration. In order to better understand and analyze the huge repository of rate regions, a notion of network hierarchy through embedding and combination operators is created. These operations are defined in a manner such that rate region of the network after each operation can be derived from the rate region of each of the networks involved in the operation. The embedding operations, as extensions of embedding operations in §2, enable us to obtain a list of forbidden network minors for the sufficiency of a class of linear codes. It is shown that for many networks that scalar binary codes do not suffice, they contain a smaller network, for which scalar binary codes also do not suffice, as a minor under the embedding operations. The combination operations enable us to solve large networks that can be obtained by combining some solvable networks. The integration of both embedding and combination operations is able to generate rate regions for even more networks than we can can be solved directly with combination alone.

Several remaining intriguing questions remain as future work. In terms of the rate region issue, one important question is when the Shannon outer bound is tight for the networks. One may investigate the common structure, if any, between the networks that Shannon outer bound is tight for. An extension of the question on tightness of Shannon outer bound could be if it is tight for all MDCS, or other subclass of networks.

On the hierarchy issue, one important question is whether or not there exists a finite list of forbidden minors for the classes of codes in general networks. That is, one may be interested to see if networks will have similar properties as graphs.

The integration of combination and embedding operations in §3 opens a door to many new avenues of network coding research. Some of the pressing future problems for investigation include: I) assessing the coverage of the operators in the space of all problems; II) if necessary, the creation of more powerful combination operations, such as node & edge merge, source & sink merge, etc; III) a notion of forbidden minors which can harness both combination and embedding operators.

Bibliography

[1] K. P. Hau, "Multilevel diversity coding with independent data streams," Master's thesis, The Chinese University of Hong Kong, June 1995.

[2] R. W. Yeung, *Information Theory and Network Coding.* Springer, 2008.

[3] R. W. Yeung and Z. Zhang, "Distributed source coding for satellite communications," *IEEE Trans. on Information Theory*, vol. 45, no. 4, pp. 1111–1120, 1999.

[4] A. Dimakis, P. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Transactions on Information Theory*, vol. 56, no. 9, pp. 4539–4551, Sept 2010.

[5] C. Tian, "Characterizing the rate region of the (4,3,3) exact-repair regenerating codes," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 5, pp. 967–975, May 2014.

[6] S. Weber, C. Li, and J. M. Walsh, "Rate region for a class of delay mitigating codes and p2p networks," *46th Annual Conference on Information Sciences and Systems*, March 2012.

[7] J. Walsh, S. Weber, and C. Maina, "Optimal rate–delay tradeoffs and delay mitigating codes for multipath routed and network coded networks," *Information Theory, IEEE Transactions on*, vol. 55, no. 12, pp. 5491–5510, Dec 2009.

[8] D. Leong, A. Qureshi, and T. Ho, "On coding for real-time streaming under packet erasures," in *Information Theory Proceedings (ISIT), 2013 IEEE International Symposium on*, July 2013, pp. 1012–1016.

[9] Z. Zhang and R W. Yeung, "On Characterization of Entropy Function via Information Inequalities," *IEEE Transactions on Information Theory*, vol. 44, no. 4, Jul. 1998.

[10] R. Dougherty, C. Freiling, and K. Zeger, "Six new non-Shannon information inequalities," in *IEEE International Symposium on Information Theory (ISIT)*, July 2006, pp. 233–236.

[11] J. G. Oxley, *Matroid Theory.* Oxford University, 2011.

[12] J. R. Roche, "Distributed information storage," Ph.D. dissertation, Standford University, March 1992.

[13] N. Robertson and P. Seymour, "Graph minors. i. excluding a forest," *Journal of Combinatorial Theory, Series B*, vol. 35, no. 1, pp. 39 – 61, 1983.

[14] ——, "Graph minors. xx. wagner's conjecture," *Journal of Combinatorial Theory, Series B*, vol. 92, no. 2, pp. 325 – 357, 2004, special Issue Dedicated to Professor W.T. Tutte.

[15] X. Yan, R. Yeung, and Z. Zhang, "An implicit characterization of the achievable rate region for acyclic multisource multisink network coding," *IEEE Transactions on Information Theory*, vol. 58, no. 9, pp. 5625–5639, Sept 2012.

[16] E. Sperner, "Ein satz über untermengen einer endlichen menge," *Mathematische Zeitschrift (in German)*, vol. 27, no. 1, pp. 544–548, 1928.

[17] A. Betten, M. Braun, H. Fripertinger, A. Kerber, A. Kohnert, and A. Wassermann, *Error-Correcting Linear Codes: Classification by Isometry and Applications*, ser. Algorithms and Computation in Mathematics. Springer Berlin Heidelberg, 2006.

[18] R. Koetter, M. Effros, and M. Medard, "A theory of network equivalence part ii: Multiterminal channels," *Information Theory, IEEE Transactions on*, vol. 60, no. 7, pp. 3709–3732, July 2014.

[19] M. Effros, S. El Rouayheb, and M. Langberg, "An equivalence between network coding and index coding," *Information Theory, IEEE Transactions on*, vol. 61, no. 5, pp. 2478–2487, May 2015.

[20] R. Ahlswede, N. Cai, S.-Y. Li, and R. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, Jul 2000.

[21] T. Chan and A. Grant, "Entropy vectors and network codes," in *IEEE International Symposium on Information Theory (ISIT).*, June 2007, pp. 1586–1590.

[22] C. Li, J. Apte, J. M. Walsh, and S. Weber, "A new computational approach for determining rate regions and optimal codes for coded networks," in *IEEE International Symposium on Network Coding (NetCod)*, Jun 2013, pp. 1 –6.

[23] C. Li, J. M. Walsh, and S. Weber, "Computational approaches for determining rate regions and codes using entropic vector bounds," in *50th Annual Allerton Conference on Communication, Control and Computing*, Oct 2012, pp. 1 –9.

[24] M. Sathiamoorthy, M. Asteris, D. S. Papailiopoulos, A. Dimakis, R. Vadali, S. Chen, and D. Borthakur, "Xoring elephants: Novel erasure codes for big data," *Proceedings of the VLDB Endowment*, vol. 6, no. 5, pp. 325–336, Mar 2013.

[25] C. Tian, "Rate region of the (4, 3, 3) exact-repair regenerating codes," in *IEEE International Symposium on Information Theory (ISIT)*, July 2013, pp. 1426–1430.

[26] J. R. Roche, R. W. Yeung, and K. P. Hau, "Symmetrical multilevel diversity coding," *IEEE Transactions on Information Theory*, vol. 43, no. 3, pp. 1059–1064, 1997.

[27] R. Yeung and Z. Zhang, "On symmetrical multilevel diversity coding," *IEEE Transactions on Information Theory*, vol. 45, no. 2, pp. 609–621, 1999.

[28] S. Thakor, T. Chan, and K. Shum, "Symmetry in distributed storage systems," in *IEEE International Symposium on Information Theory (ISIT)*, July 2013, pp. 1242–1246.

[29] R. W. Yeung, "Multilevel diversity coding with distortion," *IEEE Trans. Information Theory*, vol. 41, pp. 412–422, 1995.

[30] F.-W. Fu and R. Yeung, "On the rate-distortion region for multiple descriptions," *IEEE Transactions on Information Theory*, vol. 48, no. 7, pp. 2012–2021, Jul 2002.

[31] K. P. Hau and R. W. Yeung, "Multilevel diversity coding with three encoders," in *2014 IEEE International Symposium on Information Theory (ISIT)*, June 1997.

[32] S. Mohajer, C. Tian, and S. Diggavi, "Asymmetric multilevel diversity coding and asymmetric gaussian multiple descriptions," *IEEE Transactions on Information Theory*, vol. 56, no. 9, pp. 4367–4387, 2010.

[33] J. Apte, C. Li, and J. Walsh, "Algorithms for computing network coding rate regions via single element extensions of matroids," in *2014 IEEE International Symposium on Information Theory (ISIT)*, June 2014.

[34] S. Fujishige, "Polymatroidal dependence structure of a set of random variables," *Information and Control*, vol. 39, pp. 55–72, 1978.

[35] F. Matúš, "Infinitely Many Information Inequalities," in *IEEE International Symposium on Information Theory (ISIT)*, Jun. 2007, pp. 41–44.

[36] R. Dougherty, C. Freiling, and K. Zeger, "Networks, matroids, and non-shannon information inequalities," *IEEE Transactions on Information Theory*, vol. 53, no. 6, pp. 1949–1969, 2007.

[37] C. Li, J. M. Walsh, and S. Weber, "Software for computing entropy vector region bounds," available at http://www.ece.drexel.edu/walsh/aspitrg/software.html.

[38] D. P. Bertsekas, *Nonlinear Programming.* Athena Scientific, 1999.

[39] R. T. Rockafellar, *Convex Analysis.* Princeton University Press, 1997.

[40] Z. Zhang and R. Yeung, "On characterization of entropy function via information inequalities," *Information Theory, IEEE Transactions on*, vol. 44, no. 4, pp. 1440–1452, Jul 1998.

[41] C. Li, S. Weber, and J. M. Walsh, "Multilevel diversity coding systems: Rate regions, codes, computation, & forbidden minors," *CoRR*, vol. abs/1407.5659, 2014.

[42] J. Apte and J. M. Walsh, "Exploiting symmetry in computing polyhedral bounds on network coding rate regions," in *IEEE International Symposium on Network Coding (NetCod)*, Jun 2015.

[43] J. Apte, , and J. Walsh, "Symmetry in network coding," in *2015 IEEE International Symposium on Information Theory (ISIT)*, June 2015.

[44] J. Geelen, B. Gerards, and G. Whittle, "Solving rota's conjecture," *Notices of the American Mathematical Society*, pp. 736–743, 2014.

[45] G. Gallo, G. Longo, S. Pallottino, and S. Nguyen, "Directed hypergraphs and applications," *Discrete applied mathematics*, vol. 42, no. 2, pp. 177–201, 1993.

[46] B. Schmalz, "t-Designs zu vorgegebener Automorphismengruppe," *Bayreuther Mathematische Schriften*, no. 41, pp. 1–164, 1992, Dissertation, Universität Bayreuth, Bayreuth.

[47] T. Rehn and A. Schürmann, "C++ tools for exploiting polyhedral symmetries," *Lecture Notes in Computer Science*, vol. 6327/2010, 2010.

[48] *GAP – Groups, Algorithms, and Programming, Version 4.7.7*, The GAP Group, 2015. [Online]. Available: http://www.gap-system.org

[49] C. Li, J. M. Walsh, and S. Weber, "ITW 2015 Data: Enumeration and Exact Rate Regions for Networks," available at http://goo.gl/f1spQz.

[50] ——, "NetCod 2015 Data: Exact Rate Regions and Codes for all (K, E) IDSCs with $K, E \in \{2, 3\}$." available at http://goo.gl/WP80CK.

[51] ——, "TransIT 2015 Data: Enumeration and Rate Regions for General Hyperedge Networks," available at https://goo.gl/lyZfD9.

Appendices

Appendix A: Complete proof of Theorem 10

A.1 Converse

We need to prove that for any achievable rate tuple $\mathbf{R} \in \mathcal{R}_*(\mathsf{A})$, we have $\mathbf{R} \in \mathrm{Proj}_{\mathbf{r},\boldsymbol{\omega}}(\overline{\mathrm{con}(\Gamma_N^* \cap \mathcal{L}_{13})} \cap \mathcal{L}_4' \cap \mathcal{L}_5)$.

Pick a point $\mathbf{R} = (H(Y_1), \ldots, H(Y_K), R_1, \ldots, R_{|\mathcal{E}_U|})$ that in $\mathcal{R}_*(\mathsf{A})$. For convenience in comparing with the notations in [15], we let $\omega_s = H(Y_s)$ be the source rate to achieve. Let an arbitrarily small $\epsilon > 0$ be given. Since $\mathbf{R}$ is achievable, for all sufficiently large n, there exists a block n code such that

$$\frac{\log \eta_e}{n} \leq R_e + \epsilon, \ e \in \mathcal{E} \tag{A.1}$$

$$\omega_k \geq \tau_k \geq \omega_k - \epsilon, \ k \in \mathcal{S} \tag{A.2}$$

$$p^{n,err} \leq \epsilon, \tag{A.3}$$

where η_e is the index set of messages sent on edge e and τ_k is the transmitted source rate at source k.

From (A.1), we know for all $e \in E$,

$$\begin{aligned} H(U_e) &\leq \log |\mathcal{U}_e| \\ &= \log(\eta_e) \\ &\leq n(R_e + \epsilon) \end{aligned} \tag{A.4}$$

For all source $k \in [[K]]$, from (A.2) we have

$$n\omega_k \geq \log \lceil q^{n\tau_k} \rceil \geq n(\omega_k - \epsilon). \tag{A.5}$$

Since it is assumed that $\mathbf{R}_{\mathcal{E}}$ is achievable, there exist random variables such that

$$H(\mathbf{Y}_{1:K}) = \sum_{k=1}^{K} H(Y_k) \tag{A.6}$$

$$H(U_e|\mathrm{In}(\mathrm{Tl}(e))) = 0, e \in \mathcal{E} \tag{A.7}$$

Following Lemma 1 in [15] with a light change in the proof that considers the input of a sink t including some other sources that not in $\beta(t)$, we have

$$H(\beta(t)|\mathbf{U}_{\mathrm{In}(t)}) \leq n\phi_t(n,\epsilon), \tag{A.8}$$

where $\phi_t(n,\epsilon)$ has the following properties:

1. $\phi_t(n,\epsilon)$ is bounded;
2. $\phi_t(n,\epsilon) \to 0$ as $\epsilon \to 0$ and $\epsilon \to 0$;
3. $\phi_t(n,\epsilon)$ is monotonically decreasing with increase of n and decrease of ϵ.

From equations (A.4) – (A.8) we get the existence of entropic vector such that

$$h_{\mathbf{Y}_{1:K}} = \sum_{k=1}^{K} h_{Y_k} \tag{A.9}$$

$$h_{U_e|\mathrm{In}(\mathrm{Tl}(e))} = 0, e \in \mathcal{E} \tag{A.10}$$

$$h_{Y_k} \geq n(\omega_k - \epsilon), k \in [[K]] \tag{A.11}$$

$$h_{\beta(t)|\mathbf{U}_{\mathrm{In}(t)}} \leq n\phi_t(n,\epsilon), \forall t \in \mathcal{T} \tag{A.12}$$

$$h_{U_e} \leq n(R_e + \epsilon), e \in \mathcal{E} \tag{A.13}$$

Therefore, $\mathcal{L}_1, \mathcal{L}_2$ are satisfied. Now define the following two regions in Γ_N^* that depend on n:

$$\mathcal{L}_{4,\epsilon}^n = \{\mathbf{h} \in \Gamma_N^* : h_{U_e} \leq n(R_e + \epsilon)\}, \tag{A.14}$$

$$\mathcal{L}_{5,\epsilon}^n = \{\mathbf{h} \in \Gamma_N^* : h_{\mathbf{Y}_{1:k}|\mathbf{U}_{\mathrm{In}(t)}} \leq n\phi_t(n,\epsilon), \forall t \in \mathcal{T}\}. \tag{A.15}$$

Then from (A.9)– (A.13) we see that there exists

$$\mathbf{h} \in \Gamma_N^* \tag{A.16}$$

such that

$$\mathbf{h} \in \mathcal{L}_{13} \cap \mathcal{L}_{4,\epsilon}^{n} \cap \mathcal{L}_{5,\epsilon}^{n} \tag{A.17}$$

and $\forall s \in \mathcal{S}$

$$h_{Y_s} \geq n(\omega_s - \epsilon), \tag{A.18}$$

i.e.

$$\frac{h_{Y_s}}{n} \geq \omega_s - \epsilon. \tag{A.19}$$

From (A.16) and (A.17) we obtain

$$\mathbf{h} \in \Gamma_N^* \cap \mathcal{L}_{12} \cap \mathcal{L}_{4,\epsilon}^{n} \cap \mathcal{L}_{5,\epsilon}^{n}. \tag{A.20}$$

Since $\Gamma_N^* \cap \mathcal{L}_{13}$ contains the origin, we get that

$$n^{-1}\mathbf{h} \in \overline{\mathrm{con}(\Gamma_N^* \cap \mathcal{L}_{12})} \cap \mathcal{L}_{4,\epsilon} \cap \mathcal{L}_{5,\epsilon}, \tag{A.21}$$

where

$$\mathcal{L}_{4,\epsilon} = \{\mathbf{h} \in \Gamma_N^* : h_{U_e} \leq R_e\} \tag{A.22}$$

and

$$\mathcal{L}_{5,\epsilon} = \{\mathbf{h} \in \Gamma_N^* : h_{\beta(t)|\mathbf{U}_{\mathrm{In}(t)}} \leq n\phi_t(n,\epsilon), t \in \mathcal{D}\}. \tag{A.23}$$

For all n and ϵ, define the set

$$\mathcal{B}^{(n,\epsilon)} = \{\mathbf{h} \in \overline{\mathrm{con}(\Gamma_N^* \cap \mathcal{L}_{12})} \cap \mathcal{L}_{4,\epsilon} \cap \mathcal{L}_{5,\epsilon} : \omega_k \geq h_{Y_k} \geq \omega_k - \epsilon, \forall k \in \mathcal{S}\}. \tag{A.24}$$

The fact that $\mathcal{B}^{(n,\epsilon)}$ is closed and bounded follows immediately from a similar proof in [15]. Then, we conclude that $\mathcal{B}^{(n,\epsilon)}$ is compact.

Now from the fact that $\phi_t(n,\epsilon)$ is monotonically decreasing in both n and ϵ, so for all $\epsilon' < \epsilon$ and n,

$$\mathcal{B}^{(n+1,\epsilon)} \subset \mathcal{B}^{(n,\epsilon)} \tag{A.25}$$

and

$$\mathcal{B}^{(n,\epsilon')} \subset \mathcal{B}^{(n,\epsilon)} \tag{A.26}$$

Note that from (A.18) and (A.21) we see that for any $\epsilon > 0$, $\mathcal{B}^{(n,\epsilon)}$ is nonempty. Since $\mathcal{B}^{(n,\epsilon)}$ is

compact and nonempty,

$$\lim_{\epsilon\to 0}\lim_{n\to\infty}\mathcal{B}^{(n,\epsilon)} = \bigcap_{\epsilon}^{0}\bigcap_{n=1}^{\infty}\mathcal{B}^{(n,\epsilon)} \tag{A.27}$$

is also nonempty and compact, which equals to

$$\{\mathbf{h}\in\overline{\mathrm{con}(\Gamma_N^*\cap\mathcal{L}_{12})}\cap\mathcal{L}_4\cap\mathcal{L}_5 : h_{Y_k}=\omega_k, \forall k\in\mathcal{S}\}. \tag{A.28}$$

Hence, we conclude that

$$\mathbf{R}\in\mathrm{Proj}_{H(Y_k),k\in\mathcal{S},U_e,e\in\mathcal{E}}(\overline{\mathrm{con}(\Gamma_N^*\cap\mathcal{L}_{13})}\cap\mathcal{L}_4\cap\mathcal{L}_5). \tag{A.29}$$

A.2 Achievability

We need to prove that for any point $\mathbf{R}\in\mathrm{Proj}_{H(Y_k),k\in\mathcal{S},U_e,e\in\mathcal{E}}(\overline{\mathrm{con}(\Gamma_N^*\cap\mathcal{L}_{13})}\cap\mathcal{L}_4\cap\mathcal{L}_5)$, there exists a code such that this rate is achievable.

Similar as Lemma 3 in [15], if we define

$$\mathsf{A}_1=\overline{\mathrm{con}(\Gamma_N^*\cap\mathcal{L}_{13})} \tag{A.30}$$

and

$$\mathsf{A}_2=\overline{D(\Gamma_N^*\cap\mathcal{L}_{13})}, \tag{A.31}$$

where $D(\mathsf{A}')=\{\alpha\mathbf{h}:\mathbf{h}\in\mathsf{A}'\ \&\ 0\le\alpha\le 1\}$, we have

$$\mathsf{A}_1=\mathsf{A}_2. \tag{A.32}$$

The proof of (A.32) is identical to the proof of Lemma 3 in [15] except that we only have $\mathcal{L}_{13}$ but [15] has $\mathcal{L}_{123}$.

Let $\mathbf{R}$ be the point picked, we also have $\mathbf{R}\in\mathrm{Proj}_{H(Y_k)k\in\mathcal{S},U_e,e\in\mathcal{E}}(\overline{D(\Gamma_N^*\cap\mathcal{L}_{13})}\cap\mathcal{L}_4\cap\mathcal{L}_5)$, then there exists an $\mathbf{h}\in\overline{D(\Gamma_N^*\cap\mathcal{L}_{13})}\cap\mathcal{L}_4\cap\mathcal{L}_5$ such that $\mathbf{R}=\mathrm{Proj}_{H(Y_k),k\in\mathcal{S},U_e,e\in\mathcal{E}}(\mathbf{h})$. Furthermore, there exists an entropic vector $\hat{\mathbf{h}}\in\Gamma_N^*\cap\mathcal{L}_{13}$ and an α such that $\mathbf{h}=\alpha\hat{\mathbf{h}}$.

Since $\hat{\mathbf{h}}\in\Gamma_N^*\cap\mathcal{L}_{13}$, there exists a collection of random variables $\mathcal{N}=\{Y_k,k\in[[K]]\}\cup(U_e,e\in$

$\mathcal{E}\}, |\mathcal{N}| = N$ such that

$$\alpha\hat{H}(Y_s) = \omega_s, s \in \mathcal{S} \tag{A.33}$$

$$\hat{H}(\mathbf{Y}_{1:K}) = \sum_{k\in\{1,\ldots,K\}} \hat{H}(Y_k) \tag{A.34}$$

$$\hat{H}(U_e|\mathrm{In}(\mathrm{Tl}(e))) = 0, e \in \mathcal{E} \tag{A.35}$$

where $\omega_s = H(Y_s)$ is the source rate to achieve at source s and ϵ is an arbitrarily small positive number.

Furthermore, since $\mathbf{h} \in \mathcal{L}_4 \cap \mathcal{L}_5$ and we only need to show h is asymptotically achievable, we have

$$\alpha\alpha\hat{H}(U_e) \leq R_e + \mu, e \in \mathcal{E} \tag{A.36}$$

$$\alpha H(\beta(t)|\mathbf{U}_{\mathrm{In}(t)}) \leq \gamma, t \in \mathcal{T} \tag{A.37}$$

where μ and γ are arbitrarily small positive numbers.

Our next step is to show that the rate vector $\mathbf{R}' = (\omega_s, R_e + \mu, e \in \mathcal{E})$ is achievable. Then as $\epsilon \to 0$, we know that $\mathbf{R}$ is achievable.

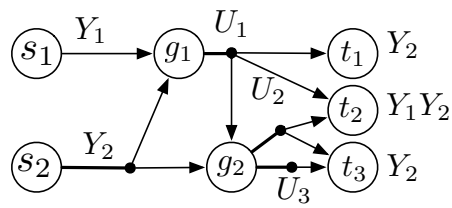
Use the similar code construction and performance analysis as in the proof of achievability in [15], we can show that $(H(Y_k), k \in \mathcal{S})$ is achievable if we set $\mathbf{R}'$ as the capacities on edges. Encoding functions at sources are simply identity functions. Equivalently, the similar construction shows that $\mathbf{R}'$ is achievable. Therefore, $\mathbf{R}$ is achievable as $\epsilon \to 0$.

Appendix B: Enumeration of Networks Based on Sperner Families

This algorithm is an extension of our previous enumeration for MDCS problems [41] based on list of Sperner families, with consideration of topologies and more minimality constraints. Compared with the algorithm introduced in §3.4, this algorithm first enumerates networks with partial isomorphisms but not all, and then remove them. The order among network variables due to the acyclic topology graph helps in removing isomorphisms at the very beginning. Though it has higher complexity than the algorithm in §3.4, it worth presenting it for comparison.

At first, we introduce the data structure used in representing a network for this enumeration. Note that a $(K, |\mathcal{E}_U|)$ network instance, mainly specifies the encoding mappings and decoding mappings. Since the sources are assumed to be independent, the encoding mappings, or the topology except the decoders, can be represented as a $|\mathcal{E}_U| \times N$ matrix top, where $N = K + |\mathcal{E}_U| = |\mathcal{E}|$. The row indices of the matrix correspond to the edge variables in $\mathcal{E}_U$ and the column indices correspond to both source and edge variables. $top_{i,j} = 1$ if the variable corresponding to j-th column is an input variable of $\mathrm{Tl}(E_i)$. At the decoders side, there are two relations: mappings between edges (potentially including sources) and decoders, and mappings between decoders and their demanded sources. One representation of the decoding mappings of a network instance is to list the accesses and demands of each decoder. Since $\mathrm{In}(t) \subseteq \mathcal{E}, \forall t \in \mathcal{T}$, one can represent $\mathrm{In}(t)$ using a N-bit vector or a corresponding integer value, where the sources and edges are indexed by $1, \ldots, N$ and the entries of the vector from left to right are mapped to $E_{|\mathcal{E}_U|}, \ldots, E_1, Y_K, \ldots, Y_1$. Similarly, the demands $\beta(t)$ can be coded as a K-bit vector with Y_1 corresponds to the last bit. We can define a bijection $\theta : n \leftrightarrow find(bin(n) \neq 0)$ between positions of indicators (non-zero values) in binary sequence and corresponding integer value. For example, $\theta(6) = \{2, 3\}$ because the binary sequence of 6 is (110) and the non-zero positions are $2, 3$ from the right side. Furthermore, we denote that $\mathbf{Y}_{\theta(n)} = \{Y_i, i \in \theta(n)\}$. With this encoding, the decoding mappings of a network instance can be easily represented by a matrix with $2^K - 1$ rows, where the row indices represent the demands of decoders and entries are integers representing the accesses of each decoder. An all-zero vector at i-th row means there is no decoder that only requires $\mathbf{Y}_{\theta(i)}$. Each row may include some zeros to

$$top = \begin{array}{c|ccccc} & Y_1 & Y_2 & U_1 & U_2 & U_3 \\ \hline U_1 & 1 & 1 & 0 & 0 & 0 \\ U_2 & 0 & 1 & 1 & 0 & 0 \\ U_3 & 0 & 1 & 1 & 0 & 0 \end{array}$$



$$dec = \begin{array}{c|cc} Y_1 & 0 & 0 \\ Y_2 & 4 & 24 \\ Y_1Y_2 & 0 & 12 \end{array}$$

Figure B.1: An example of $(2,3)$ *network with the topology and decoding matrices.*

make them the same length. Therefore, an alternative representation for a network instance is the two matrices $\{top, dec\}$, which will be used as data structure in the enumeration algorithms.

To better understand the representation of a network, we give a $(2,3)$ network instance in Fig. B.1 as an example.

Example 10: For the small network in Fig. B.1, the topology matrix $top = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}$ with row indices correspond to $U_1, \ldots, U_3$ and column indices correspond to $Y_1, Y_2, U_1, \ldots, U_3$, because U_1 is a function of Y_1, Y_2, while both U_2 and U_3 are functions of Y_2, U_1. The decoding configuration matrix for this $(2,3)$ network is $dec = \begin{bmatrix} 0 & 0 \\ 4 & 24 \\ 0 & 12 \end{bmatrix}$, where the three row vectors indicate that there is no decoder merely demanding source Y_1 ($\theta(1) = 1$), two decoders that have access to U_1 ($(00100)_2 = 4$) and U_2, U_3 ($(11000)_2 = 24$) demand Y_2 ($\theta(2) = 2$), respectively. One decoder that has access to $\{E_1, E_2\}$ ($(01100)_2 = 12$) demands Y_1, Y_2 ($\theta(3) = \{1, 2\}$).

After introducing the matrix representation of a network, we now discuss the enumeration of networks. Since the isomorphism merely permutes the sources and/or encoders, to study all possible network instances, it suffices to consider one representative in each isomorphism class, i.e., only consider non-isomorphic network instances. We will try to enumerate all non-isomorphic network instances for a give size.

One straightforward method to do this is to remove isomorphisms from the list of all network instances, which may have high complexity. The following Proposition, as a corollary of Proposition 19.1 in [2], will help in reducing the complexity because it excludes a decent number of isomorphisms among source and edge variables.

Proposition 1: Suppose a network $\mathsf{A} = (\mathcal{S}, \mathcal{G}, \mathcal{T}, \mathcal{E}, (\beta(t), t \in \mathcal{T}))$ is a directed acyclic hyper graph as defined in §3.2. It is possible to order the variables of A in a sequence such that if there is a directed path from node $i \in \mathrm{Hd}(e_1)$ to $\mathrm{Tl}(e_2)$ or $\mathrm{Tl}(e_2) \in \mathrm{Hd}(e_1)$, then edge e_1 appears before e_2 in the sequence.

Proof. As defined in §3.2, a hyperedge is an ordered pair of one node with a subset of the other

nodes. One can convert the hypergraph A with such hyperedges into a directed acyclic graph A' by replacing every hyperedge with a collections of edges and an auxiliary node. For instance, a hyperedge $(\mathrm{Tl}(e), \mathrm{Hd}(e))$ will be replaced by edges $\{(\mathrm{Tl}(e), \mathrm{Tl}'(e)), (\mathrm{Tl}'(e), i), i \in \mathrm{Hd}(e)\}$, where $\mathrm{Tl}'(e)$ is the added auxiliary node that associated with e. After conversion, the graph is still acyclic, because the construction does not introduce cycles. According to Proposition 19.1 of [2], there exists an order between the nodes in A' such that if there exists a path from node 1 to node 2, node 1 will appear first in the ordered sequence. We know $\mathrm{Tl}'(e), e \in \mathcal{E}$ are in such an order and each of them is associated with a hyperedge in A. Therefore, the proposition holds. □

Based on the proposition above, we can enumerate all network topologies (except the sinks) of a given size by defining the edge variables (since sources are independent and we can initially have all source variables) one by one in a manner that the newly defined variable is a function of a subset of the existing variables. This dependence relation determines the input of the tail of the newly defined edge. Edges having same input means they are outgoing edges of a same node. An algorithm to enumerate all non-isomorphic topologies $\mathcal{Q}$ is shown in Algorithm 7. This algorithm, using matrix representation, recursively defines the edge variables and at the same time considers the minimality constraints (**C1**) and leaves the other constraints verified later in decoder configurations. Note that (**C5**) is trivially satisfied using the coding of edges, since the matrix does not specify node and all edges have same dependent input will be automatically viewed as outgoing edges from the same node.

We now discuss the configuration of decoders for each enumerated topologies from Algorithm 7 and further obtain the non-isomorphic network instances. We will continue to use the matrix representation for the decoder configurations in the enumeration. Observe that the access of decoders with the same demands must be a *Sperner family* of the edge set $\mathcal{E}$ including outgoing edges from sources, as required by condition (**C12**). A Sperner family of $\mathcal{E}$, sometimes also called an *independent system* or a *clutter*, is a collection of subsets of $\mathcal{E}$ such that no element is contained in another. Since the sinks demanding a certain subset of the sources can be empty, the empty set, or the all-zero row vector, is used to represent the mapping between edges and such "decoders".

An algorithm to enumerate non-isomorphic minimal network instances, by removing isomorphism from a list of network instances with some isomorphism, is given in Algorithm 8. We consider the subsets of sources as demands of decoders in a binary-count order, or a quasi order, where the supersets of a subset are always considered after the subset itself. The enumeration process works as follows.

```
Input: Number of edge variables |ℰ_U|, Number of sources K
Output: All network topologies 𝒬, with partial isomorphism
Initialization: top = zeros(|ℰ_U|, K + |ℰ_U|), mapping π(Y_i) = i, i ∈ {1, ..., K} and
π(E_j) = K + j, j ∈ {1, ..., |ℰ_U|}, defined variables ℬ = {Y_1, ..., Y_K}, 𝒬' = ∅, 𝒬 = ∅;
for every 𝒜 ∈ 2^𝒮 do
    top(1, π(𝒜)) = 1;
    𝒬' = 𝒬' ∪ top;
    top(1, :) = 0;
end
ℬ = ℬ ∪ E_1;
for i = 2 : |ℰ_U| do
    for every top ∈ 𝒬 do
        if i ≠ |ℰ_U| then
            for every 𝒜 ∈ 2^ℬ do
                top(i, π(𝒜)) = 1;
                𝒬' = 𝒬' ∪ top;
                top(i, :) = 0;
            end
        end
        else
            for every 𝒜 ∈ 2^ℬ do
                if top(|ℰ_U|, π(𝒜)) satisfies (C1,C5) then
                    top(i, π(𝒜)) = 1;
                    𝒬' = 𝒬' ∪ top;
                    top(i, :) = 0;
                end
            end
        end
    end
end
Removal isomorphisms: 𝒬 = isoRemoval(𝒬');
```

Algorithm 7: Enumerate topologies for $(K, |\mathcal{E}_U|)$ network

1. List all Sperner families, Sper($\mathcal{E}$) of the edge set $\mathcal{E}$ (required by (**C12**));

2. Obtain non-isomorphic topologies from Algorithm 7;

3. For each of the obtained topology, consider possible decoder configurations. All Sperner families that do not violate (**C2, C10**), including the empty set, are possible configurations for decoders demanding first source, except that the empty set cannot be included when $K = 1$, from (**C6, C10**);

4. Suppose we have configurations for up to the decoders demanding $\mathbf{Y}_{\theta(i-1)}$. Now we will augment the existing partially-configured network instances with configurations for the decoders merely demanding $\mathbf{Y}_{\theta(i)}$. We consider all the existing instances one by one. Take one of them as an example. Since the decoders demanding $\mathbf{Y}_{\theta(i)}$ cannot have same access as the existing decoders (required by (**C11**)), we should remove the Sperner families containing at least one element that is already selected in the existing decoders. Furthermore, we have to consider (**C12, C13**)) to remove those introduced redundancies. The remaining Sperner families are possible configurations for demanders of $\mathbf{Y}_{\theta(i)}$. Repeat this step until to the last stage.

5. At last stage, the demanders of $\mathbf{X}_{1:K}$ are considered. At this step, we need to verify more minimality conditions. If a potential configuration passes the verifications of conditions (**C3-C14**), a minimal network instance has been obtained.

6. After step 4), all network instances are obtained with some (but not all) isomorphism. Then we remove isomorphism by keeping one instance in every isomorphism class, where the network instances can be obtained by permuting sources and/or encoders from one another, and remove the others in the isomorphism class from the list of all network instances.

Input: Number of edge variables $|\mathcal{E}_U|$, Number of sources K
Output: All non-isomorphic network instances $\mathcal{M}$, all network instances with some isomorphism $\mathcal{M}'$

Initialization: List all integer-coded Sperner families $\text{Sper}(\mathcal{E}_U)$ of $\mathcal{E}_U$, $pool = \text{Sper}(\mathcal{E})$, all non-isomorphic topologies $\mathcal{Q}$, initial decoding matrix $dec_0 = zeros(2^K - 1, size(pool, 2))$, $\mathcal{M}' = \emptyset$, $\mathcal{M} = \emptyset$;

```
for every top ∈ Q do
    A' = {top, dec_0};
    for i = 1 : (2^K − 1) do
        A'' = A', A' = ∅;
        for every A ∈ A'' do
            if i ≠ 2^K − 1 then
                pool = Sper(E) \ {I ∈ Sper(E) | Aug(A, I, i) violates (C11)-(C13)};
            end
            else
                pool = Sper(E) \ {I ∈ Sper(E) | Aug(A, I, i) violates (C3)-(C14)};
            end
            A = Aug(A, pool, i), A' = A' ∪ A;
        end
    end
    M' = M' ∪ A';
end
Remove isomorphisms: M = isoRemoval(M');
where the Aug is defined as follows
Function:  A = Aug(A, pool, i)
A = ∅;
for every I ∈ pool do
    dec(i, :) = I;
    A = A ∪ {top, dec};
    dec(i, :) = 0;
end
```

Algorithm 8: Enumerate isomorphic and non-isomorphic $(K, |\mathcal{E}_U|)$ network instances

Vitae

Congduan Li

Webpage: http://www.pages.drexel.edu/~cl698/

Education

- Ph.D. Electrical & Computer Engineering (ECE), Dec, 2015 GPA: 3.93/ 4.00
  Drexel University, Philadelphia, Pennsylvania, US
  Adviser: John MacLaren Walsh & Steven Weber
- M. S. Electrical Engineering (EE), May, 2011 GPA: 4.00/ 4.00
  Northern Arizona University (NAU), Flagstaff, Arizona, US
  Adviser: Sheryl Howard & Paul Flikkema
- B. S. Electrical Engineering (EE), June, 2008 Overall GPA: 3.50/ 4.00, Major GPA: 3.70/ 4.00
  University of Science & Technology Beijing, (USTB), Beijing, China

Publications

1. **Congduan Li**, Steven Weber, and John M. Walsh, "Computer aided proofs for rate regions of independent distributed source coding problems", *The 2015 IEEE International Symposium on Network Coding, Jun 2015.*
2. **Congduan Li**, Steven Weber, and John M. Walsh, "Network embedding operations preserving the insufficiency of linear network codes", *52nd Annual Allerton Conference on Communication, Control, and Computing, pp.1333-1340, Sept. 30-Oct. 3, 2014*
3. Jayant Apte, **Congduan Li**, John M. Walsh, "Algorithms for computing network coding rate regions via single element extensions of matroids", *IEEE International Symposium on Information Theory (ISIT), pp.2306-2310, June 29-July 4, 2014*
4. Jayant Apte, **Congduan Li**, John M. Walsh and Steven Weber, "Exact repair problems with multiple sources", *48th Annual Conference on Information Sciences and Systems, pp.1-6, 19-21 March, 2014*
5. **Congduan Li**, John M. Walsh and Steven Weber, "Matroid bounds on region of entropic vectors", *51st Annual Allerton Conference on Communication, Control, and Computing, pp.796-803, 2-4 Oct., 2013*
6. **Congduan Li**, Jayant Apte, John M. Walsh and Steven Weber, "A new computational approach for determining rate regions and optimal codes for coded networks", *The 2013 IEEE International Symposium on Network Coding (NetCod), Calgary, Canada, Jun 7-9, 2013.*
7. **Congduan Li**, John Walsh and Steven Weber, "A computational approach for determining rate regions and codes using entropic vector bounds," *50th Annual Allerton Conference on Communication, Control, and Computing, pp.1580-1587, 1-5 Oct., 2012*
8. Steven Weber, **Congduan Li** and John Walsh, "Rate region for a class of delay mitigating codes and P2P networks", *46th Annual Conference on Information Sciences and Systems, pp.1-6, 21-23 March, 2012. Invited.*
9. **Congduan Li**, Paul G. Flikkema and Sheryl L. Howard, "Progressive coding and iterative source-channel decoding in wireless data gathering networks," *IEEE Global Telecommunications Conference, pp.1-6, 5-9 Dec., 2011*

Submitted and In Preparation

10. **Congduan Li**, Steven Weber, and John M. Walsh, "Multilevel diversity coding systems: rate regions, codes, computation, & forbidden minors", *IEEE Transactions on Information Theory, 2014,* submitted. Available: http://arxiv.org/abs/1407.5659
11. **Congduan Li**, Steven Weber, and John M. Walsh, "Network combination operations preserving the sufficiency of linear network codes", *2015 IEEE Information Theory Workshop (ITW),* submitted.
12. **Congduan Li**, Steven Weber, and John M. Walsh, "Network operations: rate regions and sufficiency of codes", *IEEE Transactions on Information Theory, 2015,* submitted.

Work Experience

Visiting Researcher, Institute of Network Coding, Chinese University of Hong Kong (CUHK), Nov, Dec, 2014

Research/ Teaching Assistant, ECE Department, Drexel University, Summer 2011-Now

- Research on calculating rate regions and associated achieving codes for multi-terminal networks
- TA Courses: Probability, Random variables and Stochastic processes, Intro to Digital Signal Processing
- Funding: NSF CCF-1016588 & NSF CCF-1421828

Research/ Teaching Assistant, EECS Department, NAU, Fall 2009-Spring 2011

- Research on channel and source coding in wireless sensor networks
- TA Courses: Digital Logic, Electrical Engineering I, Microprocessors
- Funding: NSF ECCS-0824322

Summer Intern, ZTE Co., LTD., Beijing, Summer 2007

Teaching Assistant, College of Material Science, USTB, 2007

- TA Course: C++ Language

Honors and Awards

- Daie Endowment Scholarship, EECS Dept, NAU, 2010, 2011
- Outstanding Graduate of USTB, 2008
- Renmin/ China National Scholarship, 2005, 2006, 2007
- Excellent Student in Social Practice, 2005, 2006