2018

Network coding for function computation

Ardhendu Shekhar Tripathy
Iowa State University

Network coding for function computation

by

Ardhendu Shekhar Tripathy

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Electrical and Computer Engineering

Program of Study Committee:
Aditya Ramamoorthy, Major Professor
Nicola Elia
Chinmay Hegde
Sung-Yell Song
Zhengdao Wang

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this dissertation. The Graduate College will ensure this dissertation is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University

Ames, Iowa

2018

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

ACKNOWLEDGEMENTS

I would like to thank my advisor Prof. Aditya Ramamoorthy for his steadfast help and mentorship throughout my Ph.D. degree. I have learned a lot from him. How to formulate a good problem, ways to identify and obtain the parts necessary for a particular approach to the problem, and how to salvage something useful from failed attempts or unruly intuition — all of these have surfaced more than a few times during my research and I have been aided greatly by my advisor's guidance in these. His technical knowledge has steered me in the choice of subjects I have studied here, many ideas from which have found application in this dissertation. He has also helped me improve my writing and presentation skills. That has helped me effectively communicate my work in seminars at Iowa State, as well as national and international meetings.

I have also greatly enjoyed the interactions I have had with my committee members. Prof. Zhengdao Wang has always been enthusiastic about my research and has given me great feedback. Some of the directions explored here were prompted by his questions. I have enjoyed attending many of the courses he has taught here. Prof. Chinmay Hegde has been a friend and a mentor to me. I have always felt welcome to discuss my research and various other topics in his office. He has been very interested in my research and has always encouraged me. Prof. Nicola Elia has been a careful listener and has asked questions that have improved my understanding of the problems I have worked on. Prof. Sung-Yell Song has taught me much of the new mathematics that I have learned at Iowa State University. He has been interested and inquisitive about the way it has found applications in engineering.

Some other faculty with whom I have often talked about academic matters separate from my dissertation research are Prof. Namrata Vaswani and Prof. Yongxin Chen. They have always provided me good advice. Prof. Pavan Aduri, Prof. Jin Tian and Prof. Leslie Hogben have been patient in answering the many questions that I have asked them.

The work in this dissertation was funded in part by the National Science Foundation (NSF) under the following grants — CCF-1149860, CCF-1320416 and CCF-1718470. The support of the NSF is gratefully acknowledged.

Through the years in Coover Hall, I have enjoyed the company and discussions I have had with Hooshang Ghasemi, Li Tang and Konstantinos Konstantinidis. I am also glad to have known a wider group of graduate students who have been eager to listen and converse. Han Guo, Seyedehsara Nayer, Praneeth Narayanamurthy, Zhengyu Chen, Vahid Daneshpajooh, Mohammadreza Soltani, Gauri Jagatap, Thanh Nguyen, Viraj Shah, Songtao Lu, Pan Zhong, Qi Xiao, Rahul Singh, and the many others who have been involved in the Data Science Reading Group — I have greatly enjoyed it and I hope it continues. Apurba Das was willing to discuss research patiently, in spite of our very different research areas. I also thank Mohammadreza, Praneeth and Songtao for their help and support in matters outside of academics. The student workers and staff in the ECpE main office and student services office have been helpful and pleasant in all my requests to them. I owe a debt to Cover and Thomas, the 1991 edition of their textbook inspired me to do research in this area.

Outside of Coover Hall, there have been a group of close friends who have helped me weather the tides of graduate student life. They are Srijita Patra, Siva Konduri, Priyanka Bolel, Rhitajit Sarkar, Neelam Prabhu-Gaunkar, Viksit Kumar, Sophiya Das, Jayaprakash Selvaraj, Priyam Rastogi, Sai Pushpak, Sangeetha N.S., and Aneesh Rajendran. I have good memories from the times we spent together.

This dissertation would not have been possible without the love and affection of my girlfriend Niranjana Krishnan and my family in India. My parents and sister have always believed in my abilities and supported me. I owe to them and Niranjana my biggest thanks.

ABSTRACT

Many applications such as parallel processing, distributed data analytics and sensor networks often need to compute functions of data that are observed in a distributed manner over a network. A network can be modeled as a directed graph, each vertex of which denotes a node that can carry out computations and communicate with its neighbors. The edges of the graph denote one-way noiseless communication links. A subset of nodes - called sources - observe independent messages, and a possibly different subset of nodes - called terminals - wish to compute a particular demand function of the messages. The information transmitted on the edges are specified by a set of functions, one for each edge; this set of functions is called a network code. We are interested in network codes that allow each terminal to compute the demanded function with zero-error.

In the first part of this thesis, we assume that the message random variables are independent and uniformly distributed over a finite field. The demand function is set to be the finite field sum of all the messages observed in the network. A valid network code for this *sum-network* problem allows each terminal to compute the sum, and has an associated computation rate. We wish to find the best possible computation rate for a given sum-network; this value is called its *computation capacity*. Finding the computation capacity of a sum-network is known to be a difficult problem. Here we are able to evaluate it for certain systematically constructed sum-network problem instances. The construction procedure uses incidence structures, whose combinatorial properties allow us to be able to evaluate the computation capacity of the constructed sum-networks. An important aspect of the problem that we uncover is the strong dependence of the computation capacity on the finite field over which the sum is to computed. This is shown by a sum-network, whose computation capacity is 1 over a finite field and close to 0 over a different finite field. We also construct sum-networks whose computation capacity can take on arbitrarily many different values over different finite field alphabets.

In the second part of the thesis, we focus on a particular directed acyclic network that has four nodes and four edges. It is the simplest instance of a network that does not have a tree structure. Three of the nodes are sources that observe independent messages that are uniformly distributed over a finite discrete alphabet. The fourth node is a terminal which wants to compute a demand function of the three messages. The demand function is an arbitrary discrete-valued function. We focus on network codes that have different rates on each of the four edges, thus we have a rate tuple associated with every valid network code. The collection of rate tuples for all valid network codes form a *rate region*, and we describe a procedure to obtain an outer bound to this rate region. We illustrate our approach through different example demand functions. When the demand function is the finite field sum over $GF(2)$, we give a network code whose rate tuple matches the outer bound.

CHAPTER 1. INTRODUCTION

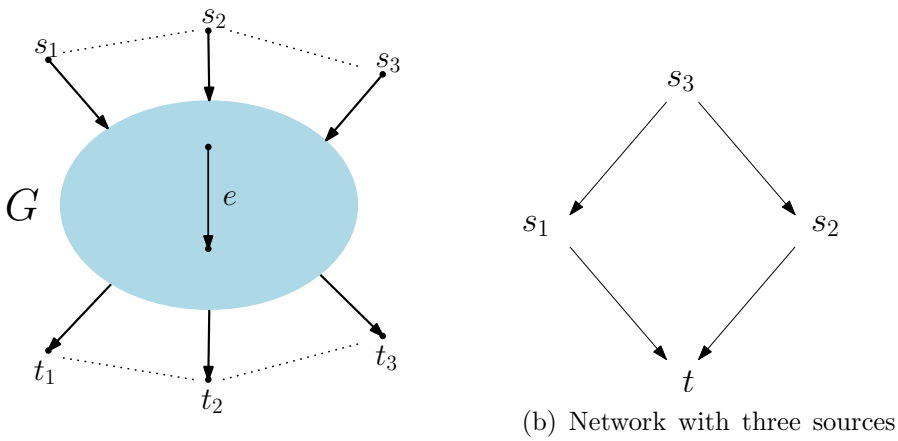
The unprecedented scale of data generation, storage and analysis in present times is a well-known phenomenon. Coupled with the formation of networks of such data nodes, this situation has posed ample engineering challenges and opened new possibilities. Complementary to speeding up data processing and connectivity speeds, one could consider the following question: *how efficiently can we combine data over a network?*

Consider a network of temperature sensors in a centrally air-conditioned building. The control unit would take in the temperature readings, i.e., the data, and perform some computation on them to decide whether to heat or cool the building. Thus, interpreting and making useful conclusions of the data can be thought of as computing certain functions on the data. Finding optimal procedures and fundamental limits on how efficiently this can be done is important for increasingly large datasets.

1.1 Function computation: Sum-networks

Sum-networks are a class of function computation problems over networks. We represent a communication network by a directed acyclic graph, as shown in Figure 1.1a. The structure of the graph is a part of the problem description.

The two components of the graph, i.e., its vertices and edges denote nodes in a network and the communication links between them, respectively. A subset of the nodes called the sources, denoted as s_1, s_2 and s_3, observe independent data, which are assumed to be elements of a finite field. A different subset of nodes called the terminals, are denoted as t_1, t_2, t_3 and each of them wants to compute with zero error the finite field sum of all the data observed at the source nodes. The edges in the network, shown as e, are one-way communication links that are error-free. The objective is to come up with a scheme, called a *network code*, that specifies what descriptions are to be transmitted



(a) Network schematic

(b) Network with three sources and one terminal

Figure 1.1: Communication networks represented as directed acyclic graphs.

over each edge in the graph, such that every terminal is satisfied. We can associate a *computation rate* with every network code that solves a sum-network; this characterizes the communication load on the network. The least upper bound to the best possible rate that can be achieved is called the computation capacity of that sum-network. Sum-networks are useful to look at because of their connections to other general classes of communication problems over networks.

We have constructed several infinite families of sum-networks using a systematic procedure on combinatorial block designs and evaluated their computation capacity analytically. As a consequence, we have shown that in general, the computation capacity of a sum-network changes if the underlying finite field alphabet of the data is changed. This is possible because some of the intermediate computations in the network become more efficient over certain finite fields. The structure of the network plays an important role in this.

For a given a rational number, there is construction procedure known in the literature that constructs a sum-network with that rational number as its computation capacity. We are able to do the same using our procedure, however the sum-networks constructed by our procedure are smaller in comparison.

1.2 Function computation: Using variable-length network codes

Consider now a more specific network shown in Figure 1.1b. Suppose each source observes a data value that is either 0 or 1, and the terminal wishes to compute with zero error the sum, over the real numbers, of the three data values. This is the simplest non-tree network structure, and its computation capacity is known to be $\log_6 4 \approx 0.77$ in the standard network code framework. This value is obtained after counting the necessary and sufficient number of distinct messages that are transmitted over the edges (s_1, t) and (s_2, t).

Suppose now that each of the data values is known to be equally likely to be 0 or 1. A traditional network code assigns the same amount of communication resources for each edge and each block of data values. However, by relaxing this requirement, i.e., by letting the edge messages have variable-length based on the current block of data values, we can use the probability information to compress the number of bits that can be represented in the messages. This allows us to reduce the communication load; in this example, we demonstrate a network code with rate 0.8 in the variable-length network code framework. We describe a method to obtain an upper bound to the rate in the variable-length code framework. This method is general and can be applied to arbitrary demand functions.

Previous literature on computation capacity in the variable-length framework is only applicable to either tree-networks or networks in which the sources are directly connected to the terminal. For directed acyclic graph networks, existing literature has mainly focused on finding the computation capacity in the standard fixed-length network code framework.

CHAPTER 2. SUM-NETWORKS FROM INCIDENCE STRUCTURES

[1] A sum-network is an instance of a function computation problem over a directed acyclic network in which each terminal node wants to compute the sum over a finite field of the information observed at all the source nodes. Many characteristics of the well-studied multiple unicast network communication problem also hold for sum-networks due to a known reduction between the two problems. In this work, we describe an algorithm to construct families of sum-network instances using *incidence structures*. The computation capacity of several of these sum-network families is evaluated. Unlike the coding capacity of a multiple unicast problem, the computation capacity of sum-networks depends on the characteristic of the finite field over which the sum is computed. This dependence is very strong; we show examples of sum-networks that have a rate-1 solution over one characteristic but a rate close to zero over a different characteristic. Additionally, a sum-network can have arbitrarily different computation capacities for different alphabets.

2.1 Introduction

Applications as diverse as parallel processing, distributed data analytics and sensor networks often deal with variants of the problem of distributed computation. This has motivated the study of various problems in the fields of computer science, automatic control and information theory. Broadly speaking, one can model this question in the following manner. Consider a directed acyclic network with its edges denoting communication links. A subset of the nodes observe certain information, these nodes are called sources. A different subset of nodes, called terminals, wish to compute functions of the observed information with a certain fidelity. The computation is carried out by the terminals with the aid of the information received over their incoming edges. The demand

[1] This chapter is adapted from an article published in the IEEE Transactions on Information Theory. Parts of this work have been presented at the 52nd Allerton Conference on Communication, Control and Computing, 2014 and the 2015 IEEE International Symposium on Information Theory.

functions and the network topology are a part of the problem instance and can be arbitrary. This framework is very general and encompasses several problems that have received significant research attention.

Prior work [1],[2],[3] concerning information theoretic issues in function computation worked under the setting of correlated information observed at the sources and *simple* network structures, which were simple in the sense that there were edges connecting the sources to the terminal without any intermediate nodes or relays. For instance, [2] characterizes the amount of information that a source must transmit so that a terminal with some correlated side-information can reliably compute a function of the message observed at the source and the side-information. Reference [3] considered distributed functional compression, in which two messages are separately encoded and given to a decoder that computes a function of the two messages with an arbitrarily small probability of error.

With the advent of network coding [4],[5], the scope of the questions considered included the setting in which the information observed at the sources is independent and the network topology is more complex. Under this setting, information is sent from a source to a terminal over a path of edges in the directed acyclic network with one or more intermediate nodes in it, these relay nodes have no limit on their memory or computational power. The communication edges are abstracted into error-free, delay-free links with a certain capacity for information transfer and are sometimes referred to as *bit-pipes*. The messages are required to be recovered with zero distortion. The *multicast* scenario, in which the message observed at the only source in the network is demanded by all terminals in the network, is solved in [4],[5],[6]. A sufficient condition for solvability in the multicast scenario is that each terminal has a max-flow from the source that is at least the entropy rate of the message random process [4]. Reference [6] established that *linear* network codes over a sufficiently large alphabet can solve this problem and [5] provided necessary and sufficient conditions for solving a multicast problem instance in an algebraic framework. The work in [5] also gave a simple algorithm to construct a network code that satisfies it.

Unlike the multicast problem, the multiple unicast problem does not admit such a clean solution. This scenario has multiple source-terminal pairs over a directed acyclic network of bit-pipes and

each terminal wants to recover the message sent by its corresponding source with the help of the information transmitted on the network. To begin with, there are problem instances where more than one use of the network is required to solve it. To model this, each network edge is viewed as carrying a vector of n alphabet symbols, while each message is a vector of m alphabet symbols. A network code specifies the relationship between the vector transmitted on each edge of the network and the message vectors, and it solves a network coding problem instance if $m = n$. It is shown that linear network codes are in general not sufficient to solve this problem [7]. One can define the notion of *coding capacity* of a network as the supremum of the ratio m/n over all network codes that allow each terminal to recover its desired message; this ratio m/n for a particular network code is called its *rate*. The coding capacity of a network is independent of the alphabet used [8]. While a network code with any rational rate less than the coding capacity exists by definition and zero-padding, a network code with rate equal to coding capacity does not exist for certain networks, even if the coding capacity is rational [9]. The multi-commodity flow solution to the multiple unicast problem is called a routing solution, as the different messages can be interpreted as distinct commodities *routed* through the intermediate nodes. It is well-known that in the case of multicast, network coding can provide a gain in rate over traditional routing of messages that scales with the size of the network [10]. However, evaluating the coding capacity for an arbitrary instance of the network coding problem is known to be hard in general [11], [12], [13], [14].

Expanding the scope of the demands of the terminals, [15] considered *function computation* over directed acyclic networks with only one terminal; the value to be recovered at the terminal was allowed to be a function of the messages as opposed to being a subset of the set of all messages. This computation is performed using information transmitted over the edges by a network code. Analogous to the coding capacity, a notion of *computation capacity* can be defined in this case. A rate-m/n network code that allows the terminal to compute its demand function has the interpretation that the function can be computed by the terminal m times in n uses of the network. Cut-set based upper bounds for the computation capacity of a directed acyclic network with one terminal were given in [15],[16]. A matching lower bound for function computation in tree-networks

was given in [15] and the computation capacity of linear and non-linear network codes for different *classes* of demand functions was explored in [17].

A different flavor of the function computation problem, often called the *sum-network* problem, considers directed acyclic networks with multiple terminals, each of which demands the finite-field sum of all the messages observed at the sources [18], [19]. Reference[20] characterized the requirements that sum-networks with two or three sources or terminals must satisfy so that each terminal can recover the sum at unit rate. Similar to the network coding scenario, a sum-network whose terminals are satisfied by a rate-1 network code are called solvable sum-networks. Reference [19] established that deciding whether an arbitrary instance of a sum-network problem instance is solvable is at least as hard as deciding whether a suitably defined multiple unicast instance is solvable. As a result of this reduction the various characteristics of the solvability problem for network coding instances are also true for the solvability problem for sum-networks; this establishes the broadness of the class of sum-networks within all communication problems on directed acyclic networks.

While solvable sum-networks and solvable network coding instances are intimately related, the results in this paper indicate that these classes of problems diverge when we focus on coding/computation capacity, which can be strictly less than one. In [8, Section VI], the coding capacity of networks is shown to be independent of the finite field chosen as the alphabet for the messages and the information transmitted over the edges. We show that an analogous statement is not true for sum-networks by demonstrating infinite families of sum-network problem instances whose computation capacity vary depending on the finite field alphabet. Moreover, the gap in computation capacity on two different finite fields is shown to scale with the network size for certain classes of sum-networks. For two alphabets $\mathcal{F}_1, \mathcal{F}_2$ of different cardinality and a network $\mathcal{N}$, the authors in [8, Theorem VI.5] described a procedure to simulate a rate-m_2/n_2 network code on $\mathcal{F}_2$ for $\mathcal{N}$ using a rate-m_1/n_1 network code on $\mathcal{F}_1$ for the same network, such that $m_2/n_2 \geq (m_1/n_1) - \epsilon$ for any $\epsilon > 0$. That procedure does not apply for sum-networks. Along the lines of the counterexample given in [20] regarding minimum max-flow connectivity required for solvability of sum-networks with three

sources and terminals, we provide an infinite family of counterexamples that mandate certain value of max-flow connectivity to allow solvability (over some finite field) of a general sum-network with more than three sources and terminals. These sum-network problem instances are arrived at using a systematic construction procedure on combinatorial objects called *incidence structures*. Incidence structures are structured set systems and include, e.g., graphs and combinatorial designs [21]. We note here that combinatorial designs have recently been used to address issues such as the construction of distributed storage systems [22; 23], coded caching systems [25; 26; 27], and in reducing the level of file splitting required for distributed computation [53].

This paper is organized as follows. Section 2.2 describes previous work related to the problem considered and summarizes the contributions. Section 2.3 describes the problem model formally and Section 2.4 describes the construction procedure we use to obtain the sum-network problem instances considered in this work. Section 2.5 gives an upper bound on the computation capacity of these sum-networks and Section 2.6 describes a method to obtain linear network codes that achieve the upper bound on rate for several families of the sum-networks constructed. Section 2.7 interprets the results in this paper and outlines the key conclusions drawn in this paper. Section 2.8 concludes the paper and discusses avenues for future work.

2.2 Background, related work and summary of contributions

The problem setting in which we will work is such that the information observed at the sources are independent and uniformly distributed over a finite field alphabet $\mathcal{F}$. The network links are error-free and assumed to have unit-capacity. Each of the possibly many terminals wants to recover the finite field sum of all the messages with zero error. This problem was introduced in the work of [18]. Intuitively, it is reasonable to assume the network resources, i.e., the capacity of the network links and the network structure have an effect on whether the sum can be computed successfully by all the terminals in the network. Reference [20] characterized this notion for the class of sum-networks that have either two sources and/or two terminals. For this class of sum-networks it was shown that if the source messages had unit-entropy, a max-flow of one between each source-terminal

pair was enough to solve the problem. It was shown by means of a counterexample that a max-flow of one was not enough to solve a sum-network with three sources and terminals. However, it was also shown that a max-flow of two between each source-terminal pair was sufficient to solve any sum-network with three sources and three terminals. Reference [28] considered the computation capacity of the class of sum-networks that have three sources and three or more terminals or vice versa. It was shown that for any integer $k \geq 2$, there exist three-source, n-terminal sum-networks (where $n \geq 3$) whose computation capacity is $\frac{k}{k+1}$. The work most closely related to this paper is [29], which gives a construction procedure that for any positive rational number p/q returns a sum-network whose computation capacity is p/q. Assuming that p and q are relatively prime, the procedure described in [29] constructs a sum-network that has $2q - 1 + \binom{2q-1}{2}$ sources and $2q + \binom{2q-1}{2}$ terminals, which can be very large when q is large. The authors asked the question if there exist smaller sum-networks (i.e., with fewer sources and terminals) that have the computation capacity as p/q. Our work in [30] answered it in the affirmative and proposed a general construction procedure that returned sum-networks with a prescribed computation capacity. The sum-networks in [29] could be obtained as special cases of this construction procedure. Some smaller instances of sum-networks for specific values were presented in [31]. Small sum-network instances can be useful in determining sufficiency conditions for larger networks. The scope of the construction procedure proposed in [30] was widened in [32], as a result of which, it was shown that there exist sum-network instances whose computation capacity depends rather strongly on the finite field alphabet. This work builds on the contributions in [30; 32]. In particular, we present a systematic algebraic technique that encompasses the prior results. We also include proofs of all results and discuss the implications of our results in depth.

2.2.1 Summary of contributions

In this work, we define several classes of sum-networks for which we can explicitly determine the computation capacity. These networks are constructed by using appropriately defined incidence structures. The main contributions of our work are as follows.

- We demonstrate novel techniques for determining upper and lower bounds on the computation capacity of the constructed sum-networks. In most cases, these bounds match, thus resulting in a determination of the capacity of these sum-networks.

- We demonstrate a strong dependence of the computation capacity on the characteristic of the finite field over which the computation is taking place. In particular, for the *same* network, the computation capacity changes based on the characteristic of the underlying field. This is unlike the coding capacity for the multiple unicast problem which is known to be independent of the network coding alphabet.

- Consider the class of networks where every source-terminal pair has a minimum cut of value at least α, where α is an arbitrary positive integer. We demonstrate that there exists a sum-network within this class (with a large number of sources and terminals) whose computation capacity can be made arbitrarily small. This implies that the capacity of sum-networks cannot be characterized just by individual source-terminal minimum cuts.

2.3 Problem formulation and preliminaries

We consider communication over a directed acyclic graph (DAG) $G = (V, E)$ where V is the set of nodes and $E \subseteq V \times V \times \mathbb{Z}_+$ are the edges denoting the delay-free communication links between them. The edges are given an additional index as the model allows for multiple edges between two distinct nodes. For instance, if there are two edges between nodes u and v, these will be represented as $(u, v, 1)$ and $(u, v, 2)$. Subset $S \subset V$ denotes the source nodes and $T \subset V$ denotes the terminal nodes. The source nodes have no incoming edges and the terminal nodes have no outgoing edges. Each source node $s_i \in S$ observes an independent random process X_i, such that the sequence of random variables $X_{i1}, X_{i2}, \ldots$ indexed by time (denoted by a positive integer) are i.i.d. and each X_{ij} takes values that are uniformly distributed over a finite alphabet $\mathcal{F}$. The alphabet $\mathcal{F}$ is assumed to be a finite field with $|\mathcal{F}| = q$ and its characteristic denoted as $\mathrm{ch}(\mathcal{F})$. Each edge represents a communication channel of unit capacity, i.e., it can transmit one symbol from $\mathcal{F}$ per time slot.

When referring to a communication link (or edge) without its third index, we will assume that it is the set of all edges between its two nodes. For such a set denoted by (u, v), we define its capacity $\text{cap}(u, v)$ as the number of edges between u and v. We use the notation $\text{In}(v)$ and $\text{In}(e)$ to represent the set of incoming edges at node $v \in V$ and edge $e \in E$. For the edge $e = (u, v)$ let $\text{head}(e) = v$ and $\text{tail}(e) = u$. Each terminal node $t \in T$ demands the sum (over $\mathcal{F}$) of the individual source messages. Let $Z_j = \sum_{\{i:s_i \in S\}} X_{ij}$ for all $j \in \mathbb{N}$ (the set of natural numbers); then each $t \in T$ wants to recover the sequence $Z \coloneqq (Z_1, Z_2, \dots)$ from the information it receives on its incoming edges, i.e., the set $\text{In}(t)$.

A network code is an assignment of local encoding functions to each edge $e \in E$ (denoted as $\tilde{\phi}_e(\cdot)$) and a decoding function to each terminal $t \in T$ (denoted as $\psi_t(\cdot)$) such that all the terminals can compute Z. The local encoding function for an edge connected to a set of sources only has the messages observed at those particular source nodes as its input arguments. Likewise, the input arguments for the local encoding function of an edge that is not connected to any source are the values received on its incoming edges and the inputs for the decoding function of a terminal are the values received on its incoming edges. As we consider directed acyclic networks, it can be seen that there is a *global* encoding function that expresses the value transmitted on an edge in terms of the source messages in the set $X \coloneqq \{X_i : s_i \in S\}$. The global encoding function for an edge e is denoted as $\phi_e(X)$.

The following notation describes the domain and range of the local encoding and decoding functions using two natural numbers m and n for a general vector network code. m is the number of i.i.d. source values that are encoded simultaneously by the local encoding function of an edge that emanates from a source node. n is the number of symbols from $\mathcal{F}$ that are transmitted across an edge in the network. Thus for such an edge e whose $\text{tail}(e) = s \in S$, the local encoding function is $\tilde{\phi}_e(X_{s1}, X_{s2}, \dots, X_{sm}) \in \mathcal{F}^n$. We will be using both row and column vectors in this paper and they will be explicitly mentioned while defining them. If u is a vector, the u^T represents its transpose.

- Local encoding function for edge $e \in E$.

$$\tilde{\phi}_e : \mathcal{F}^m \to \mathcal{F}^n \quad \text{if tail}(e) \in S,$$
$$\tilde{\phi}_e : \mathcal{F}^{n|\,\text{In}(\text{tail}(e))|} \to \mathcal{F}^n \quad \text{if tail}(e) \notin S.$$

- Decoding function for the terminal $t \in T$.

$$\psi_t : \mathcal{F}^{n|\,\text{In}(t)|} \to \mathcal{F}^m.$$

A network code is linear over the finite field $\mathcal{F}$ if all the local encoding and decoding functions are linear transformations over $\mathcal{F}$. In this case the local encoding functions can be represented via matrix products where the matrix elements are from $\mathcal{F}$. For example, for an edge e such that $\text{tail}(e) \notin S$, let $c \in \mathbb{N}$ be such that $c = |\,\text{In}(\text{tail}(e))|$ and $\text{In}(\text{tail}(e)) = \{e_1, e_2, \ldots, e_c\}$. Also, let each $\phi_{e_i}(X) \in \mathcal{F}^n$ be denoted as a column vector of size n whose elements are from $\mathcal{F}$. Then the value transmitted on e can be evaluated as

$$\phi_e(X) = \tilde{\phi}_e(\phi_{e_1}(X), \phi_{e_2}(X), \ldots, \phi_{e_c}(X)) = M_e \begin{bmatrix} \phi_{e_1}(X)^T & \phi_{e_2}(X)^T & \ldots & \phi_{e_c}(X)^T \end{bmatrix}^T,$$

where $M_e \in \mathcal{F}^{n \times nc}$ is a matrix indicating the local encoding function for edge e. For the sum-networks that we consider, a valid (m, n) fractional network code solution over $\mathcal{F}$ has the interpretation that the component-wise sum over $\mathcal{F}$ of m i.i.d. source symbols can be communicated to all the terminals in n time slots.

Definition 1 *The rate of a (m, n) network code is defined to be the ratio m/n. A sum-network is solvable if it has a (m, m) network coding solution for some $m \in \mathbb{N}$.*

Definition 2 *The computation capacity of a sum-network is defined as*

$$\sup \left\{ \frac{m}{n} : \begin{array}{c} \textit{there is a valid } (m, n) \textit{ network code} \\ \textit{for the given sum-network.} \end{array} \right\}$$

We use different types of *incidence structures* for constructing sum-networks throughout this paper. We now formally define and present some examples of incidence structures.

Definition 3 *Incidence Structure. Let $\mathcal{P}$ be a set of elements called points, and $\mathcal{B}$ be a set of elements called blocks, where each block is a subset of $\mathcal{P}$. The incidence structure $\mathcal{I}$ is defined as the pair $(\mathcal{P}, \mathcal{B})$. If $p \in \mathcal{P}, B \in \mathcal{B}$ such that $p \in B$, then we say that point p is incident to block B. In general $\mathcal{B}$ can be a multiset, i.e., it can contain repeated elements, but we will not be considering them in our work. Thus for any two distinct blocks B_1, B_2 there is at least one point which is incident to one of B_1 and B_2 and not the other.*

We denote the cardinalities of the sets $\mathcal{P}$ and $\mathcal{B}$ by the constants v and b respectively. Thus the set of points and blocks can be indexed by a subscript, and we have that

$$\mathcal{P} = \{p_1, p_2, \ldots, p_v\}, \text{ and } \mathcal{B} = \{B_1, B_2, \ldots, B_b\}.$$

Definition 4 *Incidence matrix. The incidence matrix associated with the incidence structure $\mathcal{I}$ is a $(0,1)$-matrix of dimension $v \times b$ defined as follows.*

$$A_{\mathcal{I}}(i,j) \coloneqq \begin{cases} 1 & \text{if } p_i \in B_j, \\ 0 & \text{otherwise.} \end{cases}$$

Thus, incidence matrices can be viewed as general set systems. For example, a simple undirected graph can be viewed as an incidence structure where the vertices are the points and edges are the blocks (each block is of size two). Combinatorial designs [21] form another large and well-investigated class of incidence structures. In this work we will use the properties of t-designs which are defined next.

Definition 5 *t-design. An incidence structure $\mathcal{I} = (\mathcal{P}, \mathcal{B})$ is a t-(v, k, λ) design, if*

- *it has v points, i.e., $|\mathcal{P}| = v$,*
- *each block $B \in \mathcal{B}$ is a k-subset of the point set $\mathcal{P}$, and*
- *$\mathcal{P}$ and $\mathcal{B}$ satisfy the t-design property, i.e., any t-subset of $\mathcal{P}$ is present in exactly λ blocks.*

A t-(v, k, λ) design is called *simple* if there are no repeated blocks. These designs have been the subject of much investigation when $t = 2$; in this case they are also called balanced incomplete block designs (BIBDs).

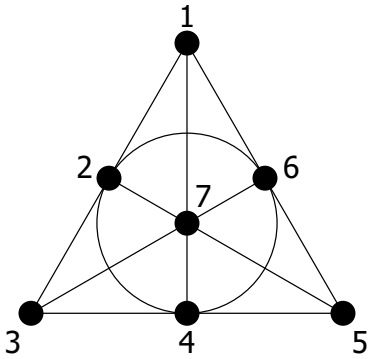


Figure 2.1: A pictorial depiction of the Fano plane. The point set $\mathcal{P} = \{1, \ldots, 7\}$. The blocks are indicated by a straight line joining their constituent points. The points $2, 4$ and 6 lying on the circle also depict a block.

Example 1 *A famous example of a 2-design with $\lambda = 1$ is the Fano plane $\mathcal{I} = (\mathcal{P}, \mathcal{B})$ shown in Figure 2.1. Letting numerals denote points and alphabets denote blocks for this design, we have:*

$\mathcal{P} = \{1, 2, 3, 4, 5, 6, 7\}, \mathcal{B} = \{A, B, C, D, E, F, G\}$, *where*

$A = \{1, 2, 3\}, B = \{3, 4, 5\}, C = \{1, 5, 6\}, D = \{1, 4, 7\}, E = \{2, 5, 7\}, F = \{3, 6, 7\}, G = \{2, 4, 6\}.$

The corresponding incidence matrix $A_{\mathcal{I}}$, with rows and columns arranged in numerical and alphabetical order, is shown below.

$$A_{\mathcal{I}} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}. \tag{2.1}$$

It can be verified that every pair of points in $\mathcal{P}$ appears in exactly one block in $\mathcal{B}$.

There are some well-known conditions that the parameters of a t-(v, k, λ) design satisfy (see [21]).

- For integer $i \leq t$ the number of blocks incident to any i-subset of $\mathcal{P}$ is the same. We let b_i denote that constant. Then,

$$b_i = \lambda \binom{v-i}{t-i} / \binom{k-i}{t-i}, \ \forall i \in \{0, 1, 2, \ldots, t\}. \tag{2.2}$$

  We note that b_0 is simply the total number of blocks denoted by b. Likewise, b_1 represents the number of blocks that each point is incident to; we use the symbol ρ to represent it. Furthermore, $b_t = \lambda$.

  It follows that a necessary condition for the existence of a t-(v, k, λ) design is that $\binom{k-i}{t-i}$ divides $\lambda\binom{v-i}{t-i}$ for all $i = 1, 2, \ldots, t$.

- Counting the number of ones in the point-block incidence matrix for a particular design in two different ways, we arrive at the equation $bk = v\rho$.

2.4 Construction of a family of sum-networks

Let $[t] \coloneqq \{1, 2, \ldots, t\}$ for any $t \in \mathbb{N}$. Our construction takes as input a $(0,1)$-matrix A of dimension $r \times c$.

Definition 6 *Notation for row and column of A. Let $\boldsymbol{p}_i$ denote the i-th row vector of A for $i \in [r]$ and $\boldsymbol{B}_j$ denote the j-th column vector of A for $j \in [c]$* [2].

It turns out that the constructed sum-networks have interesting properties when the matrix A is the incidence matrix of appropriately chosen incidence structures. The construction algorithm is presented in Algorithm 1. The various steps in the algorithm that construct components of the sum-network $G = (V, E)$ are described below.

1. *Source node set S and terminal node set T:* S and T both contain $r + c$ nodes, one for each row and column of A. The source nodes are denoted at line 4 as s_{p_i}, s_{B_j} if they correspond to the i-th row, j-th column respectively. The terminal nodes are also denoted in a similar manner at line 5. They are added to the vertex set V of the sum-network at line 6.

[2] *A justification for this notation is that later when we use the incidence matrix ($A_{\mathcal{I}}$) of an incidence structure $\mathcal{I}$ to construct a sum-network, the rows and columns of the incidence matrix will correspond to the points and blocks of $\mathcal{I}$ respectively.*

2. *Bottleneck edges:* We add r unit-capacity edges indexed as e_i for $i \in [r]$ in line 2 to the edge set E. Each edge e_i corresponds to a row of the matrix A. We also add the required tail and head vertices of these edges to V.

3. *Edges between $S \cup T$ and the bottleneck edges:* For every $i \in [r]$, we connect $\text{tail}(e_i)$ to the source node corresponding to the row $\boldsymbol{p}_i$ and to the source nodes that correspond to all columns of A with a 1 in the i-th row. This is described in line 8 of the algorithm. Line 9 describes a similar operation used to connect each $\text{head}(e_i)$ to certain terminal nodes.

4. *Direct edges between S and T:* For each terminal in T, these edges directly connect it to source nodes that do not have a path to that particular terminal through the bottleneck edges. Using the notation for rows and columns of the matrix A, they can be characterized as in lines 12 and 15.

Algorithm 1 SUM-NET-CONS

Input: A.
Output: $G = (V, E)$.

1: Initialize $V, E, S, T \leftarrow \phi$.
2: $E \leftarrow \{e_i : i \in [r]\}$.
3: $V \leftarrow \{\text{head}(e_i), \text{tail}(e_i) : i \in [r]\}$.
4: $S \leftarrow \{s_{p_i} : i \in [r]\} \cup \{s_{B_j} : j \in [c]\}$.
5: $T \leftarrow \{t_{p_i} : i \in [r]\} \cup \{t_{B_j} : j \in [c]\}$.
6: $V \leftarrow V \cup S \cup T$.
7: **for all** $i \in [r]$ **do**
8: $\quad E \leftarrow E \cup \{(s_{B_j}, \text{tail}(e_i)) : A(i,j) = 1; j \in [c]\} \cup \{(s_{p_i}, \text{tail}(e_i))\}$.
9: $\quad E \leftarrow E \cup \{(\text{head}(e_i), t_{B_j}) : A(i,j) = 1; j \in [c]\} \cup \{(\text{head}(e_i), t_{p_i})\}$.
10: **end for**
11: **for all** $i \in [r]$ **do**
12: $\quad E \leftarrow E \cup \{(s_{p_j}, t_{p_i}) : i \neq j; j \in [r]\} \cup \{(s_{B_j}, t_{p_i}) : A(i,j) = 0; j \in [c]\}$.
13: **end for**
14: **for all** $j \in [c]$ **do**
15: $\quad E \leftarrow E \cup \{(s_{p_i}, t_{B_j}) : A(i,j) = 0; i \in [r]\} \cup \{(s_{B_{j'}}, t_{B_j}) : \boldsymbol{B}_j^T \boldsymbol{B}_{j'} = 0; j' \in [c]\}$.
16: **end for**
17: **return** $G \leftarrow (V, E)$.

For an incidence structure $\mathcal{I}$, let $A_{\mathcal{I}}$ represent its incidence matrix. The sum-networks constructed in the paper are such that the matrix A used in the SUM-NET-CONS algorithm is either equal to $A_{\mathcal{I}}$ or $A_{\mathcal{I}}^T$ for some incidence structure $\mathcal{I}$. When $A = A_{\mathcal{I}}$, we call the sum-network constructed as the *normal* sum-network for $\mathcal{I}$. Otherwise when $A = A_{\mathcal{I}}^T$, we call the sum-network constructed as the *transpose* sum-network for $\mathcal{I}$. The following definitions are useful for analysis. For every $p \in \mathcal{P}$, we denote the set of blocks that contain the point p as

$$\langle p \rangle \coloneqq \{B \in \mathcal{B} : p \in B\}, \tag{2.3}$$

and for every $B \in \mathcal{B}$, the collection of blocks that have a non-empty intersection with B is denoted by the set

$$\langle B \rangle \coloneqq \{B' \in \mathcal{B} : B' \cap B \neq \phi\} \tag{2.4}$$

$$= \{B' \in \mathcal{B} : \boldsymbol{B}^T \boldsymbol{B'} \neq 0\}, \tag{2.5}$$

where boldface $\boldsymbol{B}$ indicates the column of $A_{\mathcal{I}}$ corresponding to block $B \in \mathcal{B}$.

The inner product above is computed over the reals. In the sequel, we will occasionally need to perform operations similar to the inner product over a finite field. This shall be explicitly pointed out.

We now present some examples of sum-networks constructed using the above technique.

Example 2 *Let $\mathcal{I}$ be the unique simple line graph on two vertices, with points corresponding to the vertices and blocks corresponding to the edges of the graph. Denoting the points as natural numbers, we get that $\mathcal{P} = \{1, 2\}$ and $\mathcal{B} = \{\{1, 2\}\}$. Then the associated incidence matrices are as follows.*

$$A_{\mathcal{I}} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \text{ and } A_{\mathcal{I}}^T = \begin{bmatrix} 1 & 1 \end{bmatrix}.$$

Following the SUM-NET-CONS algorithm the two sum-networks obtained are as shown in the Figure 2.2.

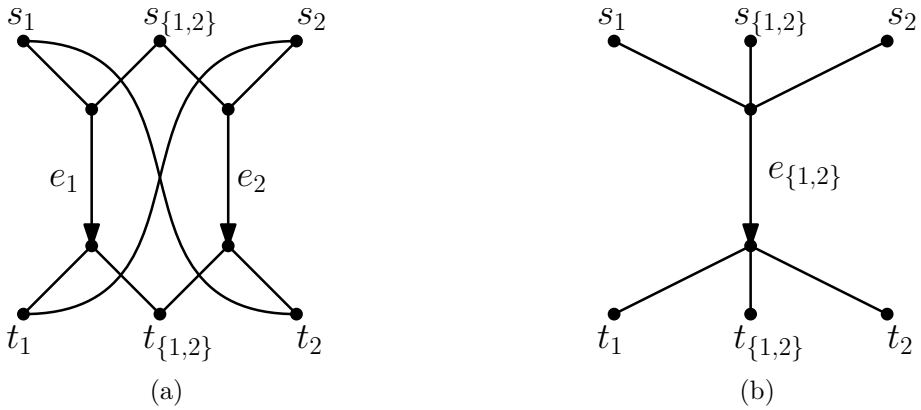


Figure 2.2: Two sum-networks obtained from the line graph on two vertices described in Example 2. The source set S and the terminal set T contain three nodes each. All edges are unit-capacity and point downward. The edges with the arrowheads are the bottleneck edges constructed in step 2 of the construction procedure. (a) Normal sum-network, and (b) transposed sum-network.

Example 3 *In this example we construct a sum-network using a simple t-design. Let $\mathcal{I}$ denote the 2-$(3, 2, 1)$ design with its points denoted by the numbers $\{1, 2, 3\}$ and its blocks denoted by the letters $\{A, B, C\}$. For this design we have that $A = \{1, 2\}, B = \{1, 3\}, C = \{2, 3\}$ and its associated incidence matrix under row and column permutations can be written as follows.*

$$A_{\mathcal{I}} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

Note that $A_{\mathcal{I}} = A_{\mathcal{I}}^T$. Hence the normal sum-network and the transposed sum-network are identical in this case. Following the SUM-NET-CONS algorithm, we obtain the sum-network shown in Figure 2.3.

Remark 1 *Note that each edge added in the SUM-NET-CONS algorithm has unit capacity. Proposition 6 in Section 2.7 modifies the SUM-NET-CONS algorithm so that each edge e in the sum-network has* $\text{cap}(e) = \alpha > 1, \alpha \in \mathbb{N}$.

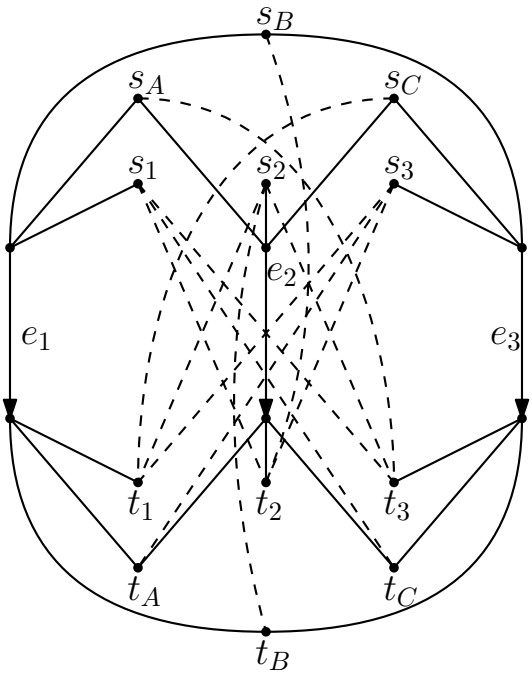


Figure 2.3: The normal sum-network obtained for the incidence structure $\mathcal{I}$ described in Example 3. All edges are unit-capacity and directed downward. The edges with the arrowheads are the bottleneck edges, and the edges denoted by dashed lines correspond to the direct edges introduced in step 4 of the construction procedure. For this case, the normal and the transposed sum-network are identical.

2.5 Upper bound on the computation capacity

In this section, we describe an upper bound on the computation capacity of a sum-network obtained from a $(0,1)$-matrix A of dimension $r \times c$. We assume that there exists a (m,n) fractional network code assignment, i.e., $\tilde{\phi}_e$ for $e \in E$ (and corresponding global encoding functions $\phi_e(X)$) and decoding functions ψ_t for $t \in T$ so that all the terminals in T can recover the sum of all the independent sources.

For convenience of presentation, we will change notation slightly and let the messages observed at the source nodes corresponding to the rows of A as X_{p_i} for $i \in [r]$ and those corresponding to the columns of A as X_{B_j} for $j \in [c]$. Each of the messages is a column vector of length m over $\mathcal{F}$. The set of all source messages is represented by X. We let $\phi_e(X)$ denote the n-length column vector of symbols from $\mathcal{F}$ that are transmitted by the edge $e \in E$, as it is the value returned by the

global encoding function ϕ_e for edge e on the set of source messages denoted by X. As is apparent, non-trivial encoding functions can only be employed on the bottleneck edges, i.e., e_i for $i \in [r]$ as these are the only edges that have more than one input. For brevity, we denote $\phi_i(X) = \phi_{e_i}(X)$. We define the following set of global encoding functions.

$$\phi_{\mathrm{In}(v)}(X) \coloneqq \{\phi_e(X) : e \in \mathrm{In}(v)\}, \ \forall v \in V.$$

Let $H(Y)$ be the entropy function for a random variable Y. We let $\{Y_i\}_1^l$ denote the set $\{Y_1, Y_2, \ldots, Y_l\}$ for any $l > 1$. The following lemma demonstrates that certain partial sums can be computed by observing subsets of the bottleneck edges.

Lemma 1 *If a network code allows each terminal to compute the demanded sum, then the value* $X'_{p_i} \coloneqq X_{p_i} + \sum_{j:A(i,j)=1} X_{B_j}$ *can be computed from* $\phi_i(X)$*, i.e.,* $H\left(X'_{p_i}|\phi_i(X)\right) = 0$ *for all* $i \in [r]$*. Similarly for any* $j \in [c]$ *the value* $X'_{B_j} \coloneqq \sum_{i:A(i,j)=1} X_{p_i} + \sum_{j':B_{j'} \in \langle B_j \rangle} X_{B_{j'}}$ *can be computed from the set of values* $\{\phi_i(X) : \text{ for } i \in [r], A(i,j) = 1\}$.

Proof: We let for any $i \in [r]$

$$Z_1 = \sum_{i' \neq i} X_{p_{i'}}, \ Z_2 = \sum_{j:A(i,j)=1} X_{B_j} \text{ and } Z_3 = \sum_{j:A(i,j)=0} X_{B_j},$$

such that the sum $Z = X_{p_i} + Z_1 + Z_2 + Z_3$ and $X'_{p_i} = X_{p_i} + Z_2$.

By our assumption that each terminal can recover the demanded sum, we know that Z can be evaluated from $\phi_{\mathrm{In}\left(t_{p_i}\right)}(X)$ for all $i \in [r]$, i.e., $H\left(Z|\phi_{\mathrm{In}(t_{p_i})}(X)\right) = 0$ for all $i \in [r]$. Since $\{X_{p_{i'}} : i' \neq i\}$ and $\{X_{B_j} : A(i,j) = 0\}$ determine the value of Z_1 and Z_3 respectively and also determine the values transmitted on each of the direct edges that connect a source node to t_{p_i}, we

get that

$$
\begin{aligned}
& H\Big(Z|\phi_{\mathrm{In}(t_{p_i})}(X)\Big) \\
&= H\left(Z|\phi_i(X), \{\phi_{(s_{p_{i'}},t_{p_i})}(X) : i' \neq i\}, \{\phi_{(s_{B_j},t_{p_i})} : A(i,j) = 0\}\right) \\
&\overset{(a)}{\geq} H\left(X_{p_i} + Z_1 + Z_2 + Z_3|\phi_i(X), \{X_{p_{i'}} : i' \neq i\}, \{X_{B_j} : A(i,j) = 0\}\right) \\
&= H\left(X'_{p_i}|\phi_i(X), \{X_{p_{i'}} : i' \neq i\}, \{X_{B_j} : A(i,j) = 0\}\right) \\
&= H\left(X'_{p_i}, \{X_{p_{i'}} : i' \neq i\}, \{X_{B_j} : A(i,j) = 0\}|\phi_i(X)\right) \\
&\quad - H\left(\{X_{p_{i'}} : i' \neq i\}, \{X_{B_j} : A(i,j) = 0\}|\phi_i(X)\right) \\
&= H\left(X'_{p_i}|\phi_i(X)\right) + H\left(\{X_{p_{i'}} : i' \neq i\}, \{X_{B_j} : A(i,j) = 0\}|X'_{p_i}, \phi_i(X)\right) \\
&\quad - H\left(\{X_{p_{i'}} : i' \neq i\}, \{X_{B_j} : A(i,j) = 0\}|\phi_i(X)\right) \\
&\overset{(b)}{=} H\left(X'_{p_i}|\phi_i(X)\right), \qquad (2.6)
\end{aligned}
$$

where inequality (a) follows from the fact that $\phi_{(s_{p_{i'}},t_{p_i})}(X)$ is a function of $X_{p_{i'}}$ for $i' \neq i$ and $\phi_{(s_{B_j},t_{p_i})}(X)$ is a function of $\{X_{B_j} : A(i,j) = 0\}$ and equality (b) is due to the fact that X'_{p_i} is conditionally independent of both $\{X_{p_{i'}} : i' \neq i\}$ and $\{X_{B_j} : A(i,j) = 0\}$ given $\phi_i(X)$. This conditional independence can be checked as follows. Let bold lowercase symbols represent specific realizations of the random variables.

$$
\begin{aligned}
& \Pr\left(X'_{p_i} = \boldsymbol{x}'_{p_i}, \{X_{p_{i'}} = \boldsymbol{x}_{p_{i'}} : i' \neq i\}, \{X_{B_j} = \boldsymbol{x}_{B_j} : A(i,j) = 0\}|\phi_i(X) = \phi_i(\boldsymbol{x})\right) \\
&\overset{(a)}{=} \frac{\Pr(X'_{p_i} = \boldsymbol{x}'_{p_i}, \phi_i(X) = \phi_i(\boldsymbol{x})) \cdot \Pr(\{X_{p_{i'}} = \boldsymbol{x}_{p_{i'}} : i' \neq i\}, \{X_{B_j} = \boldsymbol{x}_{B_j} : A(i,j) = 0\})}{\Pr(\phi_i(X) = \phi_i(\boldsymbol{x}))} \\
&\overset{(b)}{=} \Pr(X'_{p_i} = \boldsymbol{x}'_{p_i}|\phi_i(X) = \phi_i(\boldsymbol{x})) \Pr(\{X_{p_{i'}} = \boldsymbol{x}_{p_{i'}} : i' \neq i\}, \{X_{B_j} = \boldsymbol{x}_{B_j} : A(i,j) = 0\}|\phi_i(X) = \phi_i(\boldsymbol{x})),
\end{aligned}
$$

where equalities (a) and (b) are due to the fact that the source messages are independent and $\phi_i(\boldsymbol{x})$ is only a function of $\boldsymbol{x}_{p_i}$ and the set $\{\boldsymbol{x}_{B_j} : A(i,j) = 1\}$.

Since terminal t_{p_i} can compute Z, $H\left(Z|\phi_{\mathrm{In}(t_{p_i})}(X)\right) = 0$ and we get from eq. (2.6) that $H(X_{p_i} + Z_2|\phi_i(X)) = 0$.

For the second part of the lemma, we argue similarly as follows. We let for any $j \in [c]$

$$Z_1 = \sum_{i:A(i,j)=1} X_{p_i}, Z_2 = \sum_{i:A(i,j)=0} X_{p_i},$$
$$Z_3 = \sum_{B \in \langle B_j \rangle} X_B, Z_4 = \sum_{B \notin \langle B_j \rangle} X_B$$

such that $Z = Z_1 + Z_2 + Z_3 + Z_4$ and $X'_{B_j} = Z_1 + Z_3$. By our assumption, for all $j \in [c]$, $H\left(Z|\phi_{\text{In}\left(t_{B_j}\right)}(X)\right) = 0$. The sets $\{X_p : p \notin B_j\}$ and $\{X_B : B \notin \langle B_j \rangle\}$ determine the value of Z_2 and Z_4 respectively and also the values transmitted on each of the direct edges that connect a source node to the terminal t_{B_j}. Let Φ denote the set $\{\phi_i(X) : A(i,j) = 1\}$. Then,

$$\begin{aligned}
&H\left(Z|\phi_{\text{In}\left(t_{B_j}\right)}(X)\right) \\
&= H\left(Z_1 + Z_2 + Z_3 + Z_4|\Phi, \{\phi_{(s_{p_i},t_{B_j})}(X) : A(i,j) = 0\}, \{\phi_{(s_B,t_{B_j})} : B \notin \langle B_j \rangle\}\right) \\
&\overset{(a)}{\geq} H\left(Z_1 + Z_2 + Z_3 + Z_4|\Phi, \{X_{p_i} : A(i,j) = 0\}, \{X_B : B \notin \langle B_j \rangle\}\right) \\
&= H\left(X'_{B_j}|\Phi, \{X_{p_i} : A(i,j) = 0\}, \{X_B : B \notin \langle B_j \rangle\}\right) \\
&= H\left(X'_{B_j}, \{X_{p_i} : A(i,j) = 0\}, \{X_B : B \notin \langle B_j \rangle\}|\Phi\right) \\
&\qquad\qquad - H\left(\{X_{p_i} : A(i,j) = 0\}, \{X_B : B \notin \langle B_j \rangle\}|\Phi\right) \\
&= H(X'_{B_j}|\Phi) - H(\{X_{p_i} : A(i,j) = 0\}, \{X_B : B \notin \langle B_j \rangle\}|\Phi) \\
&\qquad\qquad + H(\{X_{p_i} : A(i,j) = 0\}, \{X_B : B \notin \langle B_j \rangle\}|X'_{B_j}, \Phi) \\
&\overset{(b)}{=} H(X'_{B_j}|\Phi).
\end{aligned}$$

Inequality (a) is due to the fact that $\phi_{(s_{p_i},t_{B_j})}(X)$ is a function of X_{p_i} and similarly for $\phi_{(s_B,t_{B_j})}(X)$. Equality (b) follows from the fact that $Z_1 + Z_3$ is conditionally independent of both $\{X_{p_i} : A(i,j) = 0\}$ and $\{X_{B_{j'}} : B \notin \langle B_j \rangle\}$ given the set of random variables $\{\phi_i(X) : A(i,j) = 1\}$. This can be verified in a manner similar to as was done previously. This gives us the result that $H(X'_{B_j}|\{\phi_i(X) : A(i,j) = 1\}) = 0$. ■

Next, we show the fact that the messages observed at the source nodes are independent and uniformly distributed over $\mathcal{F}^m$ imply that the random variables X'_{p_i} for all $i \in [r]$ are also uniform i.i.d. over $\mathcal{F}^m$. To do that, we introduce some notation. For a matrix $N \in \mathcal{F}^{r \times c}$, for any two

index sets $\mathcal{R} \subseteq [r], \mathcal{C} \subseteq [c]$, we define the submatrix of N containing the rows indexed by $\mathcal{R}$ and the columns indexed by $\mathcal{C}$ as $N[\mathcal{R}, \mathcal{C}]$. Consider two $(0,1)$-matrices N_1, N_2 of dimensions $r_1 \times t$ and $t \times c_2$ respectively. Here 1 and 0 indicate the multiplicative and additive identities of the finite field $\mathcal{F}$ respectively. The i-th row of N_1 is denoted by the row submatrix $N_1[i,[t]] \in \{0,1\}^t$ and the j-th column of N_2 be denoted by the column submatrix $N_2[[t],j] \in \{0,1\}^t$. Then we define a matrix function on N_1N_2 that returns a $r_1 \times c_2$ matrix $(N_1N_2)_\#$ as follows.

$$(N_1N_2)_\#(i,j) = \begin{cases} 1, & \text{if the product } N_1[i,[t]]\, N_2[[t],j] \\ & \text{over } \mathbb{Z} \text{ is positive,} \\ 0, & \text{otherwise.} \end{cases}$$

For an incidence structure $\mathcal{I} = (\mathcal{P}, \mathcal{B})$ with $r \times c$ incidence matrix A, let X_p, $\forall p \in \mathcal{P}$ and X_B, $\forall B \in \mathcal{B}$ be m-length vectors with each component i.i.d. uniformly distributed over $\mathcal{F}$. We collect all the independent source random variables in a column vector $\boldsymbol{X}$ having $m(r+c)$ elements from $\mathcal{F}$ as follows

$$\boldsymbol{X} := \begin{bmatrix} X_{p_1}^T & X_{p_2}^T & \cdots & X_{p_r}^T & X_{B_1}^T & X_{B_2}^T & \cdots & X_{B_c}^T \end{bmatrix}^T.$$

Recall that $\boldsymbol{p}_i$ denotes the i-th row and $\boldsymbol{B}_j$ denotes the j-th column of the matrix A. For all $i \in [r]$ let $\boldsymbol{e}_i \in \mathcal{F}^r$ denote the column vector with 1 in its i-th component and zero elsewhere. Then for X'_{p_i}, X'_{B_j} as defined in lemma 1, one can check that ($\otimes$ indicates the Kronecker product of two matrices)

$$X'_{p_i} = \left(\begin{bmatrix} \boldsymbol{e}_i^T & \boldsymbol{p}_i \end{bmatrix} \otimes I_m \right) \boldsymbol{X}, \quad \text{for all } i \in [r] \text{ and} \tag{2.7}$$

$$X'_{B_j} = \left(\begin{bmatrix} \boldsymbol{B}_j^T & (\boldsymbol{B}_j^T \boldsymbol{B}_1)_\# & \ldots & (\boldsymbol{B}_j^T \boldsymbol{B}_c)_\# \end{bmatrix} \otimes I_m \right) \boldsymbol{X}, \tag{2.8}$$

for all $j \in [c]$ where I_m is the identity matrix of size m. By stacking these values in the correct order, we can get the following matrix equation.

$$\begin{bmatrix} X'^T_{p_1} & \cdots & X'^T_{p_r} & X'^T_{B_1} & \cdots & X'^T_{B_c} \end{bmatrix}^T = (M_A \otimes I_m)\boldsymbol{X} \tag{2.9}$$

where the matrix $M_A \in \mathcal{F}^{(r+c)\times(r+c)}$ is defined as

$$M_A := \begin{bmatrix} I_r & A \\ A^T & (A^TA)_\# \end{bmatrix}. \tag{2.10}$$

Note that the first r rows of M_A are linearly independent. There is a natural correspondence between the rows of M_A and the points and blocks of $\mathcal{I}$ of which A is the incidence matrix. If $1 \leq i \leq r$, then the i-th row $M_A[i, [r+c]]$ corresponds to the point $p_i \in \mathcal{P}$ and if $r+1 \leq j \leq r+c$, then the j-th row $M_A[j, [r+c]]$ corresponds to the block $B_j \in \mathcal{B}$.

Lemma 2 *For a $(0,1)$-matrix A of size $r \times c$, let $X'_{p_i}, X'_{B_j} \in \mathcal{F}^m$ be as defined in Eqs. (2.7), (2.8) and matrix M_A be as defined in eq. (2.10). Let $r+t \coloneqq \operatorname{rank}_{\mathcal{F}} M_A$ for some non-negative integer t and index set $\mathcal{S}' \subseteq \{r+1, r+2, \ldots, r+c\}$ be such that $\operatorname{rank}_{\mathcal{F}} M_A[[r] \cup \mathcal{S}', [r+c]] = r+t$. Let $\mathcal{B}_{\mathcal{S}'} \coloneqq \{B_{\mathcal{S}'_1}, B_{\mathcal{S}'_2}, \ldots, B_{\mathcal{S}'_t}\} \subseteq \mathcal{B}$ be the set of blocks that correspond to the rows of M_A indexed by $\mathcal{S}'$ in increasing order. Then we have*

$$\Pr\left(X'_{p_1} = x'_1, \ldots, X'_{p_r} = x'_r, X'_{B_{\mathcal{S}'_1}} = y'_1, \ldots, X'_{B_{\mathcal{S}'_t}} = y'_t\right) = q^{-m(r+t)}, \text{ and} \tag{2.11}$$

$$\Pr\left(X'_{p_i} = x'_i\right) = \Pr\left(X'_{B_{\mathcal{S}'_j}} = y'_j\right) = q^{-m}, \forall i \in [r], j \in [t].$$

Proof: The quantities in the statement of the lemma satisfy the following system of equations

$$\left(M\left[[r] \cup \mathcal{S}', [r+c]\right] \otimes I_m\right) \begin{bmatrix} X_{p_1}^T & \cdots & X_{p_r}^T & X_{B_1}^T & \cdots & X_{B_c}^T \end{bmatrix}^T$$
$$= \begin{bmatrix} X_{p_1}'^T & \cdots & X_{p_r}'^T & X_{B_{\mathcal{S}'_1}}'^T & \cdots & X_{B_{\mathcal{S}'_t}}'^T \end{bmatrix}^T.$$

The vector $\begin{bmatrix} X_{p_1}^T & \cdots & X_{p_r}^T & X_{B_1}^T & \cdots & X_{B_c}^T \end{bmatrix}^T$ is uniform over $\mathcal{F}^{m(r+c)}$. Since the matrix $M[[r] \cup \mathcal{S}', [r+c]] \otimes I_m$ has full row rank equal to $m(r+t)$, the R.H.S. of the above equation is uniformly distributed over $\mathcal{F}^{m(r+t)}$, giving the first statement. The second statement is true by marginalization. ∎

Theorem 1 *The computation capacity of any sum-network constructed by the SUM-NET-CONS algorithm is at most* 1.

Proof: By the construction procedure, there is a terminal t_{p_i} which is connected to the sources s_{p_i} and $\{s_{B_j} : A(i,j) = 1\}$ through the edge e_i. By lemmas 1 and 2 we have that $H(\phi_i(X)) \geq m \log_2 q$ bits. From the definition of n the maximum amount of information transmitted on e_i is $n \log_2 q$ bits and that implies that $m \leq n$. ∎

Next, we show that the upper bound on the computation capacity exhibits a strong dependence on the characteristic of the field (denoted $\mathrm{ch}(\mathcal{F})$) over which the computation takes place.

Theorem 2 *Let A be a $(0,1)$-matrix of dimension $r \times c$ and suppose that we construct a sum-network corresponding to A using the SUM-NET-CONS algorithm. The matrix M_A is as defined in eq. (2.10). If $\mathrm{rank}_{\mathcal{F}} M_A = r + t$, the upper bound on computation capacity of the sum-network is $r/(r+t)$.*

Proof: Let $\mathcal{B}_{S'} \subseteq \mathcal{B}$ be as defined in lemma 2. Then from lemmas 1 and 2, we have $H\left(X'_{p_i}|\phi_i(X)\right) = 0,\ \forall i \in [r]$ and $H\left(X'_{B_{S'_j}}|\{\phi_i(X) : A(i,j) = 1\}\right) = 0,\ \forall j \in [t]$. Hence we have that $H(\{\phi_i(X)\}_1^r) \geq m(r+t)\log q$. From the definition of n, we get $nr \log q \geq H(\{\phi_i(X)\}_1^r) \geq m(r+t)\log q \implies m/n \leq r/(r+t)$. ■

Proposition 1 *We have that $\mathrm{rank}_{\mathcal{F}} M_A = r + t$ if and only if $\mathrm{rank}_{\mathcal{F}}\left((A^TA)_{\#} - A^TA\right) = t$. Furthermore, $\mathrm{rank}_{\mathcal{F}} M_A = r + c$ if and only if $\mathrm{ch}(\mathcal{F}) \nmid \det_{\mathbb{Z}} M_A$, where $\det_{\mathbb{Z}}$ indicates the determinant of the matrix with its elements interpreted as 0 or 1 in $\mathbb{Z}$.*

Proof: From eq. (2.10), we have that

$$M_A = \begin{bmatrix} I_r & A \\ A^T & (A^TA)_{\#} \end{bmatrix} = \begin{bmatrix} I_r & \mathbf{0} \\ A^T & I_c \end{bmatrix} \begin{bmatrix} I_r & \mathbf{0} \\ \mathbf{0} & (A^TA)_{\#} - A^TA \end{bmatrix} \begin{bmatrix} I_r & A \\ \mathbf{0} & I_c \end{bmatrix}, \tag{2.12}$$

which gives us the rank condition. Since M_A is a $(0,1)$-matrix, if it has full rank, then its determinant is some non-zero element of $\underline{\mathcal{F}}$, where $\underline{\mathcal{F}}$ is the base subfield of $\mathcal{F}$ having prime order. We could also interpret the elements of M_A as integers and evaluate its determinant $\det_{\mathbb{Z}} M_A$. Then if M_A has full rank, we have that $\mathrm{ch}(\underline{\mathcal{F}}) \nmid \det_{\mathbb{Z}} M_A$. ■

Example 4 *Consider the normal sum-network obtained from using the Fano plane for which the incidence matrix $A_{\mathcal{I}}$ is as defined in eq. (2.1), so that $r = c = 7$. It can be verified that $\mathrm{rank}_{GF(2)} M_{A_{\mathcal{I}}} = 7$. Hence theorem 2 gives an upper bound of 1 for the computation capacity. In fact, there is a rate-1 network code that satisfies all terminals in the normal sum-network obtained using the Fano plane as described later in proposition 4.*

We can obtain a different upper bound on the computation capacity by considering submatrices of M_A that do not necessarily contain all the initial r rows. To do this we define a new index set $\mathcal{S}''$ based on an index set $\mathcal{S} \subseteq [r]$ as follows.

$$\mathcal{S}'' \subseteq \{r+1, r+2, \ldots, r+c\} \text{ such that}$$
$$\forall i \in \mathcal{S}'', A^T[i-r, [r]] \in \text{Span}\{I_r[j, [r]] : j \in \mathcal{S}\}. \tag{2.13}$$

Here Span indicates the subspace spanned by the vectors in a set. The submatrix of M_A that contains all the rows indexed by numbers in $\mathcal{S} \cup \mathcal{S}''$ is $M[\mathcal{S} \cup \mathcal{S}'', [r+c]]$.

Theorem 3 *Let A be a $(0,1)$-matrix of dimension $r \times c$ and suppose that we construct a sum-network corresponding to A using the SUM-NET-CONS algorithm. For any (m,n)-network code that enables all the terminals to compute the sum, we must have that*

$$\frac{m}{n} \leq \min_{\mathcal{S} \subseteq [r]} \left\{ \frac{|\mathcal{S}|}{x_{\mathcal{S}}} \right\},$$

where $x_{\mathcal{S}} \coloneqq \text{rank}_{\mathcal{F}} M_A[\mathcal{S} \cup \mathcal{S}'', [r+c]]$ and $\mathcal{S}''$ is as defined in eq. (2.13).

Proof: Note that for the choice $\mathcal{S} = [r]$, the index set $\mathcal{S}''$ is the same as the index set $\mathcal{S}'$ defined in lemma 2 and $x_{\mathcal{S}} = \text{rank}_{\mathcal{F}} M_A$, thus recovering the $r/\text{rank}_{\mathcal{F}} M_A$ upper bound on the computation capacity from theorem 2. For $\mathcal{S} = \{\mathcal{S}_1, \ldots, \mathcal{S}_{|\mathcal{S}|}\} \subset [r]$, we have an index set $\mathcal{T} \subseteq \mathcal{S}''$ such that

$$\begin{aligned} x_{\mathcal{S}} &= \text{rank}_{\mathcal{F}} M_A[\mathcal{S} \cup \mathcal{S}'', [r+c]], \\ &= \text{rank}_{\mathcal{F}} M_A[\mathcal{S} \cup \mathcal{T}, [r+c]] = |\mathcal{S}| + |\mathcal{T}|. \end{aligned}$$

We collect the blocks indexed in increasing order by $\mathcal{T}$ in the set $\mathcal{B}_{\mathcal{T}} = \{B_{\mathcal{T}_1}, \ldots, B_{\mathcal{T}_y}\} \subseteq \mathcal{B}$, where $y \coloneqq |\mathcal{T}|$. Then one can recover the L.H.S. of the following equation from the set of messages $\{\phi_i(X) : i \in \mathcal{S}\}$

$$\begin{bmatrix} X'^T_{p\mathcal{S}_1} & \cdots & X'^T_{p\mathcal{S}_{|\mathcal{S}|}} & X'^T_{B_{\mathcal{T}_1}} & \cdots & X'^T_{B_{\mathcal{T}_y}} \end{bmatrix}^T = \left(\begin{bmatrix} M_A[\mathcal{S}, [r+c]] \\ M_A[\mathcal{T}, [r+c]] \end{bmatrix} \otimes I_m \right) \boldsymbol{X}.$$

Hence we have that $q^{n|\mathcal{S}|} \geq q^{m(|\mathcal{S}|+y)} \implies m/n \leq |\mathcal{S}|/x_{\mathcal{S}}$. The same reasoning is valid for any choice of $\mathcal{S} \subseteq [r]$ and that gives us the result. ∎

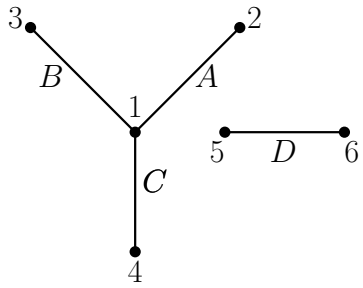


Figure 2.4: A simple undirected graph G with two connected components. It has 6 vertices and 4 edges.

Example 5 *Consider the transposed sum-network corresponding to the undirected graph G shown in Figure 2.4. One can check that the matrix $M_{A_G^T}$ when the rows and columns of the incidence matrix A_G^T are arranged in increasing alphabetical and numeric order is as follows.*

$$M_{A_G^T} = \left[\begin{array}{cccc|cccccc} 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ \hline 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{array}\right]$$

We choose our finite field alphabet to be $GF(3)$ in this example. Then $\text{rank}_{GF(3)}\, M_{A_G^T} = 5$ and theorem 2 gives that the computation capacity is at most $4/5$. However, theorem 3 gives a tighter upper bound in this case. Specifically, if $\mathcal{S} = \{1, 2, 3\}$ then $\mathcal{S}'' = \{5, 6, 7, 8\}$ and $\text{rank}_{GF(3)}\, M_{A_G^T}[\mathcal{S} \cup \mathcal{S}'', [10]] = 4$. Hence theorem 3 states that the computation capacity of the transposed sum-network for the graph G is at most $3/4$.

We apply the above theorems to obtain characteristic dependent upper bounds on the computation capacity of some infinite families of sum-networks constructed using the given procedure.

Corollary 1 *Let $\mathcal{I} = (\mathcal{P}, \mathcal{B})$ be an incidence structure obtained from a simple undirected graph where $\mathcal{P}$ denotes the set of vertices and $\mathcal{B}$ consists of the 2-subsets of $\mathcal{P}$ corresponding to the edges. Let $\deg(p) \in \mathbb{Z}$ represent the degree of vertex $p \in \mathcal{P}$. The incidence matrix $A_{\mathcal{I}}$ has dimension $|\mathcal{P}| \times |\mathcal{B}|$. The computation capacity of the normal sum-network constructed using $A_{\mathcal{I}}$ is at most $\frac{|\mathcal{P}|}{|\mathcal{P}|+|\mathcal{B}|}$ for any finite field $\mathcal{F}$.*

Let $\mathcal{F}$ be the finite field alphabet of operation and define $\mathcal{P}' \subseteq \mathcal{P}$ as $\mathcal{P}' \coloneqq \{p : \mathrm{ch}(\mathcal{F}) \nmid (\deg(p) - 1), p \in \mathcal{P}\}$. Consider the set of edges $\mathcal{B}' \coloneqq \cup_{p \in \mathcal{P}'} \langle p \rangle$. The computation capacity of the transposed sum-network is at most $\frac{|\mathcal{B}'|}{|\mathcal{B}'|+|\mathcal{P}'|}$.

Proof: Recall that $\boldsymbol{B}_i^T$ is the i-th row of $A_{\mathcal{I}}^T$ for all $i \in [|\mathcal{B}|]$. Then the inner product over $\mathcal{F}$ between two rows is

$$\boldsymbol{B}_i^T \boldsymbol{B}_j = \begin{cases} 2 \pmod{\mathrm{ch}(\mathcal{F})}, & \text{if } i = j, \\ 1, & \text{if edges indexed by } i \text{ and } j \text{ have a common vertex,} \\ 0, & \text{otherwise.} \end{cases}$$

It can be observed that the matrix of interest, i.e., $(A_{\mathcal{I}}^T A_{\mathcal{I}})_{\#} - A_{\mathcal{I}}^T A_{\mathcal{I}} = -I_{|\mathcal{B}|}$ has full rank over every finite field.

The transposed sum-network for $\mathcal{I}$ is obtained by applying the SUM-NET-CONS algorithm on the $|\mathcal{B}| \times |\mathcal{P}|$ matrix $A_{\mathcal{I}}^T$, so that the parameters $r = |\mathcal{B}|, c = |\mathcal{P}|$. We apply theorem 3 by choosing the index set $\mathcal{S} \subseteq [|\mathcal{B}|]$ such that $\mathcal{S} = \{j : B_j \in \mathcal{B}'\}$. Defined this way, $|\mathcal{S}| = |\mathcal{B}'|$ and $\mathcal{S}''$ is obtained from $\mathcal{S}$ using eq. (2.13). We collect all the points corresponding to the rows in the submatrix $M_{A_{\mathcal{I}}^T}[\mathcal{S}'', [r + c]]$ in a set $\mathcal{P}_{\mathcal{S}''} \subseteq \mathcal{P}$. Note that $\mathcal{P}_{\mathcal{S}''}$ depends on the set of edges $\mathcal{B}'$. By definitions of $\mathcal{B}'$ and $\mathcal{S}''$, we have that $\mathcal{P}' \subseteq \mathcal{P}_{\mathcal{S}''}$. This is true because $\mathcal{B}'$ consists of all the edges that are incident to at least one point in $\mathcal{P}'$ while indices in the set $\mathcal{S}''$ correspond to all points that are not incident to any edge outside $\mathcal{B}'$. For instance, in Example 5 above, as $\mathcal{F} = GF(3)$, $\mathcal{P}' = \{1\}$. Then $\mathcal{B}' = \{A, B, C\}$ and $\mathcal{P}_{\mathcal{S}''} = \{1, 2, 3, 4\}$.

We now show that $\text{rank}_{\mathcal{F}}\, M_A[\mathcal{S} \cup \mathcal{S}^{''}] = |\mathcal{B}'| + |\mathcal{P}'|$ and that gives us the result using theorem 3. Recall that $\boldsymbol{p}_i$ denotes the i-th row of $A_{\mathcal{I}}$, which corresponds to the vertex p_i for all $i \in [|\mathcal{P}|]$. It follows that the inner product between $\boldsymbol{p}_i, \boldsymbol{p}_j$ over $\mathcal{F}$ is

$$\boldsymbol{p}_i \boldsymbol{p}_j^T = \begin{cases} \deg(p_i) \pmod{\text{ch}(\mathcal{F})}, & \text{if } i = j, \\ 1, & \text{if } \{i, j\} \in \mathcal{B}, \\ 0, & \text{otherwise.} \end{cases}$$

Because of the above equation, all the off-diagonal terms in the matrix $(A_{\mathcal{I}} A_{\mathcal{I}}^T)_{\#} - A_{\mathcal{I}} A_{\mathcal{I}}^T$ are equal to zero. We focus on the submatrix $M[\mathcal{S} \cup \mathcal{S}^{''}, [r+c]]$ obtained from eq. (2.12), letting $\mathcal{S}^{''}_{|\mathcal{B}|} = \{j - |\mathcal{B}| : j \in \mathcal{S}^{''}\}$ we get that

$$M[\mathcal{S} \cup \mathcal{S}^{''}, [r+c]] = \begin{bmatrix} I_{|\mathcal{B}|}[\mathcal{S}, \mathcal{S}] & \mathbf{0} \\ A_{\mathcal{I}}[\mathcal{S}^{''}_{|\mathcal{B}|}, \mathcal{S}] & I_{|\mathcal{P}|}\left[\mathcal{S}^{''}_{|\mathcal{B}|}, \mathcal{S}^{''}_{|\mathcal{B}|}\right] \end{bmatrix} \cdot \Lambda \cdot \begin{bmatrix} I_{|\mathcal{B}|} & A_{\mathcal{I}}^T \\ \mathbf{0} & I_{|\mathcal{P}|} \end{bmatrix},$$

where

$$\Lambda := \begin{bmatrix} I_{|\mathcal{B}|}[\mathcal{S}, [|\mathcal{B}|]] & \mathbf{0} \\ \mathbf{0} & \left((A_{\mathcal{I}} A_{\mathcal{I}}^T)_{\#} - A_{\mathcal{I}} A_{\mathcal{I}}^T\right)\left[\mathcal{S}^{''}_{|\mathcal{B}|}, [|\mathcal{P}|]\right] \end{bmatrix}.$$

By definition of $\mathcal{P}'$ the points in the set $\mathcal{P}_{\mathcal{S}^{''}} \setminus \mathcal{P}'$ are such that $\deg(p_i) - 1 \equiv 0 \pmod{\text{ch}(\mathcal{F})}$, i.e., the diagonal entry corresponding to those points in $(A_{\mathcal{I}} A_{\mathcal{I}}^T)_{\#} - A_{\mathcal{I}} A_{\mathcal{I}}^T$ in the matrix Λ is zero. Thus, Λ has exactly $|\mathcal{B}'| + |\mathcal{P}'|$ rows which are not equal to the all-zero row vector. The first and third matrices are invertible, and hence we get that $\text{rank}_{\mathcal{F}}\, M_A[\mathcal{S} \cup \mathcal{S}^{''}, [r+c]] = |\mathcal{B}'| + |\mathcal{P}'|$. ■

Corollary 2 *Let $\mathcal{I} = (\mathcal{P}, \mathcal{B})$ be a 2-$(v, k, 1)$ design. For the normal sum-network constructed using the $|\mathcal{P}| \times |\mathcal{B}|$ incidence matrix $A_{\mathcal{I}}$, the computation capacity is at most $\frac{|\mathcal{P}|}{|\mathcal{P}|+|\mathcal{B}|}$ if $\text{ch}(\mathcal{F}) \nmid (k-1)$. For the transposed sum-network constructed using $A_{\mathcal{I}}^T$, the computation capacity is at most $\frac{|\mathcal{B}|}{|\mathcal{P}|+|\mathcal{B}|}$ if $\text{ch}(\mathcal{F}) \nmid \frac{v-k}{k-1}$.*

Proof: We first describe the case of the transposed sum-network. From eq. (2.2) each point in a 2-$(v, k, 1)$ design is incident to $\rho = \frac{v-1}{k-1}$ blocks. Moreover any two points occur together in

exactly one block. Thus, we have the inner product over $\mathcal{F}$ as

$$\boldsymbol{p}_i \boldsymbol{p}_j^T = \begin{cases} \frac{v-1}{k-1} \pmod{\mathrm{ch}(\mathcal{F})}, & \text{if } j = i, \\ 1, & \text{otherwise.} \end{cases}$$

This implies that $A_{\mathcal{I}} A_{\mathcal{I}}^T - (A_{\mathcal{I}} A_{\mathcal{I}}^T)_{\#} = \left[\left(\frac{v-1}{k-1} - 1\right)\right] I_v = \left[\frac{v-k}{k-1}\right] I_v$ and setting its determinant non-zero gives the result.

For the normal sum-network, we argue as follows. Note that $\boldsymbol{B}_i^T \boldsymbol{B}_i = k \pmod{\mathrm{ch}(\mathcal{F})}$ for any i. Since any two points determine a unique block, two blocks can either have one point or none in common. Hence, for $i \neq j$, the inner product over $\mathcal{F}$ is

$$\boldsymbol{B}_i^T \boldsymbol{B}_j = \begin{cases} 1, & \text{if } B_i \cap B_j \neq \emptyset, \\ 0, & \text{otherwise.} \end{cases}$$

Then $A_{\mathcal{I}}^T A_{\mathcal{I}} - (A_{\mathcal{I}}^T A_{\mathcal{I}})_{\#} = [(k-1)]\, I_b$ and setting its determinant as non-zero gives the result.

■

Corollary 3 *Let $\mathcal{I} = (\mathcal{P}, \mathcal{B})$ be a t-(v, k, λ) design, for $t \geq 2$. From eq. (2.2), each point is present in $\rho := \lambda\binom{v-1}{t-1}/\binom{k-1}{t-1}$ blocks and the number of blocks incident to any pair of points is given by $b_2 := \lambda\binom{v-2}{t-2}/\binom{k-2}{t-2}$. Consider the transposed sum-network constructed using the incidence matrix $A_{\mathcal{I}}^T$ which has dimension $|\mathcal{B}| \times |\mathcal{P}|$. The computation capacity of the transposed sum-network is at most $\frac{|\mathcal{B}|}{|\mathcal{B}|+|\mathcal{P}|}$ if*

$$\mathrm{ch}(\mathcal{F}) \nmid [\rho - b_2 + v(b_2 - 1)](\rho - b_2)^{v-1}.$$

Proof: By definition, we have that the inner product over $\mathcal{F}$ between two rows is

$$\boldsymbol{p}_i \boldsymbol{p}_j^T = \begin{cases} \rho \pmod{\mathrm{ch}(\mathcal{F})}, & \text{if } j = i, \\ b_2 \pmod{\mathrm{ch}(\mathcal{F})}, & \text{otherwise.} \end{cases}$$

It follows that $A_{\mathcal{I}} A_{\mathcal{I}}^T - (A_{\mathcal{I}} A_{\mathcal{I}}^T)_{\#}$ has the value $(\rho - 1)$ on the diagonal and $(b_2 - 1)$ elsewhere. Hence

$$A_{\mathcal{I}} A_{\mathcal{I}}^T - (A_{\mathcal{I}} A_{\mathcal{I}}^T)_{\#} = [(\rho - b_2) \pmod{\mathrm{ch}(\mathcal{F})}]\, I_v + [(b_2 - 1) \pmod{\mathrm{ch}(\mathcal{F})}]\, J_v,$$

where J_v denotes the square all ones matrix of dimension v. Then by elementary row and columns operations, $\det\left[A_{\mathcal{I}}A_{\mathcal{I}}^T - (A_{\mathcal{I}}A_{\mathcal{I}}^T)_{\#}\right]$ can be evaluated to be equal to $[\rho - b_2 + v(b_2-1)](\rho-b_2)^{v-1} \pmod{\mathrm{ch}(\mathcal{F})}$. ■

Corollary 4 *Let $\mathcal{D} = (\mathcal{P}, \mathcal{B})$ be a t-$(v, t+1, \lambda)$ design with $\lambda \neq 1$ and incidence matrix $A_{\mathcal{D}}$. We define a higher incidence matrix $A_{\mathcal{D}'}$ of dimension $\binom{|\mathcal{P}|}{t} \times |\mathcal{B}|$ such that each row corresponds to a distinct t-subset of $\mathcal{P}$ and each column corresponds to a block in $\mathcal{B}$. $A_{\mathcal{D}'}$ is a $(0,1)$-matrix such that for any $i \in \left[\binom{v}{t}\right], j \in [|\mathcal{B}|]$, its entry $A_{\mathcal{D}'}(i,j) = 1$ if each of the points in the t-subset corresponding to the i-th row is incident to the block $B_j \in \mathcal{B}$ and zero otherwise. The computation capacity of the normal sum-network constructed using $A_{\mathcal{D}'}$ is at most $\frac{\binom{v}{t}}{\binom{v}{t}+|\mathcal{B}|} = \frac{t+1}{\lambda+t+1}$ if $\mathrm{ch}(\mathcal{F}) \nmid t$. The computation capacity of the transposed sum-network constructed using $A_{\mathcal{D}'}^T$ is at most $\frac{|\mathcal{B}|}{|\mathcal{B}|+\binom{v}{t}} = \frac{\lambda}{\lambda+t+1}$ if $\mathrm{ch}(\mathcal{F}) \nmid (\lambda-1)$.*

Proof: The incidence matrix $A_{\mathcal{D}'}$ is a $(0,1)$ matrix of dimension $\binom{v}{t} \times \frac{\lambda}{t+1}\binom{v}{t}$. Let $\boldsymbol{p}_i, \boldsymbol{B}_u$ denote the i-th row and u-th column respectively of $A_{\mathcal{D}'}$ for $i \in \left[\binom{v}{t}\right], u \in \left[\frac{\lambda}{t+1}\binom{v}{t}\right]$. Each row of $A_{\mathcal{D}'}$ corresponds to a distinct t-subset of $\mathcal{P}$. By t-design criterion, any set of t points belongs to exactly λ blocks. Since the columns have a one-to-one correspondence with the blocks in $\mathcal{B}$, each row of $A_{\mathcal{D}'}$ has exactly λ 1's. Two rows will have a 1 in the same column if the block corresponding to the column is incident to both the t-subsets corresponding to the two rows. Since each block has $t+1$ points, there cannot be more than one block incident to two different t-subsets. Hence, for the inner product over $\mathcal{F}$, we have that $\boldsymbol{p}_i\boldsymbol{p}_i^T = \lambda \pmod{\mathrm{ch}(\mathcal{F})}$ and for all $i \neq j; i, j, \in \left[\binom{v}{t}\right]$,

$$\boldsymbol{p}_i\boldsymbol{p}_j^T = \begin{cases} 1, & \text{if the union of the } t\text{-subsets corresponding to} \\ & \text{the } i\text{-th and } j\text{-th rows is a block in } \mathcal{B}, \\ 0, & \text{otherwise.} \end{cases}$$

Then $A_{\mathcal{D}'}A_{\mathcal{D}'}^T - (A_{\mathcal{D}'}A_{\mathcal{D}'}^T)_{\#} = [(\lambda-1) \pmod{\mathrm{ch}(\mathcal{F})}]\, I_{\binom{v}{t}}$ and that gives the result for the transposed sum-network.

For the normal sum-network, we look at the columns of $A_{\mathcal{D}'}$ in a similar manner. Each column of $A_{\mathcal{D}'}$ corresponds to a block in $\mathcal{B}$. Since the size of each block is $t+1$, each column has exactly

$\binom{t+1}{t} = t+1$ elements as 1. Also, two different blocks can have at most t points in common, and only when that happens, will the two columns have a 1 in the same row. Hence, for the inner product over $\mathcal{F}$, we have that $\boldsymbol{B}_u^T \boldsymbol{B}_u = (t+1) \pmod{\mathrm{ch}(\mathcal{F})}$ and for all $u \neq v; u, v \in \left[\binom{v}{t}\right]$,

$$\boldsymbol{B}_u^T \boldsymbol{B}_v = \begin{cases} 1, & \text{if the } u\text{-th and } v\text{-th blocks have } t \text{ points in common,} \\ 0, & \text{otherwise.} \end{cases}$$

Then $A_{\mathcal{D}'}^T A_{\mathcal{D}'} - (A_{\mathcal{D}'}^T A_{\mathcal{D}'})_{\#} = t \pmod{\mathrm{ch}(\mathcal{F})} I_{\frac{\lambda}{t+1}\binom{v}{t}}$ and theorem 2 gives the result. ■

2.6 Linear network codes for constructed sum-networks

In this section, we propose linear network codes for the sum-networks constructed using the SUM-NET-CONS algorithm. Recall that the algorithm takes a $(0,1)$-matrix A that has r rows and c columns as its input. In Section 2.5, we demonstrated that the incidence matrix of certain incidence structures result in sum-networks whose capacity can be upper bounded (*cf.* Corollaries 1, 2, 4). We now demonstrate that under certain conditions, we can obtain network codes whose rate matches the corresponding upper bound. Thus, we are able to characterize the capacity of a large family of sum-networks.

We emphasize that random linear network codes that have been used widely in the literature for multicast code constructions are not applicable in our context. In particular, it is not too hard to argue that a random linear network code would result in each terminal obtaining a different linear function or subspace. Thus, constructing codes for these sum-networks requires newer ideas. We outline the key ideas by means of the following example.

Example 6 *Consider the sum-network shown in Figure 2.2a. The matrix $A_{\mathcal{I}}$ used in its construction is of dimension $r \times c$ where $r = 2, c = 1$ and is described in Example 2. It can be observed that $A_{\mathcal{I}}^T A_{\mathcal{I}} - \left(A_{\mathcal{I}}^T A_{\mathcal{I}}\right)_{\#} = 1$. Then theorem 2 states that the computation capacity of this sum-network is at most $2/3$. We describe a network code with $m = 2, n = 3$. The global encoding functions for the two bottleneck edges are shown in Table 2.1. Using the values transmitted, all three terminals can recover the sum in the following manner. t_1 receives the value of X_2 from the direct*

Table 2.1: The function values transmitted across e_1, e_2 in Figure 2.2a for a network code with rate $= 2/3$. Each message $X_1, X_2, X_{\{1,2\}}$ is a vector with 2 components, and $\phi_1(X), \phi_2(X)$ are vectors with 3 components each. A number within square brackets adjoining a vector indicates a particular component of the vector.

Component	$\phi_1(X)$	$\phi_2(X)$
1	$X_1[1] + X_{\{1,2\}}[1]$	$X_2[1] + X_{\{1,2\}}[1]$
2	$X_1[2] + X_{\{1,2\}}[2]$	$X_2[2] + X_{\{1,2\}}[2]$
3	$X_{\{1,2\}}[1]$	$X_{\{1,2\}}[2]$

edge (s_2, t_1) *while* t_2 *receives the value of* X_1 *from the direct edge* (s_1, t_2). *Then* t_1 *recovers the sum using the first two components of* $\phi_1(X)$ *while* t_2 *recovers the sum using the first two components of* $\phi_2(X)$. *Additionally,* $t_{\{1,2\}}$ *receives both* $\phi_1(X), \phi_2(X)$ *and can carry out the operation* $(X_1 + X_{\{1,2\}}) + (X_2 + X_{\{1,2\}}) - X_{\{1,2\}}$. *Thus, each terminal is satisfied.*

The network code in the example has the following structure. For each bottleneck edge, the first r components of the global encoding vector are the sum of all messages that are incident to that bottleneck. The remaining c components of the encoding vectors transmit certain components of messages observed at source nodes that correspond to columns in the matrix $A_{\mathcal{I}}$. In the example, $t_{\{1,2\}}$ received the first component of $X_{\{1,2\}}$ from $\phi_1(X)$ and the second component from $\phi_2(X)$. Thus it was able to recover the value of $X_{\{1,2\}}$, which it used in computing the demanded sum.

Our construction of network codes for sum-networks will have this structure, i.e., the first r components on a bottleneck edge will be used to transmit a *partial* sum of the messages observed at the sources that are connected to that bottleneck edge and the remaining c components will transmit portions of certain sources in an uncoded manner. For a given incidence matrix A, our first step is to identify (if possible) a corresponding non-negative integral matrix D of the same dimensions with the following properties.

- $D(i, j) = 0$ if $A(i, j) = 0$.
- Each row in D sums to r.
- Each column in D sums to c.

Under certain conditions on the incidence matrix A, we will show that D can be used to construct suitable network codes for the sum-networks under consideration.

The existence of our proposed network codes are thus intimately related to the existence of non-negative integral matrices that satisfy certain constraints. The following theorem [33, Corollary 1.4.2] is a special case of a more general theorem in [34] that gives the necessary and sufficient conditions for the existence of non-negative integral matrices with constraints on their row and column sums. We give the proof here since we use some ideas from it in the eventual network code assignment.

Theorem 4 *Let $R = (r_1, r_2, \ldots, r_m)$ and $S = (s_1, s_2, \ldots, s_n)$ be non-negative integral vectors satisfying $r_1 + \ldots + r_m = s_1 + \ldots + s_n$. There exists an $m \times n$ nonnegative integral matrix D such that*

$$\begin{aligned} &0 \leq D(i,j) \leq c_{ij}, \quad \forall i \in [m], \forall j \in [n], \\ &\sum_{j=1}^{n} D(i,j) = r_i, \quad \forall i \in [m], \text{ and} \\ &\sum_{i=1}^{m} D(i,j) = s_j, \quad \forall j \in [n] \end{aligned}$$

if and only if for all $I \subseteq [m]$ and $J \subseteq [n]$, we have that

$$\sum_{i \in I} \sum_{j \in J} c_{ij} \geq \sum_{j \in J} s_j - \sum_{i \notin I} r_i. \tag{2.14}$$

Proof: Consider a capacity-limited flow-network modelled using a bipartite graph on $m+n$ nodes. The left part has m nodes denoted as $x_i, \forall i \in [m]$ and the right part has n nodes denoted as $y_j, \forall j \in [n]$. For all i, j there is a directed edge (x_i, y_j) with capacity c_{ij}. There are two additional nodes in the flow-network, the source node S^* and terminal node T^*. There are directed edges (S^*, x_i) with capacity r_i for all $i \in [m]$ and directed edges (y_j, T^*) with capacity s_j for all $j \in [n]$. Let x_I be the set of all nodes in the left part whose indices are in I and let $y_{\bar{J}}$ be the set of all nodes in the right part whose indices are *not* in J. Consider a cut separating nodes in $\{S^*\} \cup x_I \cup y_{\bar{J}}$ from its complement. Let f^* be the value of the maximum S^*-T^* flow in this network. Then we

must have that for all possible choice of subsets $I \subseteq [m], J \subseteq [n]$,

$$\sum_{i \notin I} r_i + \sum_{(i,j):i \in I, j \in J} c_{ij} + \sum_{j \notin J} s_j \geq f^*. \tag{2.15}$$

In particular, suppose that $f^* = \sum_{j \in [n]} s_j$ in the flow-network. Substituting this in eq. (2.15), we get the condition that for all possible subsets $I \subseteq [m], J \subseteq [n]$,

$$\sum_{i \in I} \sum_{j \in J} c_{ij} \geq \sum_{j \in J} s_j - \sum_{i \notin I} r_i. \tag{2.16}$$

Note that by choosing all possible subsets I, J, we are considering every possible S^*-T^* cut in the network. Then by the maxflow-mincut theorem, the set of conditions of the form of eq. (2.16) for all I, J are not only necessary but also sufficient for the existence of a flow of value $f^* = \sum_{j \in [n]} s_j$ in the network.

A feasible flow with this value can be used to arrive at the matrix D as follows. We set the value of element $D(i, j)$ in the matrix to be equal to the value of the feasible flow on the edge (x_i, y_j) for all $i \in [m], j \in [n]$. It is easy to verify that the matrix D satisfies the required conditions. ■

Using the existence theorem for nonnegative integral matrices, we can obtain network codes for sum-networks constructed from certain incidence structures. The following theorem describes a set of sufficient conditions that, if satisfied by an incidence structure, allow us to construct a linear network code that has the same rate as the computation capacity of that sum-network. The proof of the theorem is constructive and results in an explicit network code.

Theorem 5 *Let $\mathcal{I} = (\mathcal{P}, \mathcal{B})$ be an incidence structure and let $A_{\mathcal{I}}$ denote the corresponding incidence matrix of dimension $v \times b$. Suppose that the following conditions are satisfied.*

- $A_{\mathcal{I}}^T A_{\mathcal{I}} - (A_{\mathcal{I}}^T A_{\mathcal{I}})_{\#} = \text{diag}(\mu_1, \ldots, \mu_b) \pmod{\text{ch}(\mathcal{F})}$, *where μ_i is a non-zero element of $\mathcal{F}$ $\forall i \in \{1, 2, \ldots, b\}$.*

- *There exists a matrix $D_{\mathcal{I}}$ with integer elements of the same dimension as $A_{\mathcal{I}}$ whose entries satisfy*

$$D_{\mathcal{I}}(i,j) = 0, \ \textit{if } A_{\mathcal{I}}(i,j) = 0, \tag{2.17}$$

$$\sum_{i=1}^{v} D_{\mathcal{I}}(i,j) = v, \ \textit{and} \tag{2.18}$$

$$\sum_{j=1}^{b} D_{\mathcal{I}}(i,j) = b. \tag{2.19}$$

Then the computation capacity of the sum-network constructed using $A_{\mathcal{I}}$ via the SUM-NET-CONS algorithm is $\frac{v}{v+b}$. This rate can be achieved by a linear network code.

Proof: Note that $A_{\mathcal{I}}^T A_{\mathcal{I}} - (A_{\mathcal{I}}^T A_{\mathcal{I}})_{\#}$ has full rank by assumption, theorem 2 states that the computation capacity of the sum-network is at most $v/(v+b)$. We construct a (m,n) linear network code with $m = v, n = v+b$ using the matrix $D_{\mathcal{I}}$. Since $m = v$, each message vector has v components. For a vector $t \in \mathcal{F}^v$, the notation $t[l_1 : l_2]$ for two positive integers $l_1, l_2 \in [v]$ denotes a $(l_2 - l_1 + 1)$ length vector that contains the components of t with indices in the set $\{l_1, l_1+1, \ldots, l_2\}$ in order. We need to specify the global encoding vectors $\phi_i(X)$ only for the bottleneck edges $e_i, i \in [v]$ as all the other edges in the network act as repeaters. The linear network code is such that the first v components of the vector transmitted along $e_i \ \forall i \in [v]$ is

$$\phi_i(X)[1:v] = X_{p_i} + \sum_{j:A_{\mathcal{I}}(i,j)=1} X_{B_j}.$$

By construction, each $t_{p_i} \forall i \in [v]$ is connected to the source nodes in $\{s_{p_{i'}} : i' \neq i\} \cup \{s_{B_j} : A_{\mathcal{I}}(i,j) = 0\}$ by direct edges. t_{p_i} can then compute the following value from the information received on the direct edges.

$$\sum_{i' \neq i} X_{p_i} + \sum_{j:A_{\mathcal{I}}(i,j)=0} X_{B_j}.$$

Adding the above value to $\phi_i(X)[1:v]$ enables t_{p_i} to compute the required sum. In what follows, we focus on terminals of the form $t_{B_j} \forall j \in [b]$.

Since $n = v + b$, each vector $\phi_i(X) \in \mathcal{F}^n$ has b components that haven't been specified yet. We describe a particular assignment for the b components on every $\phi_i(X), i \in [v]$ using the matrix $D_{\mathcal{I}}$ that enables each $t_{B_j} \forall j \in [b]$ to compute the sum.

Recall the bipartite flow network constructed in the proof of theorem 4. The nodes in the left part are denoted as $p_i \forall i \in [v]$ and the nodes in the right part are denoted as $B_j \forall j \in [b]$. There is an edge (p_i, B_j) if and only if $A_{\mathcal{I}}(i,j) = 1$. The flow on the edge (p_i, B_j) is denoted as $f(p_i, B_j)$ and its value is determined by $D_{\mathcal{I}}(i,j)$, i.e., $f(p_i, B_j) \coloneqq D_{\mathcal{I}}(i,j)$.

By constraints on the row and column sums of $D_{\mathcal{I}}$, we conclude that the value of the flow through any $p_i \forall i \in [v]$ is b and the value of the flow through any $B_j \forall j \in [b]$ is v. Without loss of generality, assume that $B_j = \{p_1, p_2, \cdots, p_{|B_j|}\}$. We can partition the v components of message vector X_{B_j} into $|B_j|$ parts such that the i-th partition contains $f(p_i, B_j)$ distinct components of X_{B_j}. Such a partitioning can be done for all message vectors $X_{B_j}, j \in [b]$. Then the flow $f(p_i, B_j)$ indicates that the vector $\phi_i(X)[v+1 : v+b]$ includes $f(p_i, B_j)$ uncoded components of X_{B_j}. Assigning such an interpretation to every edge in the flow-network is possible as the total number of components available in each $\phi_i(X)$ is b and that is also equal to the flow through the point p_i.

By construction, terminal t_{B_j} is connected to all bottleneck edges in the set $\{e_i : A_{\mathcal{I}}(i,j) = 1\}$. From the assignment based on the flow, t_{B_j} receives $f(p_i, B_j)$ distinct components of X_{B_j} from $\phi_i(X)$ for all $\{i : A_{\mathcal{I}}(i,j) = 1\}$. Since $\sum_{i=1}^{v} f(p_i, B_j) = v$, it can recover all v components of X_{B_j} in a piecewise fashion.

By adding the first v components transmitted on all the bottleneck edges that are connected to t_{B_j}, it can recover

$$\begin{aligned}
&\sum_{i:A_{\mathcal{I}}(i,j)=1} \phi_i(X)[1:v] \\
&= \sum_{i:A_{\mathcal{I}}(i,j)=1} X_{p_i} + \sum_{i:A_{\mathcal{I}}(i,j)=1} \sum_{l:A_{\mathcal{I}}(i,l)=1} X_{B_l}, \\
&= \sum_{i:A_{\mathcal{I}}(i,j)=1} X_{p_i} + \sum_{B_l \in \langle B_j \rangle} \boldsymbol{B}_j^T \boldsymbol{B}_l X_{B_l}.
\end{aligned}$$

Because of the condition that $A_{\mathcal{I}}^T A_{\mathcal{I}} - (A_{\mathcal{I}}^T A_{\mathcal{I}})_{\#} = \mathrm{diag}(\mu_1, \mu_2, \ldots, \mu_b)$, one can verify that

$$\sum_{B_l \in \langle B_j \rangle} \boldsymbol{B}_j^T \boldsymbol{B}_l X_{B_l} = (\mu_j + 1) X_{B_j} + \sum_{B_l \in \langle B_j \rangle \setminus B_j} X_{B_l}.$$

By the flow-based assignment, each t_{B_j} obtains the value of X_{B_j} in a piecewise manner. It can then carry out the following

$$\begin{aligned}
&\sum_{i: A_{\mathcal{I}}(i,j)=1} \phi_i(X)[1:v] - \mu_j X_{B_j} \\
&= \sum_{i: A_{\mathcal{I}}(i,j)=1} X_{p_i} + (\mu_j + 1) X_{B_j} + \sum_{B_l \in \langle B_j \rangle \setminus B_j} X_{B_l} - \mu_j X_{B_j}, \\
&= \sum_{p \in B_j} X_p + \sum_{B_l \in \langle B_j \rangle} X_{B_l}.
\end{aligned}$$

The messages not present in this partial sum, i.e., $\{X_p : p \notin B_j\} \cup \{X_B : B \notin \langle B_j \rangle\}$ are available at t_{B_j} through direct edges by construction. Hence, terminals that correspond to a column of $A_{\mathcal{I}}$ are also able to compute the required sum. ■

We illustrate the linear network code proposed above by means of the following example.

Example 7 *Consider the normal sum-network obtained from the undirected simple graph G shown in Figure 2.5a. A part of the sum-network is shown in Figure 2.5b. The 4×5 incidence matrix A_G satisfies the condition of theorem 4 and therefore has an associated matrix D_G with row-sum as 5 and column-sum 4 as shown below. The rows and columns of A_G are arranged in increasing numeric and alphabetical order.*

$$A_G = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}, \quad D_G = \begin{bmatrix} 2 & 0 & 0 & 2 & 1 \\ 2 & 3 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 3 \\ 0 & 0 & 3 & 2 & 0 \end{bmatrix}.$$

Using the matrix D_G, one can construct a structured linear network code with rate $= v/(v+b) = 4/9$ as shown in Table 2.2. One can check that it enables all the terminals to compute the required sum. The flow-network corresponding to D_G is shown in Figure 2.5c, and the messages corresponding to the flow on the solid edges are shown alongside the respective edge.

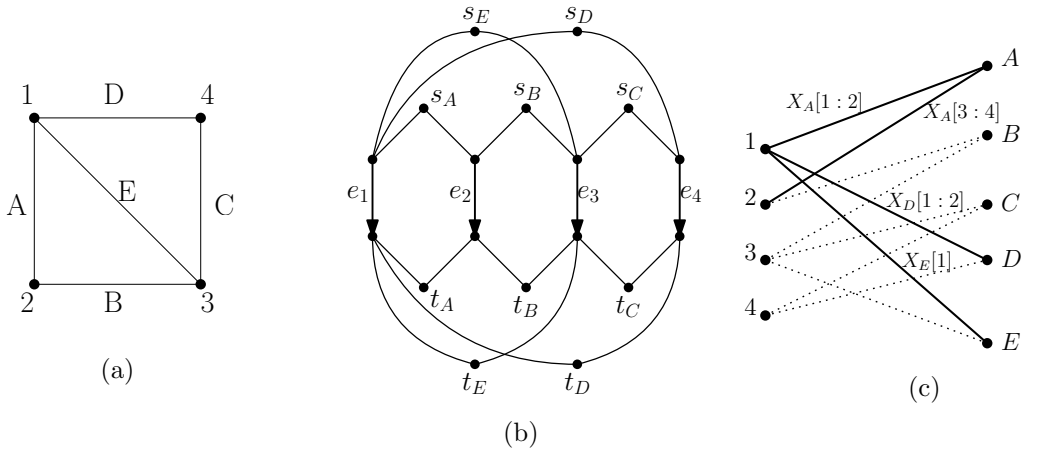


Figure 2.5: (a) Undirected graph considered in Example 7. (b) Part of the corresponding normal sum-network constructed for the undirected graph in (a). The full normal sum-network has nine nodes each in the source set S and the terminal set T. However, for clarity, only the five sources and terminals that correspond to the columns of the incidence matrix of the graph are shown. Also, the direct edges constructed in Step 4 of the construction procedure are not shown. All edges are unit-capacity and point downward. The edges with the arrowheads are the bottleneck edges constructed in step 2 of the construction procedure. (c) Bipartite flow network as constructed in the proof of theorem 4 for this sum-network. The message values corresponding to the flow on the solid lines are also shown.

We can also consider the transposed sum-network for the same graph G. Corollary 1 gives an upper bound on the computation capacity that depends on $\mathcal{F}$. If $\mathcal{F} = GF(2)$, then the subset of points $\mathcal{P}' = \{2, 4\}$ and the upper bound is $4/6$. Note that theorem 5 is not applicable here as the matrix $A_G^T A_G - (A_G^T A_G)_\#$ does not have all its diagonal elements as non-zero over $GF(2)$. Proposition 3 gives a condition for the existence of a network code for transposed sum-networks obtained using irregular graphs. We apply that condition to the transposed sum-network of the graph G considered here in Example 8.

In the following proposition we show that certain infinite families of incidence structures satisfy the requirements stated in theorem 5. In particular, the incidence structures considered in Corollaries 1, 2 and 4 satisfy the conditions and hence the computation capacity of the associated sum-networks can be calculated.

Table 2.2: The function values transmitted across e_1, e_2, e_3, e_4 in Figure 2.5b for a network code with rate = 4/9. Each message X_A, X_B, X_C, X_D, X_E is a vector with 4 components, and $\phi_1(X), \phi_2(X), \phi_3(X), \phi_4(X)$ are vectors with 9 components each. The number inside square brackets adjoining a vector indicates a particular component of the vector.

Component	$\phi_1(X)$	$\phi_2(X)$	$\phi_3(X)$	$\phi_4(X)$
1 *to* 4	$X_1 + X_A + X_D + X_E$	$X_2 + X_A + X_B$	$X_3 + X_B + X_C + X_E$	$X_4 + X_C + X_D$
5	$X_A[1]$	$X_A[3]$	$X_B[4]$	$X_C[2]$
6	$X_A[2]$	$X_A[4]$	$X_C[1]$	$X_C[3]$
7	$X_D[1]$	$X_B[1]$	$X_E[2]$	$X_C[4]$
8	$X_D[2]$	$X_B[2]$	$X_E[3]$	$X_D[3]$
9	$X_E[1]$	$X_B[3]$	$X_E[4]$	$X_D[4]$

Proposition 2 *The following incidence structures and their transposes satisfy condition (ii) in theorem 5, i.e., if their incidence matrix of dimension $v \times b$ is denoted by $A_{\mathcal{I}}$, there exists a corresponding non-negative integral matrix $D_{\mathcal{I}}$ that satisfies the conditions in equations (2.17) – (2.19).*

1. *Incidence structures derived from a regular graph or a biregular bipartite graph.*
2. *t-(v, k, λ) designs with $\lambda = 1$.*
3. *The higher incidence structure of a t-$(n, t+1, \lambda)$ design with $\lambda \neq 1$ obtained using the procedure described in corollary 4.*

Proof: The existence of $D_{\mathcal{I}}$ with row-sums as v and column-sums b is the same as the existence of $D_{\mathcal{I}}^T$ with row-sums as b and column-sums v. Thus, it suffices to argue for $D_{\mathcal{I}}$. To check the validity of the condition we first choose the bounds on the elements of the matrix $D_{\mathcal{I}}$. We set $r_i = b$ and $s_j = v$ for all $i \in [v], j \in [b]$ and

$$c_{ij} = \begin{cases} 0, & \text{if } A_{\mathcal{I}}(i,j) = 0, \\ \infty, & \text{if } A_{\mathcal{I}}(i,j) = 1. \end{cases}$$

By this choice the condition in inequality (2.14) is trivially satisfied whenever I, J are chosen such that there is a point in I which is incident to some block in J, i.e., there exist $i \in I, j \in J$ such that $A_{\mathcal{I}}(i,j) = 1$. Hence we restrict our attention to choices of I and J such that none of the

points in I are incident to any block in J. Under this restriction, the L.H.S. of inequality (2.14) is 0 and the condition is equivalent to $(v - |I|)b \geq |J|v$. We will assume that

$$\exists I \subseteq [v], J \subseteq [b] \text{ such that} \tag{2.20}$$
$$A_{\mathcal{I}}(i,j) = 0 \ \forall i \in I, j \in J, \text{ and } (v - |I|)b < |J|v,$$

and show that it leads to a contradiction for each of the three incidence structures considered.

If $\mathcal{I}$ corresponds to a d-regular simple graph, then $b = dv/2$. Consider the point-block incidence matrix $A_{\mathcal{I}}$, which is a $(0,1)$-matrix of size $v \times b$. For the chosen I in eq. (2.20), we look at the submatrix $A_{\mathcal{I}}[I,[b]]$ of size $|I| \times b$ that consists of the rows of $A_{\mathcal{I}}$ indexed by the points in I and all the columns. Let l_1 be the number of columns with a single 1 in $A_{\mathcal{I}}[I,[b]]$ and l_2 be the number of columns with two 1s in $A_{\mathcal{I}}[I,[b]]$. By counting the total number of 1s in $A_{\mathcal{I}}[I,[b]]$ in two ways, we get that

$$d|I| = l_1 + 2l_2 \leq 2(l_1 + l_2) \implies l_1 + l_2 \geq \frac{d|I|}{2}.$$

Since the number of edges incident to at least one point in I is $l_1 + l_2$, any subset J of the edges that has no incidence with any point in I satisfies $|J| \leq b - d|I|/2$. Using these in eq. (2.20) we get that

$$(v - |I|)b < |J|v \implies (v - |I|)\frac{dv}{2} < \left(\frac{dv}{2} - \frac{d|I|}{2}\right)v,$$

which is a contradiction.

Now suppose that $\mathcal{I}$ corresponds to a biregular bipartite graph, with L vertices having degree d_L in the left part and R vertices having degree d_R in the right part. Then $b = Ld_L = Rd_R$. Consider a subset $I_L \cup I_R$ of its vertices. Let E_L (resp. E_R) be the set of edges which are incident to some vertex in I_L (resp. I_R) but not incident to any vertex in I_R (resp. I_L). The number of edges that are not incident to any vertex in $I_L \cup I_R$ is equal to $(L - |I_L|)d_L - |E_R| = (R - |I_R|)d_R - |E_L|$. Suppose there is a choice of I in eq. (2.20) is such that $I = I_L \cup I_R$ for some I_L, I_R. Then we have

that

$$\begin{aligned}
&(v-|I|)b < |J|v,\\
&\implies (L+R-(|I_L|+|I_R|))\frac{Ld_L+Rd_R}{2}\\
&\quad < \frac{(L-|I_L|)d_L - E_R + (R-|I_R|)d_R - |E_L|}{2}(L+R),\\
&\implies \frac{|I_l|d_L+|I_R|d_R+|E_L|+|E_R|}{Ld_L+Rd_R} < \frac{|I_L|+|I_R|}{L+R},\\
&\implies (L+R)(|E_L|+|E_R|) < (L-R)|I_L|d_L+(R-L)|I_R|d_R,\\
&\implies (L+R)(|E_L|+|E_R|) < (L-R)(|E_L|-|E_R|).
\end{aligned}$$

If $L > R$ or $|E_L| > |E_R|$, then we have a contradiction. That leaves the case when $L < R$ and $|E_L| < |E_R|$, which implies $(L+R)(|E_L|+|E_R|) < (R-L)(|E_R|-|E_L|)$ and that is also a contradiction.

Next, consider a t-$(v,k,1)$ design with b blocks such that repetition degree of each point is ρ and we have that $bk = v\rho$. With the I of eq. (2.20), we employ a similar procedure as for the case of the d-regular graph. We choose the submatrix $A_{\mathcal{I}}[I,[b]]$ of size $|I|\times b$ that corresponds to the rows indexed by the points in I and let $l_i, \forall i \in [k]$ denote the number of columns with exactly i 1s in $A_{\mathcal{I}}[I,[b]]$. We count the total number of 1s in $A_{\mathcal{I}}[I,[b]]$ in two ways, yielding

$$\begin{aligned}
\rho|I| = l_1 + 2l_2 + \cdots + (k-1)l_{k-1} + kl_k \le k\sum_{i=1}^{k} l_i,\\
\implies \sum_{i=1}^{k} l_i \ge \frac{\rho|I|}{k} = \frac{b|I|}{v}.
\end{aligned}$$

The number of blocks that are incident to at least one point in I is equal to $\sum_{i=1}^{k} l_i$. Hence any subset J of blocks that has no incidence with any point in I satisfies $|J| \le b - |I|b/v$. Using this in eq. (2.20) we get that

$$(v-|I|)b < |J|v \implies (v-|I|)b < \left(b - \frac{|I|b}{v}\right)v,$$

which is a contradiction.

If $\mathcal{I} = (\mathcal{P},\mathcal{B})$ is the higher incidence structure obtained from a t-$(n,t+1,\lambda)$ design as described in corollary 4, then we have that $|\mathcal{P}| = \binom{n}{t}$ and $|\mathcal{B}| = \frac{\lambda}{t+1}\binom{n}{t}$. By definition of t for the original

design, we have that each of the points in $\mathcal{P}$ are incident to exactly λ blocks. Also, each block in $\mathcal{B}$ consists of $\binom{t+1}{t} = t+1$ points. For the submatrix $A_{\mathcal{I}}[I,[b]]$ whose rows correspond to the points in I from Condition 2.20, we let $l_i, \forall i \in [t+1]$ denote the number of columns that have exactly i 1s in them. By counting the total number of 1s in $A_{\mathcal{I}}[I,[b]]$ in two ways we get that

$$\lambda|I| = \sum_{i=1}^{t+1} i l_i \leq (t+1)\sum_{i=1}^{t+1} l_i \implies \sum_{i=1}^{t+1} l_i \geq \frac{\lambda|I|}{t+1}.$$

The total number of blocks incident to at least one point in I is $\sum_{i=1}^{t+1} l_i$. Then the number of blocks $|J|$ that are not incident to any point in I satisfy $|J| \leq |\mathcal{B}| - |I|\lambda/(t+1)$. Using these we get that

$$\begin{gathered}(v-|I|)b < |J|v, \\ \implies \left[\binom{n}{t} - |I|\right]\frac{\lambda}{t+1}\binom{n}{t} < \frac{\lambda}{t+1}\left[\binom{n}{t} - |I|\right]\binom{n}{t},\end{gathered}$$

which is a contradiction. Thus in all the three kinds of incidence structures considered, we have shown that they admit the existence of the associated matrix $D_{\mathcal{I}}$ under the stated qualifying conditions. This enables us to apply theorem 5 and obtain a lower bound on the computation capacity of these sum-networks. ■

For an undirected graph $\mathcal{I} = (\mathcal{P}, \mathcal{B})$ that is not regular, proposition 2 is not applicable. Theorem 5 describes a sufficient condition for the existence of a linear network code that achieves the upper bound on the computation capacity of normal sum-networks constructed from undirected graphs that are not necessarily regular. The upper bound on the capacity of the transposed sum-network constructed using the incidence matrix $A_{\mathcal{I}}^T$ however can be different from $\frac{|\mathcal{B}|}{|\mathcal{B}|+|\mathcal{P}|}$ depending on the finite field $\mathcal{F}$ (*cf.* corollary 1) and theorem 5 needs to be modified to be applicable in that case. The following example illustrates this.

Example 8 *Consider the transposed sum-network for the irregular graph G described in Example 7. Corollary 1 gives an upper bound of $4/6$ on the computation capacity when $\mathcal{F} = GF(2)$, as for that case $\mathcal{P}' = \{2,4\}$ and $\mathcal{B}' = \{A,B,C,D\}$. We show the submatrix $A_G^T[\mathcal{B}',\mathcal{P}']$ in the equation below and also an associated matrix D_G whose support is the same as that of $A_G^T[\mathcal{B}',\mathcal{P}']$ and whose row-sum $= 6-4 = 2$ and column-sum $= 4$. The rows and columns are arranged in increasing*

Table 2.3: The function values transmitted across the bottleneck edges of the transposed sum-network corresponding to the graph shown in Figure 2.5a for a rate-4/6 network over $GF(2)$. Each message X_2, X_4 is a vector with 4 components, and $\phi_A(X), \phi_B(X), \phi_C(X), \phi_D(X), \phi_E(X)$ are vectors with 6 components each. The number inside square brackets adjoining a vector indicates a particular component of the vector. A dash indicates that the value transmitted on that component is not used in decoding by any terminal.

Component	$\phi_A(X)$	$\phi_B(X)$	$\phi_C(X)$	$\phi_D(X)$	$\phi_E(X)$
1 *to* 4	$X_1 + X_2 + X_A$	$X_2 + X_3 + X_B$	$X_3 + X_4 + X_C$	$X_1 + X_4 + X_D$	$X_1 + X_3 + X_E$
5	$X_2[1]$	$X_2[3]$	$X_4[1]$	$X_4[3]$	–
6	$X_2[2]$	$X_2[4]$	$X_4[2]$	$X_4[4]$	–

alphabetical and numeric order.

$$A_G^T[\mathcal{B}', \mathcal{P}'] = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}, D_G = \begin{bmatrix} 2 & 0 \\ 2 & 0 \\ 0 & 2 \\ 0 & 2 \end{bmatrix}.$$

Using D_G we can construct a rate-4/6 linear network code, shown in Table 2.3, that achieves the computation capacity for $\mathcal{F} = GF(2)$ of the transposed sum-network constructed using the irregular graph G shown in Figure 2.5a. In particular, terminals t_1, t_3 don't need any information other than the partial sums obtained over their respective bottleneck edges to compute the sum. Terminals t_2, t_4 need the value X_2, X_4 respectively, and that is transmitted in a piecewise fashion according to the matrix D_G over the bottleneck edges.

For an undirected graph $\mathcal{I} = (\mathcal{P}, \mathcal{B})$ that is not regular, let $\mathcal{P}', \mathcal{B}'$ be the set of points and edges as chosen in the statement of corollary 1. We describe a condition on the submatrix $A_{\mathcal{I}}^T[\mathcal{B}', \mathcal{P}']$ which consists of the rows and columns of $A_{\mathcal{I}}^T$ corresponding to the blocks and points in the sets $\mathcal{B}', \mathcal{P}'$ respectively. This condition allows us to construct a capacity-achieving linear network code for the transposed sum-network.

Proposition 3 *For an undirected graph $\mathcal{I} = (\mathcal{P}, \mathcal{B})$, let $|\mathcal{P}'| = v', |\mathcal{B}'| = b'$, where $\mathcal{P}', \mathcal{B}'$ are subsets of points and blocks as defined in corollary 1 and let $A_{\mathcal{I}}^T[\mathcal{B}', \mathcal{P}'](i, j)$ indicate an element of*

the submatrix for indices $i \in [b'], j \in [v']$. *Suppose there is a matrix* $D_{\mathcal{I}}$ *of dimension* $b' \times v'$ *such that*

$$\begin{aligned} D_{\mathcal{I}}(i,j) &= 0, \text{ if } A_{\mathcal{I}}^T[\mathcal{B}', \mathcal{P}'](i,j) = 0, \\ \sum_{i=1}^{b'} D_{\mathcal{I}}(i,j) &= b', \text{ for all } j \in [v'], \text{ and} \\ \sum_{j=1}^{v'} D_{\mathcal{I}}(i,j) &= v', \text{ for all } i \in [b']. \end{aligned}$$

Then there is linear network code of rate $\frac{b'}{b'+v'}$ *that allows each terminal in the transposed sum-network constructed using* $\mathcal{I}$ *to compute the required sum.*

Proof: We describe a rate-$b'/(b'+v')$ network code that enables each terminal to compute the sum. Then by corollary 1 we know that this is a capacity-achieving code. Since this is a transposed sum-network, the bottleneck edges in the sum-network correspond to the blocks in the undirected graph $\mathcal{I}$. The first b' components transmitted over each bottleneck is obtained by the following equation.

$$\phi_i(X)[1:b'] = X_{B_i} + \sum_{j:p_j \in B_i} X_{p_j}, \text{ for all } B_i \in \mathcal{B}.$$

We show that this partial sum satisfies all the terminals in the set $\{t_{B_i} : B_i \in \mathcal{B}\} \cup \{t_{p_j} : p_j \notin \mathcal{P}'\}$. Terminals in $\{t_{B_i} : B_i \in \mathcal{B}\}$ can recover the sum as all messages not present in the partial sum are available to t_{B_i} through direct edges. For terminals in the set $\{t_p : p \notin \mathcal{P}'\}$, they carry out the following operation as a part of their decoding procedure.

$$\sum_{i:B_i \in \langle p \rangle} \phi_i(X)[1:b'] = \sum_{i:B_i \in \langle p \rangle} \left(X_{B_i} + \sum_{j:p_j \in B_i} X_{p_j} \right) \tag{2.21}$$

$$= \sum_{i:B_i \in \langle p \rangle} X_{B_i} + \sum_{j:\{p,p_j\} \in \mathcal{B}} \boldsymbol{p}\boldsymbol{p}_j^T X_{p_j} + \deg(p) X_p. \tag{2.22}$$

For $p_j \neq p$, we have that $\boldsymbol{p}\boldsymbol{p}_j^T = 1$ if $\{p, p_j\} \in \mathcal{B}$. Also by condition on the points that are not in $\mathcal{P}'$, we have that $\deg(p) \equiv 1 \pmod{\text{ch}(\mathcal{F})}$, and hence all the coefficients in the above partial sum are 1. The messages in the set $\{X_B : B \notin \langle p \rangle\} \cup \{X_{p_j} : \{p_j, p\} \notin \mathcal{B}\}$ are available to t_p through direct edges and hence it can recover the sum.

The remaining v' components available on the bottleneck edges $\{e_i : B_i \in \mathcal{B}'\}$ are used to transmit information that enable the terminals in the set $\{t_p : p \in \mathcal{P}'\}$ to compute the sum. Specifically, we construct a flow on a bipartite graph whose one part corresponds to the points in $\mathcal{P}'$ and the other part corresponds to the blocks in $\mathcal{B}'$, with incidence being determined by the submatrix $A_{\mathcal{I}}^T[\mathcal{B}', \mathcal{P}']$. Since there exists a matrix $D_{\mathcal{I}}$ with specified row and column sums, we can use it to construct a flow on the bipartite graph such that the messages in the set $\{X_{p_i} : p_i \in \mathcal{P}'\}$ are transmitted in a piecewise fashion over the bottleneck edges $\{e_j : B_j \in \mathcal{B}'\}$ in a manner similar to the proof of theorem 5. Arguing in the same way, one can show that the network code based on the flow solution allows each t_p $\forall p \in \mathcal{P}'$ to obtain the value of X_p from the information transmitted over the bottleneck edges in the set $\{e_i : B_i \in \langle p \rangle\}$. Terminal t_p computes the sum in eq. (2.21) as a part of its decoding procedure. Since $\deg(p) \not\equiv 1 \pmod{\mathrm{ch}(\mathcal{F})}$, every term in the RHS of eq. (2.22) except X_p has its coefficient as 1. But since t_p knows the value of X_p it can subtract a multiple of it and recover the relevant partial sum. The messages not present in this partial sum are available to t_p through direct edges and hence it can also compute the value of the sum. ■

Proposition 2 describes families of incidence structures for which the sum-networks constructed admit capacity-achieving linear network codes. The upper bound on the computation capacity of these sum-networks is obtained from Corollaries 1, 2 and 4. We now describe a rate-1 linear network code for the sum-networks when their corresponding incidence structures do not satisfy the qualifying conditions for the upper bounds. By theorem 1, the computation capacity of any sum-network obtained using the SUM-NET-CONS algorithm is at most 1.

Proposition 4 *For an incidence structure $\mathcal{I} = (\mathcal{P}, \mathcal{B})$ and a finite field $\mathcal{F}$, there exists a rate-1 linear network code that satisfies the following listed sum-networks. If*

- *$\mathcal{I}$ is a 2-$(v, k, 1)$ design:*
  - *the normal sum-network with* $\mathrm{ch}(\mathcal{F}) \mid k-1$,
  - *the transpose sum-network with* $\mathrm{ch}(\mathcal{F}) \mid \frac{v-k}{k-1}$,
- *$\mathcal{I}$ is a t-$(v, t+1, \lambda)$ design:*

– *the normal sum-network obtained using the higher incidence matrix with* $\mathrm{ch}(\mathcal{F}) \mid t$,

– *the transpose sum-network obtained using the higher incidence matrix with* $\mathrm{ch}(\mathcal{F}) \mid \lambda - 1$.

Proof: Suppose we construct a sum-network using the SUM-NET-CONS algorithm on a $(0,1)$-matrix A of dimension $r \times c$. If $A^T A = (A^T A)_{\#}$, the following rate-1 linear network code

$$\phi_i(X) = X_{p_i} + \sum_{j:B_j \in \langle p_i \rangle} X_{B_j}, \ \forall\ i \in [r],$$

satisfies every terminal in the sum-network in the following manner. A terminal t_{p_i}, $\forall i \in [r]$ receives all the messages not present in the partial sum transmitted along e_i through direct edges, and hence it can compute the sum. A terminal t_B, $\forall B \in \mathcal{B}$ can carry out the following operation.

$$\begin{aligned}\sum_{i:p_i \in B_j} \phi_i(X) &= \sum_{p_i \in B} X_{p_i} + \sum_{p_i \in B} \sum_{B_j \in \langle p_i \rangle} X_{B_j} \\ &= \sum_{p_i \in B} X_{p_i} + \sum_{l:B_l \in \langle B_j \rangle} \boldsymbol{B}_l^T \boldsymbol{B}_j X_{B_l}.\end{aligned}$$

Since $A^T A = (A^T A)_{\#}$, all the coefficients in the above sum are 1 and $\sum_{i:p_i \in B_j} \phi_i(X)$ is equal to the sum of all the messages in the set $\{X_{p_i} : p_i \in B_j\} \cup \{X_B : B \in \langle B_j \rangle\}$. All the messages that are not present in this set are available to t_{B_j} through direct edges.

Such a rate-1 linear network code gives us our proposition in the following manner. Let $A_{\mathcal{I}}$ be the $v \times \frac{v-1}{k-1}$ incidence matrix for a 2-$(v,k,1)$ design and let $A'_{\mathcal{I}}$ be the higher incidence matrix as defined in corollary 2 for a t-$(v, t+1, \lambda)$ design with $\lambda \neq 1$. Then, we have (from proofs of Corollaries 2, 4)

$$\begin{aligned}A_{\mathcal{I}}^T A_{\mathcal{I}} - (A_{\mathcal{I}}^T A_{\mathcal{I}})_{\#} &= (k-1)I, \\ A_{\mathcal{I}} A_{\mathcal{I}}^T - (A_{\mathcal{I}} A_{\mathcal{I}}^T)_{\#} &= \frac{v-k}{k-1} I, \\ A_{\mathcal{I}}'^T A'_{\mathcal{I}} - (A_{\mathcal{I}}'^T A'_{\mathcal{I}})_{\#} &= tI, \\ A'_{\mathcal{I}} A_{\mathcal{I}}'^T - (A'_{\mathcal{I}} A_{\mathcal{I}}'^T)_{\#} &= (\lambda - 1)I.\end{aligned}$$

Thus, whenever any of the above matrices is a zero matrix, we have a scalar linear network code that achieves the computation capacity of the associated sum-network. ■

2.7 Discussion and comparison with prior work

The discussion in Sections 2.5 and 2.6 establishes the computation capacity for sum-networks derived from several classes of incidence structures. We now discuss the broader implications of these results by appealing to existence results for these incidence structures. BIBDs have been the subject of much investigation in the literature on combinatorial designs. In particular, the following two theorems are well-known.

Theorem 6 *[21, Theorem 6.17] There exists a* $(v, 3, 1)$*-BIBD (also known as a Steiner triple system) if and only if* $v \equiv 1, 3 \pmod 6; v \geq 7$.

Theorem 7 *[21, Theorem 7.31] There exists a* $(v, 4, 1)$*-BIBD if and only if* $v \equiv 1, 4 \pmod{12}; v \geq 13$.

In particular, these results show that there are an infinite family of Steiner triple systems and BIBDs with block size 4 and $\lambda = 1$. Since $k = 3$ for any Steiner triple system, we can demonstrate the existence of sum-networks whose computation capacity is greatly affected by the choice of the finite field $\mathcal{F}$ used for communication.

Proposition 5 *Consider the normal sum-network constructed using a* 2-$(v, 3, 1)$ *design. If* $\mathrm{ch}(\mathcal{F}) = 2$*, then the computation capacity of the sum-network is* 1*. For odd* $\mathrm{ch}(\mathcal{F})$*, the computation capacity is* $\frac{6}{5+v}$*. For the normal sum-network constructed using a* $(v, 4, 1)$*-BIBD, the computation capacity is* 1 *if* $\mathrm{ch}(\mathcal{F}) = 3$ *and* $\frac{12}{11+v}$ *otherwise.*

Proof: The number of blocks in a 2-$(v, 3, 1)$ design is equal to $v(v-1)/6$. From corollary 2, if $\mathrm{ch}(\mathcal{F})$ is odd, then the computation capacity of the sum-network constructed using a Steiner triple system is at most $\frac{v}{v+v(v-1)/6} = \frac{6}{5+v}$. Moreover by proposition 2, we can construct a linear network code with rate equal to the upper bound. On the other hand, if $\mathrm{ch}(\mathcal{F}) = 2$, then the computation capacity of the same sum-network is 1 by proposition 4.

The number of blocks in a 2-$(v, 4, 1)$ design is $v(v-1)/12$. We can recover the result for the computation capacity of a normal sum-network constructed using it in a manner similar to the previous case. ■

Thus, this result shows that for the *same* network, computing the sum over even characteristic has capacity 1, while the capacity goes to zero as $O(1/v)$ for odd characteristic. Moreover, this dichotomy is not unique to the prime number 2. Similar results hold for sum-networks derived from higher incidence structures (*cf.* corollary 4).

Theorem 8 *[35] For two integers t, v such that $v \geq t+1 > 0$ and $v \equiv t \pmod{(t+1)!^{2t+1}}$, a t-$(v, t+1, (t+1)!^{2t+1})$ design with no repeated blocks exists.*

The number of blocks in a t-$(v, t+1, (t+1)!^{2t+1})$ design can be evaluated to be $\binom{v}{t}\frac{(t+1)!^{2t+1}}{t+1}$. We consider the normal sum-network obtained using the higher incidence matrix of this t-design. If $\mathrm{ch}(\mathcal{F}) \nmid t$, then by corollary 4 and proposition 2, we have that the computation capacity of this sum-network is

$$\frac{\binom{v}{t}}{\binom{v}{t} + \binom{v}{t}\frac{(t+1)!^{2t+1}}{t+1}} = \frac{1}{1 + t!^2(t+1)!^{2t-1}}.$$

On the other hand, if $\mathrm{ch}(\mathcal{F})$ is a divisor of t, then by theorem 1 and proposition 4 we have that the computation capacity of the normal sum-network constructed using the higher incidence matrix is 1. Thus for the same sum-network, computing the sum over a field whose characteristic divides the parameter t can be done at rate $= 1$. However, if the field characteristic does not divide t, zero-error computation of the sum can only be done at a rate which goes to zero as $O\left(\left(\frac{t}{e}\right)^{-t^2}\right)$.

Theorem 6 describes an infinite family of BIBDs with $k = 3$ and $\lambda = 1$. There are further existence results for BIBDs with $\lambda = 1$ and $k \neq 3$. In particular, for $\lambda = 1, k \leq 9$ there exist BIBDs with value of v as given in Table 3.3 in [36, Section II.3.1]. As an example, if $k = 5$, then there exists a 2-$(v, 5, 1)$ design whenever $v \equiv 1, 5 \pmod 2)0$. For any choice of a BIBD from this infinite family, we can construct a corresponding normal sum-network, whose computation capacity for a particular finite field can be found using corollary 2 and proposition 2. Even though theorem 4 states the existence of t-designs for v, t that satisfy the qualifying conditions, explicit constructions of such t-designs with $t \geq 6$ are very rare.

For a transposed sum-network obtained from an undirected graph that is not regular, the computation capacity can show a more involved dependence on the finite field alphabet as the following example demonstrates.

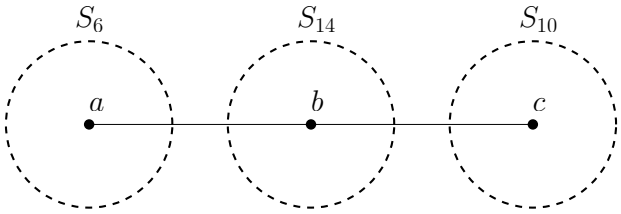


Figure 2.6: The schematic shown represents an undirected graph with three components: S_6, S_{14} and S_{10}. S_t denotes the star graph on $t+1$ vertices, with only one vertex having degree t while the rest have degree 1. The vertices with the maximum degree in the three star graphs are a, b, c respectively. In addition, a is connected to b and b is connected to c, such that $\deg(a) = 7, \deg(b) = 16, \deg(c) = 11$.

Example 9 *Consider the transposed sum-network obtained by applying the SUM-NET-CONS algorithm on the undirected graph $\mathcal{I}$ shown in Figure 2.6. Corollary 1 gives us an upper bound on the computation capacity of the transposed sum-network based on the finite field alphabet $\mathcal{F}$. The upper bound for three different choices of $\mathcal{F}$ is as follows.*

- *$\mathcal{F} = GF(2)$: Then $\mathcal{P}' = \{b\}$, so the upper bound is $16/(16+1) = 16/17$.*
- *$\mathcal{F} = GF(3)$: Then $\mathcal{P}' = \{c\}$, so the upper bound is $11/(11+1) = 11/12$.*
- *$\mathcal{F} = GF(5)$: Then $\mathcal{P}' = \{a\}$, so the upper bound is $7/(7+1) = 7/8$.*

We use proposition 3 to check if we can construct a linear network code in each case that has the same rate as the respective upper bound. To do that, we focus on the appropriate submatrix of $A_{\mathcal{I}}$ for each case and see if it satisfies the required condition on row and column sums. The rows of $A_{\mathcal{I}}$ corresponding to the vertices a, b, c (in order) are shown below.

$$\begin{bmatrix} \mathbf{1}_6 & 1 & 0 & \cdots & 0 \\ \mathbf{0}_6 & 1 & \mathbf{1}_{14} & 1 & \mathbf{0}_{10} \\ 0 & \cdots & 0 & 1 & \mathbf{1}_{10} \end{bmatrix},$$

where $\mathbf{1}, \mathbf{0}$ indicate all-one and all-zero row vectors of size specified by their subscripts. Using this, one can verify that the appropriate submatrix for each of the three choices of $\mathcal{F}$ satisfies the conditions of proposition 3 and hence we can construct a capacity-achieving linear network code in each case.

Thus, as the previous example demonstrates, the computation capacity of a particular sum-network need not take just one of two possible values, and can have a range of different values based on the finite field chosen. We can generalize the example to obtain sum-networks that have arbitrary different possible values for their computation capacity.

Our constructed sum-networks have a unit maximum flow between any source and any terminal. We can modify our construction so that each edge in the network has a capacity of $\alpha > 1$. Specifically, the following result can be shown.

Proposition 6 *Let $\mathcal{N}$ denote the sum-network obtained by applying the SUM-NET-CONS algorithm on a matrix A of dimension $r \times c$. For an integer $\alpha > 1$, let $\mathcal{N}_\alpha$ denote the sum-network obtained by modifying the SUM-NET-CONS algorithm such that $\mathcal{N}_\alpha$ has the same structure as $\mathcal{N}$ but each edge e_α in $\mathcal{N}_\alpha$ has $\text{cap}(e_\alpha) = \alpha > 1$. Then, if A satisfies the qualifying conditions in Theorems 2 and 5, the computation capacity of $\mathcal{N}_\alpha$ is $\frac{\alpha r}{r+c}$.*

Proof: Since A satisfies the conditions in theorem 5, there exists a (m, n) vector linear network code with $m = r, n = r + c$. For every unit-capacity edge in $\mathcal{N}$, we have α unit-capacity edges between the same tail and head in $\mathcal{N}_\alpha$. At the tail of every edge in $\mathcal{N}_\alpha$, we can apply the same network code except now we have α distinct edges on which we can transmit the encoded value. Thus we need transmit only $\frac{r+c}{\alpha}$ symbols on each of those edges. If $\frac{r+c}{\alpha}$ is not an integer, one can appropriately multiply both m, n with a constant. This modified network code has rate $= \frac{\alpha r}{r+c}$. Since A also satisfies the conditions in theorem 2, we have that an upper bound on the computation capacity of $\mathcal{N}$ is $r/(r+c)$. Applying the same argument on $\mathcal{N}_\alpha$, we get that an upper bound on the computation capacity of $\mathcal{N}_\alpha$ is $\frac{\alpha r}{r+c}$. This matches the rate of the modified vector linear network code described above. ■

This result can be interpreted as follows. Consider the class of sum-networks where the maximum flow between any source-terminal pair is at least α. Our results indicate, that for any α, we can always demonstrate the existence of a sum-network, where the computation capacity is strictly smaller than 1. Once again, this indicates the crucial role of the network topology in function computation.

2.7.1 Comparison with prior work

The work of Rai and Das [29] is closest in spirit to our work. In [29], the authors gave a construction procedure to obtain a sum-network with computation capacity equal to p/q, where p, q are any two co-prime natural numbers. The procedure involved first constructing a sum-network whose capacity was $1/q$. Each edge in this sum-network had unit-capacity. By inflating the capacity of each edge in the sum-network to $p > 1$, the modified sum-network was shown to have computation capacity as p/q.

Our work is a significant generalization of their work. In particular, their sum-network with capacity $1/q$ can be obtained by applying the SUM-NET-CONS algorithm to the incidence matrix of a complete graph on $2q - 1$ vertices [30]. We provide a systematic procedure for constructing these sum-networks for much larger classes of incidence structures.

In [29], the authors also posed the question if *smaller* sum-networks (with lesser sources and terminals) with capacity as p/q existed. Using the procedure described in this paper, we can answer that question in the affirmative.

Example 10 *The normal sum-network for the undirected graph in Figure 2.5a has computation capacity* $= 4/9$ *and has nine sources and terminals. To obtain a sum-network with the same computation capacity using the method described in [29] would involve constructing the normal sum-network for a complete graph on* 17 *vertices, and such a sum-network would have* 153 *source nodes and terminal nodes each.*

In [20], it was shown by a counter-example that for the class of sum-networks with $|S| = |T| = 3$, a maximum flow of 1 between each source-terminal pair was not enough to guarantee solvability (i.e., no network code of rate 1 exists for the counterexample). It can be observed that their counter-example is the sum-network shown in Figure 2.2a. Our characterization of computation capacity for a family of sum-networks provides significantly more general impossibility results in a similar vein. In particular, note that for the α-capacity edge version of a sum-network, the maximum flow between any source-terminal pair is at least α. Then suppose we consider the class of sum-networks

with $|S| = |T| = x = \beta(\beta+1)/2$ for some $\beta \in \mathbb{N}$. Consider a complete graph $K_\beta = (V, E)$ on β vertices; then $|V| + |E| = x$. Consider the sum-network obtained by applying the procedure on K_β, with each edge added having capacity as α. Then the computation capacity of this sum-network is $\alpha\beta/x$, which is less than 1 if $\alpha < (\beta+1)/2$. This implies that a max-flow of $(\beta+1)/2$ between each source-terminal pair is a necessary condition for ensuring all sum-networks with $|S| = |T| = x$ are solvable. When x cannot be written as $\beta(\beta+1)/2$ for some β, a similar argument can be made by finding an undirected graph $G = (V, E)$ (whose incidence matrix A_G satisfies the condition in theorem 5) such that $|V|$ is minimal and $|V| + |E| = x$.

2.8 Conclusions and future work

Sum-networks are a large class of function computation problems over directed acyclic networks. The notion of computation capacity is central in function computation problems, and various counterexamples and problem instances have been used by different authors to obtain a better understanding of solvability and computation capacity of general networks. We provide an algorithm to systematically construct sum-network instances using combinatorial objects called incidence structures. We propose novel upper bounds on their computation capacity, and in most cases, give matching achievable schemes that leverage results on the existence of non-negative integer matrices with prescribed row and column sums. We demonstrate that the dependence of computation capacity on the underlying field characteristic can be rather strong.

There are several opportunities for future work. Our proposed linear network codes for the constructed sum-networks require the correspondence incidence structures to have a specific property. In particular, our techniques only work in the case when $A^TA - (A^TA)_\#$ is a diagonal matrix. It would be interesting to find capacity achieving network codes in cases when $A^TA - (A^TA)_\#$ is not diagonal. More generally, it would be interesting to obtain achievability schemes and upper bounds for sum-networks with more general topologies.

CHAPTER 3. FUNCTION COMPUTATION ON A DIRECTED ACYCLIC NETWORK

[1] We study the rate region of variable-length source-network codes that are used to compute a function of messages observed over a network. The particular network considered here is the simplest instance of a directed acyclic graph (DAG) that is not a tree. Existing work on zero-error function computation in DAG networks provides bounds on the *computation capacity*, which is a measure of the amount of communication required per edge in the worst case. This work focuses on the average case: an achievable rate tuple describes the expected amount of communication required on each edge, where the expectation is over the probability measure of the source messages. We describe a procedure to obtain outer bounds to the rate region for computing an arbitrary demand function at the terminal. A key fact used to specify these outer bounds is the Schur-concave property of the entropy function. We evaluate these bounds for certain example demand functions. When the demand function is the finite field sum of messages from $GF(2)$, we show a network code that achieves the outer bound. When the demand function is the sum of three bit messages over the real numbers, there is a gap in the sum-rate between the outer bound and the network code that we use.

3.1 Introduction

Computing functions of data observed over a network is a well-motivated problem, and different frameworks addressing the problem have been studied in the literature. Broadly speaking, a general function computation problem can be modeled in the following manner. A directed acyclic graph (DAG) is used to model a communication network. Its vertices denote nodes of the network that are assumed to have unrestricted computational power and storage capacity. The edges denote one-way

[1] A part of this work was presented at the 2016 IEEE International Symposium on Information Theory.

communication links that can be thought of as noiseless *bit-pipes*. Each node can act as a decoder on the information received on its incoming edges and as an encoder for information transmitted on its outgoing edges. Some of the nodes in the network, called the *source* nodes, observe discrete-valued source *messages* that take values in a finite alphabet. The random process generating the source messages is assumed to be stationary and memoryless. There is a single *terminal* node that wishes to compute losslessly a discrete-valued function of all the source messages using the information it receives on its incoming edges. A solution to such a function computation problem will specify the communication carried out on each link and the information processing done at each node of the network. We are interested in finding the minimal communication required for solving a given problem instance. The amount of communication in a network can be specified by a *rate tuple* that has as many entries as the number of edges in the network. Each entry denotes the rate of the code employed on the corresponding edge in the network. As remarked in [37], this problem in its full generality encompasses several different areas in information theory, and as such, existing literature focuses on simplified versions that highlight different aspects of the problem.

If there is just one encoder and one decoder connected by a noiseless communication link, and the decoder wants to compute the identity function for the message, then it is a standard source coding problem[2]. When there is some coded side information available through an additional link at the decoder, then the optimal rate pair is given by the Ahlswede–Körner–Wyner solution [38, Thm. 10.2]. Extending this one-help-one solution to a two-help-one scenario is not optimal, as was demonstrated for a particular two-help-one problem instance by Körner and Marton [1].

Consider now the case of two encoders, connected by two separate links to a decoder which is interested in computing the identity function on the pair of source messages. The optimal rate pair for this distributed source coding problem is known to be the Slepian–Wolf rate region, and this solution can be extended to multiple encoders each of which are directly connected to the decoder via a separate link [38, Chap. 10].

[2]If the demand function is not identity, then it is a source coding problem for the random function value, as the function can be precomputed in the encoding process.

In reproducing the source messages at the terminal in the above scenarios, the coding schemes allow for an ϵ-block error that can be made as small as desired by choosing asymptotically large block lengths. Concurrently, work on zero-error source coding with side information was initiated by Witsenhausen [39]. With X being the source message and Y the side information, he defined a *confusability graph* G_X for X that tells us which realizations of X must be given distinct codewords by the encoder in order to attain zero-error. The optimum number of distinct codewords required was shown to be the graph chromatic number $\chi(G_X)$; for encoding multiple (say, k) instances it is $\chi(G_X^k)$ where G_X^k denotes the k-wise AND product graph of G_X. Thus when $\chi(G_X^k) < (\chi(G_X))^k$, block encoding multiple instances allows one to reduce the number of distinct codewords required. The above idea along with the probability information of X was used to find the expected number of bits that must be transmitted by the encoder in [40]. For the case when the codewords are restricted to be prefix-free, their expected length was shown in [40] to be within one bit of the chromatic entropy $H_\chi(G_X, X)$ of the probabilistic graph (G_X, X). When multiple instances are block encoded, the asymptotic per-instance expected codeword length is $\lim_{k\to\infty} \frac{1}{k} H_\chi(G_X^{\vee k}, X^k)$, where $G_X^{\vee k}$ is the k-wise OR product graph of G_X. This limit was shown to be the *graph entropy* of (G_X, X) in [40]. More information about the possible savings due to encoding multiple instances can be found in [41, Sec. XI].

In a function computing problem, the terminal would want to compute a general demand function of the messages and not necessarily the identity function. The work done in [2] considered computing a function on the setup of the source coding with side information problem, and allowed a vanishing block-error probability. They defined the conditional graph entropy $H(G_X, X|Y)$ for the probabilistic graph $(G_X, X|Y)$ of X given Y and showed that it is equal to the optimal number of bits per-instance of X that must be communicated by the encoder when asymptotically large block lengths are used. This was extended by the authors in [3] where they defined a conditional chromatic entropy of a probabilistic graph and showed that $\lim_{k\to\infty} \frac{1}{k} H_\chi(G_X^{\vee k}, X^k|Y^k) = H(G_X, X|Y)$. This gave a graph coloring procedure to obtain codes with rate close to the lower bound of [2].

Function computation was also approached in the distributed setting, where two encoders separately encode X and Y such that the decoder is able to compute $F(X,Y)$ losslessly. This scenario is closer to the source coding with side information problem than the distributed source coding problem because of the following. Any demand function can be computed at the decoder after communicating each of the source messages to it, thus the Slepian–Wolf region is an inner bound to the rate region of a function computation problem on the same network. Körner and Marton showed in [1] that if the decoder wants to decode just Z, then the rate tuple $(R_X, R_Y, R_Z) = (H(Z), H(Z), 0)$ is achievable for the case when $Z = X \oplus Y$, X and Y are a doubly symmetric binary source pair and $\oplus$ denotes modulo-2 sum. This rate tuple, with $R_Z = 0$, can be interpreted as the decoder wanting to compute the modulo-2 sum using the information obtained from the X and Y encoders. This view was taken by Han and Kobayashi in [42] where, subject to the constraint $\Pr(x,y) > 0, \forall (x,y) \in \mathcal{X} \times \mathcal{Y}$, they gave necessary and sufficient conditions on the demand function $F(X,Y)$ for which the Slepian–Wolf region for the source pair (X,Y) coincides with the rate region of the function computation problem. This gave a dichotomy in the class of bivariate functions that only depended on the $\mathcal{X} \times \mathcal{Y}$ function realization table and not on $\Pr(X,Y)$, except through the positivity constraint.

The authors in [3] also considered distributed function computation, where they showed that if $\Pr(X,Y)$ satisfied a restrictive 'zigzag' condition, then coloring the confusability graphs $G_X^{\vee k}$ and $G_Y^{\vee k}$ and then using Slepian–Wolf code is optimal for asymptotic block length. This sufficient condition was relaxed in [43] to a coloring connectivity condition by taking into account the function value. This condition was shown to characterize the rate region for function computation on one-stage tree-networks. It also gave an inner bound to the rate region of a general tree network. The rate region for a multi-stage tree-network when every source message at a vertex in the network satisfies a local Markovian property was characterized in [44]. The function computation scenarios described above all allowed for a vanishing block-error probability with asymptotically large block lengths.

In this paper, we study the problem of zero-error function computation over a simple DAG network that is not a tree, shown in Fig. 3.1. We assume that the three source messages are independent and the terminal wants to compute an arbitrary specified demand function of the messages. We allow for block encoding and decoding of multiple instances at the sources and the terminal. Even with the independence assumption, characterizing the rate region of this function computation problem is difficult as was observed in [49], where the problem of computing a particular *arithmetic sum* demand function on the same DAG network was considered. We describe works closely related to our problem next.

3.1.1 Related work

Zero-error function computation over a graphical network using network coding [4], [5] was studied in [45]. There they considered two variants of the communication load on the network, called *worst-case* and *average-case* complexity, depending on whether the probability information of the source messages was used or not. They characterized the rate region of achievable rate tuples that allowed zero-error function computation in tree-networks, each entry in a rate tuple was the rate of a code employed on the corresponding edge in the tree-network. They also made the observation that finding the rate region of a DAG network is significantly challenging because of multiple paths between a source node and the terminal, which allows for different ways of combining information at the intermediate nodes.

Zero-error function computation was also studied in [15], where the authors defined the *computation capacity*, which is a single number, of a function computation problem instance. This is a generalization of the coding capacity of a network (*c.f.* [8, Sec. VI], [9]), which is the supremum of the ratio $\frac{k}{n}$ over all achievable (k, n) fractional coding solutions for that communication network. A (k, n) fractional network code is one in which k source messages are block encoded at each encoder and every edge in the network transmits n symbols from the alphabet in one channel use. The authors in [15] characterized the computation capacity of multi-stage tree-networks by finding the necessary and sufficient amount of information that must be transmitted across all graph cuts that

separate one or more source nodes from the terminal. Upper bounds on the computation capacity of DAG networks are more complicated and have been obtained in [16], [47]. These upper bounds were shown to be unachievable for a function computation problem on a particular DAG in recent work [52, Sec. V]. An illuminating example in the search for improved upper bounds for function computation on DAG networks has been the problem of computing the arithmetic sum of three source bits over the network shown in Fig. 3.1. By counting the necessary and sufficient number of codewords required, the computation capacity for this problem was evaluated to be $\log_6 4$ in [15, Sec. V].

While the upper bounds for DAG networks described above hold for the worst-case scenario, it can be seen in the arithmetic sum example that we can do better in the average-case scenario by using the probability information of the source messages. For instance, suppose that the three source bits observed at s_1, s_2 and s_3 are equally likely to be 0 or 1, i.e., they are independent Bernoulli(1/2) random variables. By letting the number of symbols transmitted across the edges be a random variable N (instead of a fixed n for every k source bits as in the (k, n) fractional network code framework[3]), it can be seen that the computation rate $\frac{k}{\mathbb{E}N} = 0.8 > \log_6 4$ is achievable (*c.f.* Eg. 11). The work in [49] also gave an upper bound of 8/9 for the computation capacity of this arithmetic sum example in the framework of *variable-length* network codes.

It can be seen that an upper bound to the computation capacity corresponds to an outer bound to the rate region of a function computation problem. We use the framework of a *source-network* code as used for zero-error network coding in [50]. Correspondingly, the quantity of interest here is the zero-error function computation rate region, and we provide outer bounds to this rate region for computing an arbitrary specified demand function on the network shown in Fig. 3.1. We summarize our contributions below.

[3]As described in [49], for finding the computation capacity, it can be assumed w.l.o.g. that the message vector X_3^k is available to both the encoders at s_1 and s_2. Then the random variable N is defined as a stopping time w.r.t. the pair of random variables transmitted over the edges (s_1, t) and (s_2, t).

3.1.2 Main contributions

- We compute lower bounds on the rates that must be used on the edges of the DAG in Fig. 3.1 in order to compute with zero-error an arbitrary demand function of the messages at the terminal. This provides us an outer bound to the achievable rate region.
- The technique used for obtaining the lower bounds involved lower bounding the conditional entropy of the descriptions transmitted on the edges given the demand function value. This was done by first finding a family of probability mass functions (p.m.f.) that must necessarily contain the conditional p.m.f. of the descriptions transmitted by any valid source-network code for the problem. The required lower bound was then obtained by finding the entropy-minimizing p.m.f. in this family. This p.m.f. was found using the Schur-concave property of the entropy function.
- Computing the arithmetic sum of three bits over the DAG in Fig. 3.1 was considered in [49]. For that case, our procedure here gives a tighter lower bound for the sum-rate. We also demonstrate that our outer bound for the rate region is tight when the demand function is set to be the sum of three bits over $GF(2)$.

The paper is organized as follows. Section 3.2 describes the problem setup formally and motivates the use of variable-length network codes as defined in Section 3.2.1. Section 3.3 describes the procedure used to obtain outer bounds to the rate region for computing an arbitrary demand function over the network in Figure 3.1. Central to our approach are lower bounds to the conditional entropy of the descriptions transmitted, and these are described and illustrated using a running example in Section 3.3.1. We also consider two other example demand functions in Sections 3.3.3 and 3.3.4. Section 3.4 concludes the chapter and lists some avenues for future work.

3.2 Problem formulation

The edges in Figure 3.1 (later denoted by an ordered pair of vertices) have unit-capacity. We use the notation of a standard network code from [15]. In what follows, all logarithms denoted as log are

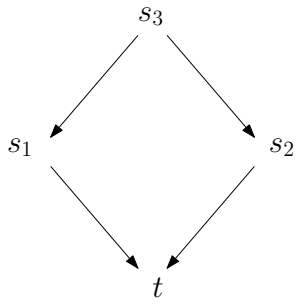


Figure 3.1: A directed acyclic network with three sources, two of which also act as relay nodes, and one terminal.

to the base 2 unless specified otherwise. Suppose that $\mathcal{Z}$ is the alphabet used for communication, and $|\mathcal{Z}| > 1$. Vertices s_1, s_2, s_3 are the three source nodes that observe source messages X_1, X_2, X_3 respectively, each from a discrete alphabet $\mathcal{A}$ with size $|\mathcal{A}| > 1$. The sources are assumed to be i.i.d. uniformly distributed over $\mathcal{A}$. Terminal node t wants to compute with zero error and zero distortion a known demand function:

$$f : \mathcal{A} \times \mathcal{A} \times \mathcal{A} \to \mathcal{B}$$

$$f(X_1, X_2, X_3) \mapsto B$$

where $\mathcal{B}$ is a discrete alphabet of size $|\mathcal{B}| > 1$. To avoid trivialities we assume that the demand function is not constant in any of its three arguments. A (k, n) network code that satisfies the terminal has the following components.

- An *encoding function* $h^{(e)}(\cdot)$ associated with every edge e in the network:

$$h^{(s_3,s_1)}(X_3^k) : \mathcal{A}^k \to \mathcal{Z}^n$$

$$h^{(s_3,s_2)}(X_3^k) : \mathcal{A}^k \to \mathcal{Z}^n$$

$$h^{(s_1,t)}\left(X_1^k, h^{(s_3,s_1)}(X_3^k)\right) : \mathcal{A}^k \times \mathcal{Z}^n \to \mathcal{Z}^n$$

$$h^{(s_2,t)}\left(X_2^k, h^{(s_3,s_2)}(X_3^k)\right) : \mathcal{A}^k \times \mathcal{Z}^n \to \mathcal{Z}^n$$

  Here the notation $X_j^k, j \in \{1, 2, 3\}$ denotes a block of k i.i.d. uniform *messages*.

- A *decoding function* $\psi(\cdot)$ used by the terminal t:

$$\psi\left(h^{(s_1,t)}(X_1^k, h^{(s_3,s_1)}(X_3^k)), h^{(s_2,t)}(X_2^k, h^{(s_3,s_2)}(X_3^k))\right) : \mathcal{Z}^n \times \mathcal{Z}^n \to \mathcal{B}^k$$

With slight abuse of notation we let $f(X_1^k, X_2^k, X_3^k) \in \mathcal{B}^k$ denote the k values returned by the demand function $f(X_1, X_2, X_3)$ when applied component-wise on a block of k i.i.d. (X_1, X_2, X_3) triples. That is,

$$f(X_1^k, X_2^k, X_3^k) = \left(f(X_1^{(1)}, X_2^{(1)}, X_3^{(1)}), f(X_1^{(2)}, X_2^{(2)}, X_3^{(2)}), \ldots, f(X_1^{(k)}, X_2^{(k)}, X_3^{(k)})\right),$$

where $X_j^{(i)}$ denotes the ith component of X_j^k. By the zero-error criterion we have that

$$\Pr\left\{\psi\left(h^{(s_1,t)}(X_1^k, X_3^k), h^{(s_2,t)}(X_2^k, X_3^k)\right) \neq f(X_1^k, X_2^k, X_3^k)\right\} = 0,$$

where the sample space consists of all realizations of the i.i.d. messages $X_j^k, j \in \{1, 2, 3\}$.

If such a set of encoding and decoding functions exist for a choice of positive integers k and n, we say that the network code *computes* the demand function and has *computation rate* $= k \log|\mathcal{A}|/(n \log|\mathcal{Z}|)$. The *computation capacity* for a particular demand function is defined to be the supremum of all achievable computation rates.

The above framework is restrictive in the sense that the block length of the network code used (i.e., the value of n) is the same for each edge in the network. If we know the probability distribution associated with the message random variables and allow the block length of the network code on edge e be a random variable N_e, then we can compress the descriptions transmitted on the edges and obtain savings in the expected length $\mathbb{E}[N_e]$.

Example 11 *Consider the problem of computing the sum over the real numbers (arithmetic sum) of three bits, i.e.* $f(X_1, X_2, X_3) = X_1 + X_2 + X_3$, *on the network shown in Figure 3.1. The messages* $X_1, X_2, X_3 \in \{0, 1\}^k$ *and* $f(X_1, X_2, X_3) \in \{0, 1, 2, 3\}^k$. *This example was first considered in [15], where they gave an optimal network code having computation rate* $\log_6 4 \approx 0.77$ *to solve it. The optimal network code transmits the value of* X_3 *on the edges* $(s_3, s_1), (s_3, s_2)$ *and the values*

transmitted on the other two edges are as follows, assuming block length k to be a multiple of 2.

$$h^{(s_1,t)}(X_1, X_3) = \begin{cases} X_1^{(i)} + X_3^{(i)}, & \text{for all } 1 \le i \le k/2 \\ X_1^{(i)}, & \text{otherwise.} \end{cases}$$

$$h^{(s_2,t)}(X_2, X_3) = \begin{cases} X_2^{(i)}, & \text{for all } 1 \le i \le k/2 \\ X_2^{(i)} + X_3^{(i)}, & \text{otherwise.} \end{cases}$$

The number of bits to be transmitted on the edges using the above encoding functions can be seen to be $k(1+\log 3)/2$. Now suppose that the messages are such that each $X_j^{(i)} \overset{i.i.d.}{\sim} Bern(0.5)$, i.e., they are equally likely random bits. Then both $X_1^{(i)} + X_3^{(i)}$ and $X_2^{(i)} + X_3^{(i)}$ in the encoding functions above have a biased distribution and can be compressed. The entropy $H(X_1^{(i)} + X_3^{(i)}) = 1.5$, and hence its ϵ-typical set [46] for $k/2$ instances can be enumerated using $1.5k/2$ bits. Thus the expected number of bits needed to be transmitted in this case can be seen to be

$$\mathbb{E}\, N_{(s_1,t)} = \mathbb{E}\, N_{(s_2,t)} = (1-\epsilon)(3k/4 + k/2) + \epsilon(k \log_2 3 + k/2) \approx 5k/4,$$

where the ϵ can be made small enough using large k. We note that $5/4 < (1+\log 3)/2$, and that an analogous definition for the computation rate of a network code in the variable-length framework would give that $k/\max\{\mathbb{E}[N_{(s_1,t)}], \mathbb{E}[N_{(s_2,t)}]\} = 0.8 > 0.77$.

Example 12 *Consider computing the maximum over the real numbers $f(X_1, X_2) = \max\{X_1, X_2\}$, where $X_1, X_2 \in \{0, 1\}$, over the reverse butterfly network shown in Figure 3.2. This problem was considered in [52], and they gave an upper bound of 2 for the computation rate of any valid network code. They also gave a valid network code with rate 3/2 that has the following edge functions, assuming block length k to be a multiple of 3.*

$$h^{(s_1,r_3)}(X_1^k) = (X_1^{(1)}, X_1^{(2)}, \ldots, X_1^{(k/3)}),$$
$$h^{(s_1,r_1)}(X_1^k) = (X_1^{(1+k/3)}, X_1^{(2+k/3)}, \ldots, X_1^{(k)}),$$
$$h^{(s_2,r_4)}(X_2^k) = (X_2^{(1)}, X_2^{(2)}, \ldots, X_2^{(k/3)}),$$
$$h^{(s_2,r_1)}(X_2^k) = (X_2^{(1+k/3)}, X_2^{(2+k/3)}, \ldots, X_1^{(k)}),$$

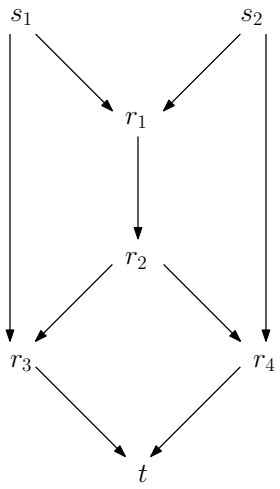


Figure 3.2: A directed acyclic network with two sources, four relay nodes and one terminal.

$$h^{(r_1,r_2)}(h^{(s_1,r_1)}(X_1^k), h^{(s_2,r_1)}(X_2^k)) = (\max\{X_1^{(1+k/3)}, X_2^{(1+k/3)}\}, \ldots, \max\{X_1^{(k)}, X_2^{(k)}\}),$$

$$h^{(r_2,r_3)}(h^{(r_1,r_2)}(h^{(s_1,r_1)}(X_1^k), h^{(s_2,r_1)}(X_2^k))) = (\max\{X_1^{(1+k/3)}, X_2^{(1+k/3)}\}, \ldots, \max\{X_1^{(2k/3)}, X_2^{(2k/3)}\}),$$

$$h^{(r_2,r_4)}(h^{(r_1,r_2)}(h^{(s_1,r_1)}(X_1^k), h^{(s_2,r_1)}(X_2^k))) = (\max\{X_1^{(1+2k/3)}, X_2^{(1+2k/3)}\}, \ldots, \max\{X_1^{(k)}, X_2^{(k)}\}),$$

$$h^{(r_3,t)}(h^{(s_1,r_3)}(\cdot), h^{(r_2,r_3)}(\cdot)) = (h^{(s_1,r_3)}(\cdot), h^{(r_2,r_3)}(\cdot)),$$

$$h^{(r_4,t)}(h^{(s_2,r_4)}(\cdot), h^{(r_2,r_4)}(\cdot)) = (h^{(s_2,r_4)}(\cdot), h^{(r_2,r_4)}(\cdot)),$$

where the last two equations state that the edges $(r_3, t), (r_4, t)$ *concatenate their inputs and forward it, and we have omitted the arguments of certain edge functions (represented by a dot) for brevity.*

Now consider the case when each $X_j^{(i)} \overset{i.i.d.}{\sim} Bern(0.75)$*, i.e., it takes the value* 1 *with probability* 0.75 *and* 0 *with probability* 0.25*. Since* $\max\{X_1^{(i)}, X_2^{(i)}\}$ *has a biased p.m.f., we are able to compress the descriptions transmitted on the edges. We describe a variable-length network code that has a similar structure, except that the partitioning of the* k *components of* X_1 *along the edges* (s_1, r_3) *and* (s_1, r_1) *is different from the* 1:2 *ratio used before, and it uses typical set encoding. For the* i*th component, the entropy values* $H(X_j^{(i)}) \approx 0.8113$ *and* $H(\max\{X_1^{(i)}, X_2^{(i)}\}) \approx 0.3373$ *can be verified. Set the value* $c \triangleq 1/(2 - 0.3373/(2 \cdot 0.8113)) \approx 0.558$*. Then the description transmitted on the edges*

(s_1, r_3) and (s_1, r_1) are as follows.

$$h^{(s_1,r_3)}(X_1) = \mathcal{T}_\epsilon[(X_1^{(1)}, X_1^{(2)}, \ldots, X_1^{(k-ck)})]$$
$$h^{(s_1,r_1)}(X_1) = \mathcal{T}_\epsilon[(X_1^{(1+k-ck)}, X_1^{(2+ck)}, \ldots, X_1^{(k)})],$$

where $\mathcal{T}_\epsilon[\cdot]$ indicates typical set encoding and a similar description of X_2 is transmitted on the edges (s_2, r_1) and (s_2, r_4). This partitioning is chosen so as to have the best possible rate within a class of network codes that have the same structure as the initial network code described, as the following computations indicate.

$$\mathbb{E}\, N_{(s_1,r_1)} = \mathbb{E}\, N_{(s_2,r_1)} = 0.8113 \cdot ck = 0.4527k,\ \mathbb{E}\, N_{(s_1,r_3)} = \mathbb{E}\, N_{(s_2,r_4)} = 0.8113 \cdot (1-c)k = 0.3586k,$$
$$\mathbb{E}\, N_{(r_1,r_2)} = 0.3373 \cdot ck = 0.1882k,\ \mathbb{E}\, N_{(r_2,r_3)} = \mathbb{E}\, N_{(r_2,r_4)} = 0.1882k/2 = 0.0941k,$$
$$\mathbb{E}\, N_{(r_3,t)} = \mathbb{E}\, N_{(r_4,t)} = 0.0941k + 0.3586k = 0.4527k.$$

Thus we have that $k/(\max_{all\ edges\ e} \mathbb{E}[N_e]) = k/(0.4527k) = 2.209 > 2$.

The above examples illustrate the reduction in communication load that can be achieved by using the knowledge of the probability distribution of the messages. The objective in this paper is to find bounds on rates of variable-length network codes that are valid for a given function computation problem. We first define the framework of variable-length network codes as adapted to the network in Figure 3.1 next.

3.2.1 Variable-length network code for network in figure 3.1

We use the *Source-Network Code* framework as described in [50] and adapt it to the function computation setting described above. The quantity of interest here is the rate region $\mathcal{R}$, which is a bounded region containing all achievable rate tuples $\boldsymbol{R}$. Each rate tuple has four components, one for each edge in the network. We define the source-network code and the admissible rate tuples below.

Definition 7 *Let $\mathcal{Z}^*$ denote the set of all finite-length sequences with alphabet $\mathcal{Z}$. A source-network code $\mathcal{C}_{f,k}$ for computing $f(X_1^k, X_2^k, X_3^k)$ in the network of Figure 3.1 has the following components:*

1. *Encoding functions for edges* $e \in \{(s_3, s_1), (s_3, s_2), (s_1, t), (s_2, t)\}$*:*

$$\phi_{(s_3,s_1)}(X_3^k) : \mathcal{A}^k \to \mathcal{Z}^*$$
$$\phi_{(s_3,s_2)}(X_3^k) : \mathcal{A}^k \to \mathcal{Z}^*$$
$$\phi_{(s_1,t)}\left(X_1^k, \phi_{(s_3,s_1)}(X_3^k)\right) : \mathcal{A}^k \times \mathcal{Z}^* \to \mathcal{Z}^*$$
$$\phi_{(s_2,t)}\left(X_2^k, \phi_{(s_3,s_2)}(X_3^k)\right) : \mathcal{A}^k \times \mathcal{Z}^* \to \mathcal{Z}^*$$

   For brevity, we denote $\phi_{(s_1,t)}\left(X_1^k, \phi^{(s_3,s_1)}(X_3^k)\right)$ *by the random variable* $\boldsymbol{Z}_1$*, and similarly define the r.v.s* $\boldsymbol{Z}_2, \boldsymbol{Z}_{31}, \boldsymbol{Z}_{32}$*.*

2. *Decoding function for terminal* t*:* $\psi_t : \mathcal{Z}^* \times \mathcal{Z}^* \to \mathcal{B}^k$ *is such that*

$$\Pr\{\psi_t(\boldsymbol{Z}_1, \boldsymbol{Z}_2) \neq f(X_1^k, X_2^k, X_3^k)\} = 0.$$

Thus the outputs of the encoders are variable length, and the terminal is equipped with a decoder that takes in a pair of variable length inputs and returns without any error the block of k function computations on the message tuple. The rate of a source-network code is defined below, taking into account the different alphabets in which the messages and the codewords reside.

Definition 8 $\boldsymbol{R} = (R_{31}, R_{32}, R_1, R_2)$ *is an admissible rate pair for the code* $\mathcal{C}_{f,k}$ *if for any* $\epsilon > 0$ *there exists a sufficiently large* k *such that*

$$\log|\mathcal{Z}|\,\mathbb{E}\,\ell(\boldsymbol{Z}_1) \leq k \log|\mathcal{A}|(R_1 + \epsilon)$$

where $\mathbb{E}\,\ell(\boldsymbol{Z}_1)$ *is the expected length (in symbols from* $\mathcal{Z}$*, and over the probability space of all message realizations) of the codeword* $\boldsymbol{Z}_1$*. A similar definition is used for the rates* R_{31}, R_{32} *and* R_2*.*

3.3 Bounds on the rate region for network in figure 3.1

We use a lower bound, reproduced from [51], on the expected length of the codewords transmitted on the edges in terms of their entropy.

Lemma 3 (Adapted from Theorem 3 in [51]) *The expected length of the best non-singular code $\mathcal{C}^{\star}_{NS}(\boldsymbol{Z})$ for a r.v. $\boldsymbol{Z}$ satisfies the following lower bound:*

$$\mathbb{E}\,\ell\left(\mathcal{C}^{\star}_{NS}(\boldsymbol{Z})\right) \geq H_{|\mathcal{Z}|}(\boldsymbol{Z}) - 2\log_{|\mathcal{Z}|}\left(H_{|\mathcal{Z}|}(\boldsymbol{Z}) + |\mathcal{Z}|\right).$$

Proof: Theorem 3 in [51] gives a lower bound for the expected length of a non-singular code over binary alphabet in terms of the entropy of the source. We adapt their proof procedure for codes over non-binary alphabet $\mathcal{Z}$.

Let $l_1 \leq l_2 \leq \cdots \leq l_{\max}$ be the lengths over $\mathcal{Z}$ of the best non-singular code for $\boldsymbol{Z}$. We demonstrate a function g such that the set of lengths $\{g(l_i) : i = 1, 2, \ldots\}$ satisfy Kraft's inequality, i.e., $\sum_{i=1}^{\infty} |\mathcal{Z}|^{-g(l_i)} \leq 1$. Choose $g(l_i) \triangleq l_i + 2\lfloor \log_{|\mathcal{Z}|}(l_i + |\mathcal{Z}| - 1)\rfloor$. Since there are $|\mathcal{Z}|^{l_i}$ different non-singular codewords with length l_i and if $l_{\max} > l_i$ then all of them must have been used in the best non-singular code, we have that

$$\begin{aligned}
\sum_i |\mathcal{Z}|^{-l_i - 2\lfloor \log_{|\mathcal{Z}|}(l_i+|\mathcal{Z}|-1)\rfloor} &= \sum_{l_1}^{l_{\max}} |\mathcal{Z}|^{l_i} |\mathcal{Z}|^{-l_i - 2\lfloor \log_{|\mathcal{Z}|}(l_i+|\mathcal{Z}|-1)\rfloor} \leq \sum_{l=1}^{\infty} |\mathcal{Z}|^{-(2\lfloor \log_{|\mathcal{Z}|}(l+|\mathcal{Z}|-1)\rfloor)} \\
&= \sum_{l=1}^{|\mathcal{Z}|^2-|\mathcal{Z}|} \frac{1}{|\mathcal{Z}|^2} + \sum_{l=|\mathcal{Z}|^2-|\mathcal{Z}|+1}^{|\mathcal{Z}|^3-|\mathcal{Z}|} \frac{1}{|\mathcal{Z}|^4} + \sum_{l=|\mathcal{Z}|^3-|\mathcal{Z}|+1}^{|\mathcal{Z}|^4-|\mathcal{Z}|} \frac{1}{|\mathcal{Z}|^6} + \cdots \\
&= \frac{|\mathcal{Z}|-1}{|\mathcal{Z}|} + \frac{|\mathcal{Z}|-1}{|\mathcal{Z}|^2} + \frac{|\mathcal{Z}|-1}{|\mathcal{Z}|^3} + \cdots = 1.
\end{aligned}$$

The above is also true if $l_{\max} \to \infty$. Thus there exists an uniquely decodable code for $\boldsymbol{Z}$ whose codeword lengths are $\{\lceil g(l_i)\rceil : i = 1, 2, \ldots\}$. Using random variable ℓ to denote the lengths of the codewords in the best non-singular code for $\boldsymbol{Z}$, we have that

$$\begin{aligned}
H_{|\mathcal{Z}|}(\boldsymbol{Z}) &\leq \mathbb{E}(\ell + 2\lfloor \log_{|\mathcal{Z}|}(\ell + |\mathcal{Z}| - 1)\rfloor) \leq \mathbb{E}\,\ell + 2\,\mathbb{E}\log_{|\mathcal{Z}|}(\ell + |\mathcal{Z}| - 1) \overset{\text{(i)}}{\leq} \mathbb{E}\,\ell + 2\log_{|\mathcal{Z}|}(\mathbb{E}\,\ell + |\mathcal{Z}| - 1) \\
&\leq \mathbb{E}\,\ell + 2\log_{|\mathcal{Z}|}(1 + H_{|\mathcal{Z}|} + |\mathcal{Z}| - 1) \implies \mathbb{E}\,\ell \geq H_{|\mathcal{Z}|}(\boldsymbol{Z}) - 2\log_{|\mathcal{Z}|}(|\mathcal{Z}| + H_{|\mathcal{Z}|}(\boldsymbol{Z})),
\end{aligned}$$

where inequality (i) above is true due to Jensen's inequality. ■

Since the identity mapping is also a non-singular code for $\boldsymbol{Z}$, we have that

$$\mathbb{E}\,\ell(\boldsymbol{Z}) = \sum_{\boldsymbol{z}} \Pr\{\boldsymbol{Z} = \boldsymbol{z}\}\ell(\boldsymbol{z}) \geq \mathbb{E}\,\ell\left(\mathcal{C}^{\star}_{\mathrm{NS}}(\boldsymbol{Z})\right). \tag{3.1}$$

Because of the zero-error requirement for the decoding function at the terminal, we can give a value for what the sum rate $R_{31} + R_{32}$ must be greater than.

Lemma 4 *Consider an equivalence relation on $\mathcal{A}^k$ for which $\boldsymbol{x}_3 \equiv \boldsymbol{x}'_3$ if and only if for all $(\boldsymbol{x}_1, \boldsymbol{x}_2) \in \mathcal{A}^k \times \mathcal{A}^k$, we have that $f(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3) = f(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}'_3)$. Define the function $g(X_3^k)$ which returns the equivalence class that X_3^k belongs to under the above relation. Then the range of $g(X_3^k)$ is a subset of $\{1, 2, \ldots, |\mathcal{A}|^k\}$ and we have that $R_{31} + R_{32} \geq H(g(X_3^k))/k \log |\mathcal{A}|$.*

Proof: Suppose that $H_{|\mathcal{Z}|}(g(X_3^k)|\boldsymbol{Z}_{31}, \boldsymbol{Z}_{32}) > 0$, then one cannot obtain $g(X_3^k)$ from the pair $(\boldsymbol{Z}_{31}, \boldsymbol{Z}_{32})$, i.e., there exist $\boldsymbol{x}_3 \not\equiv \boldsymbol{x}'_3$ but their associated codewords satisfy $\boldsymbol{z}_{31} = \boldsymbol{z}'_{31}$ and $\boldsymbol{z}_{32} = \boldsymbol{z}'_{32}$. There exists a pair $(\boldsymbol{x}_1, \boldsymbol{x}_2) \in \mathcal{A}^k \times \mathcal{A}^k$ such that $f(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3) \neq f(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}'_3)$. However, since $\boldsymbol{z}_{31} = \boldsymbol{z}'_{31}$ and $\boldsymbol{z}_{32} = \boldsymbol{z}'_{32}$, the codewords transmitted on the edges $(s_1, t), (s_2, t)$ in the two cases satisfy $\boldsymbol{z}_1 = \boldsymbol{z}'_1$ and $\boldsymbol{z}_2 = \boldsymbol{z}'_2$. Thus the decoder receives the same input arguments in both the cases and consequently causes an error.

Thus we have that $H_{|\mathcal{Z}|}(g(X_3^k)|\boldsymbol{Z}_{31}, \boldsymbol{Z}_{32}) = 0$. That gives us $H_{|\mathcal{Z}|}(g(X_3^k)) \leq H_{|\mathcal{Z}|}(\boldsymbol{Z}_{31}) + H_{|\mathcal{Z}|}(\boldsymbol{Z}_{32})$, and using the upper bound to the entropy in terms of the expected codeword length (*c.f.* equation (3.1) and lemma 3), we have the following.

$$
\begin{aligned}
& H_{|\mathcal{Z}|}(g(X_3^k)) \leq \mathbb{E}\,\ell(\boldsymbol{Z}_{31}) + 2\log_{|\mathcal{Z}|}(H_{|\mathcal{Z}|}(\boldsymbol{Z}_{31}) + |\mathcal{Z}|) + \mathbb{E}\,\ell(\boldsymbol{Z}_{32}) + 2\log_{|\mathcal{Z}|}(H_{|\mathcal{Z}|}(\boldsymbol{Z}_{32}) + |\mathcal{Z}|), \\
\implies & H(g(X_3^k)) \leq k(R_{31} + R_{32} + \epsilon) \log |\mathcal{A}|,
\end{aligned}
$$

the second inequality uses the definition of rate, and ϵ can be made small enough because $H_{|\mathcal{Z}|}(\boldsymbol{Z}_{31}) \leq H_{|\mathcal{Z}|}(X_3^k) = k$ and similarly for $H_{|\mathcal{Z}|}(\boldsymbol{Z}_{32})$. ■

Accordingly, in the rest of the paper, we focus on the quantities R_1 and R_2. Using inequality (3.1) and Lemma 3, we can conclude the following for the sum rate $R_1 + R_2$.

$$\begin{aligned}
\frac{\mathbb{E}\,\ell(\boldsymbol{Z}_1) + \mathbb{E}\,\ell(\boldsymbol{Z}_2)}{2} &\geq \frac{H_{|\mathcal{Z}|}(\boldsymbol{Z}_1) + H_{|\mathcal{Z}|}(\boldsymbol{Z}_2)}{2} - \log_{|\mathcal{Z}|}(H_{|\mathcal{Z}|}(\boldsymbol{Z}_1) + |\mathcal{Z}|) - \log_{|\mathcal{Z}|}(H_{|\mathcal{Z}|}(\boldsymbol{Z}_2) + |\mathcal{Z}|) \\
&\overset{(a)}{\geq} \left(H_{|\mathcal{Z}|}(f(X_1^k, X_2^k, X_3^k)) + H_{|\mathcal{Z}|}(\boldsymbol{Z}_1, \boldsymbol{Z}_2 | f(X_1^k, X_2^k, X_3^k))\right)/2 \\
&\quad - \log_{|\mathcal{Z}|}(H_{|\mathcal{Z}|}(\boldsymbol{Z}_1) + |\mathcal{Z}|) - \log_{|\mathcal{Z}|}(H_{|\mathcal{Z}|}(\boldsymbol{Z}_2) + |\mathcal{Z}|) \\
&\overset{(b)}{\geq} \left(H_{|\mathcal{Z}|}(f(X_1^k, X_2^k, X_3^k)) + H_{|\mathcal{Z}|}(\boldsymbol{Z}_1, \boldsymbol{Z}_2 | f(X_1^k, X_2^k, X_3^k), X_3^k)\right)/2 \\
&\quad - k\left(\log_{|\mathcal{Z}|}(H_{|\mathcal{Z}|}(\boldsymbol{Z}_1) + |\mathcal{Z}|) + \log_{|\mathcal{Z}|}(H_{|\mathcal{Z}|}(\boldsymbol{Z}_2) + |\mathcal{Z}|)\right)/k \\
\implies \frac{R_1 + R_2 + \epsilon}{2} &\geq \frac{\mathbb{E}\,\ell(\boldsymbol{Z}_1) + \mathbb{E}\,\ell(\boldsymbol{Z}_2)}{2k \log_{|\mathcal{Z}|} |\mathcal{A}|} \\
&\geq \frac{\alpha + H_{|\mathcal{Z}|}(f(X_1^k, X_2^k, X_3^k))/k}{2 \log_{|\mathcal{Z}|} |\mathcal{A}|} - \epsilon' \text{ for } \epsilon, \epsilon' > 0 \text{ and large } k, \qquad (3.2)
\end{aligned}$$

where inequality (a) is because $H(\boldsymbol{Z}_1, \boldsymbol{Z}_2, f(X_1^k, X_2^k, X_3^k)) = H(\boldsymbol{Z}_1, \boldsymbol{Z}_2)$ by the zero error criterion, inequality (b) is true as conditioning reduces entropy, and implication (3.2) is true by the value of α obtained in Section 3.3.2, by the definition of rate, and the fact that $H_{|\mathcal{Z}|}(\boldsymbol{Z}_1) \leq H(X_1^k, X_3^k) = 2k \log_{|\mathcal{Z}|} |\mathcal{A}|$.

Focusing on just $H_{|\mathcal{Z}|}(\boldsymbol{Z}_u)$ for either $u = 1$ or 2, we have the following inequality.

$$\begin{aligned}
\mathbb{E}\ell(\boldsymbol{Z}_u) &\geq H_{|\mathcal{Z}|}(\boldsymbol{Z}_u) - 2\log_{|\mathcal{Z}|}(H_{|\mathcal{Z}|}(\boldsymbol{Z}_u) + |\mathcal{Z}|) \\
&\overset{(a)}{\geq} H_{|\mathcal{Z}|}(\boldsymbol{Z}_u | X_3^k) - 2(\log_{|\mathcal{Z}|}(H_{|\mathcal{Z}|}(\boldsymbol{Z}_u) + |\mathcal{Z}|)) \overset{(b)}{\geq} \gamma k - 2k(\log_{|\mathcal{Z}|}(H_{|\mathcal{Z}|}(\boldsymbol{Z}_u) + |\mathcal{Z}|)/k) \\
\implies R_u + \epsilon &\geq \log_{|\mathcal{A}|} |\mathcal{Z}| \mathbb{E}\ell(\boldsymbol{Z}_u)/k \geq \gamma / \log_{|\mathcal{Z}|} |\mathcal{A}| - \epsilon' \quad \text{for large } k, \qquad (3.3)
\end{aligned}$$

where inequality (a) is true because conditioning reduces entropy. A procedure to obtain the value of γ in inequality (b) will be described in the next section (*c.f.* equation (3.5)).

3.3.1 Lower bound on the conditional entropy

In this section we use the structure of the demand function to obtain a lower bound on the conditional entropy of the descriptions transmitted on the edges (s_3, s_1) and (s_3, s_2). This enables us to find the values of α and γ used in the previous section. To do this, we define quantities

which denote the minimum number of distinct $(\boldsymbol{Z}_1, \boldsymbol{Z}_2)$-labels that allow the terminal to recover $f(X_1^k, X_2^k, X_3^k)$ with zero error. These quantities have been defined generally in [16], [47] and we adapt them to our particular network instance. Let lowercase letters $x_j, y_j \in \mathcal{A}$ denote realizations of the message random variable $X_j, j \in \{1, 2, 3\}$. For ease of notation, the ordering of the arguments of $f(X_1, X_2, X_3)$ is ignored and subscripts indicate which message random variable a particular realization corresponds to.

References [16], [47] describe a modification of a equivalence relation originally defined in [15] that is useful in obtaining upper bounds on the computation capacity of general directed acyclic networks. We apply those modified relations to our particular network instance.

Definition 9 *For two different message realizations* $(x_1, x_2, x_3), (y_1, y_2, y_3) \in \mathcal{A}^3$ *such that* $x_3 = y_3 \triangleq a_3$, *we say*[4] *that* $x_1 \overset{a_3}{\equiv} y_1|_1$ *if and only if* $f(x_1, x_2, a_3) = f(y_1, y_2, a_3)$ *for all* $x_2 = y_2 \in \mathcal{A}$. *Similarly, for two different message realizations* $(x_1, x_2, x_3), (y_1, y_2, y_3) \in \mathcal{A}^3$ *such that* $x_3 = y_3 \triangleq a_3$, *we say that* $x_2 \overset{a_3}{\equiv} y_2|_2$ *if and only if* $f(x_1, x_2, a_3) = f(y_1, y_2, a_3)$ *for all* $x_1 = y_1 \in \mathcal{A}$.

Note that for both $u = 1, 2$ and any value of $a_3 \triangleq x_3 = y_3$,

- $x_u = y_u$ implies $x_u \overset{a_3}{\equiv} y_u|_u$,
- $x_u \overset{a_3}{\equiv} y_u|_u$ implies $y_u \overset{a_3}{\equiv} x_u|_u$, and
- $x_u \overset{a_3}{\equiv} w_u|_u$ and $w_u \overset{a_3}{\equiv} y_u|_u$ implies $x_u \overset{a_3}{\equiv} y_u|_u$.

Thus $\overset{a_3}{\equiv} |_u$ is an equivalence relation on $\mathcal{A}$ for any demand function $f(X_1, X_2, X_3)$, choice of $u \in \{1, 2\}$ and $a_3 \in \mathcal{A}$. The number of equivalence classes of $\mathcal{A}$ induced by $\overset{a_3}{\equiv} |_u$ is denoted as $V_u(a_3)$ for both $u = 1$ and 2.

Illustration 1 *Throughout the paper we will use a running example to illustrate the various quantities defined. In the network of Figure 3.1 we choose the message alphabet to be the finite field of order* 3 *(denoted* $GF(3)$*) and the demand function*[5] *values are as listed in Table 3.1. From Table*

[4] *Read as 'x_1 is a_3-equivalent to y_1'.*

[5] *Any function from* $GF(3)^3 \to GF(3)$ *can be written as a multivariate polynomial in its arguments. Here it is* $X_1^2X_2^2X_3 - X_1^2X_2X_3^2 + X_1X_2^2X_3^2 + X_1^2X_2X_3 + X_1X_2^2X_3 + X_1^2X_3^2 - X_2^2X_3^2 + X_1^2X_3 + X_2^2X_3 + X_1X_3^2 - X_2X_3^2 - X_1X_3 - X_2X_3 - X_3^2 + X_1 - X_2 + X_3$.

Table 3.1: Function table for a demand function to be computed over the network in Figure 3.1. The message alphabet is $\mathcal{A} = GF(3)$. Table 3.1a shows the function values for all (X_1, X_2) pairs when $X_3 = 0$, table 3.1b shows the function values when $X_3 = 1$ and table 3.1c shows the function values when $X_3 = 2$.

$X_3 = 0$		X_2		
		0	1	2
X_1	0	0	2	1
	1	1	0	2
	2	2	1	0

(a)

$X_3 = 1$		X_2		
		0	1	2
X_1	0	0	0	0
	1	0	0	0
	2	1	0	0

(b)

$X_3 = 2$		X_2		
		0	1	2
X_1	0	1	1	0
	1	1	1	1
	2	1	1	1

(c)

3.1b we have that $0 \overset{1}{\equiv} 1|_1$ *and* $1 \overset{1}{\equiv} 2|_2$. *This is because for any value of* X_2, *the entries in the rows corresponding to* $X_1 = 0$ *and* $X_1 = 1$ *in Table 3.1b are the same, giving* $0 \overset{1}{\equiv} 1|_1$. *A similar statement is true for the columns corresponding to* $X_2 = 1$ *and* $X_2 = 2$ *in Table 3.1b. Since* $0 \overset{1}{\equiv} 1 \overset{1}{\not\equiv} 2|_1$, *we see that the relation* $\overset{1}{\equiv}|_1$ *partitions the alphabet* $\mathcal{A} = GF(3)$ *into two equivalence classes i.e.* $\{0, 1\}$ *and* $\{2\}$ *and hence* $V_1(1) = 2$. *From Table 3.1 one can verify the following equivalence classes of* $GF(3)$ *for each kind of relation in the considered demand function.*

$$\overset{0}{\equiv}|_1 : \{0\} \cup \{1\} \cup \{2\}, \quad \overset{1}{\equiv}|_1 : \{0, 1\} \cup \{2\}, \quad \overset{2}{\equiv}|_1 : \{0\} \cup \{1, 2\}, \tag{3.4a}$$

$$\overset{0}{\equiv}|_2 : \{0\} \cup \{1\} \cup \{2\}, \quad \overset{1}{\equiv}|_2 : \{0\} \cup \{1, 2\}, \quad \overset{2}{\equiv}|_2 : \{0, 1\} \cup \{2\}. \tag{3.4b}$$

Hence for this demand function we have that $V_1(0) = V_2(0) = 3, V_1(1) = V_2(1) = 2$ *and* $V_1(2) = V_2(2) = 2$.

Lemma 5 (Adapted from Lemma 3 in [47]) *The source node* s_1 *is the only source disconnected from the terminal if the edge* (s_1, t) *is removed from the network, even though* s_3 *has a path connecting it to the terminal through the edge* (s_1, t). *A similar statement also holds for the case of source node* s_2 *and edge* (s_2, t). *For two realizations* $(x_1, x_2, x_3), (y_1, y_2, y_3) \in \mathcal{A}^3$ *such that* $x_3 = y_3 = a_3$ *a valid network code must transmit different* $\boldsymbol{Z}_1$*-labels on the edge* (s_1, t) *if* $x_1 \overset{a_3}{\not\equiv} y_1|_1$ *and different* $\boldsymbol{Z}_2$*-labels labels on the edge* (s_2, t) *if* $x_2 \overset{a_3}{\not\equiv} y_2|_2$.

Thus, for a given realization a_3 of X_3, what matters is whether the function takes different values for different realizations of X_1 for some value of $X_2 \in \mathcal{A}$. If it does, then those two realizations of

X_1 belong to different $\overset{a_3}{\equiv}|_1$ equivalence classes and hence by Lemma 5 must have distinct labels transmitted over the edge (s_1, t). A similar argument can be made for the labels transmitted on the (s_2, t) edge based on the $\overset{a_3}{\equiv}|_2$ equivalence class of $X_2 \in \mathcal{A}$. For either $u = 1$ or 2, the $\overset{a_3}{\equiv}_u$ relation has a natural extension to vector realizations $\boldsymbol{x}_u \in \mathcal{A}^k$. We then have the following lemma, whose proof is in Appendix B.

Lemma 6 *We use lowercase boldface to denote vectors whose length is inferred from the context henceforth, like* $\boldsymbol{a}_3 \triangleq \left(a_3^{(1)}, a_3^{(2)}, \ldots, a_3^{(k)}\right)$. *Consider a block of* k *independent realizations of* X_1, X_2 *and* X_3 *and let* $\boldsymbol{a}_3 \in \mathcal{A}^k$ *be the realization for* X_3^k. *Then for* $u \in \{1, 2\}$, *the number of distinct* $\boldsymbol{Z}_u$*-labels that must be transmitted on the edge* (s_u, t) *to allow the terminal to recover* $f(X_1^k, X_2^k, X_3^k)$ *with zero error is at least* $V_u(\boldsymbol{a}_3) \triangleq \prod_{i=1}^k V_u(a_3^{(i)})$. *Equivalently, let* $\overset{\boldsymbol{a}_3}{\equiv}|_u$ *denote the collection of equivalence relations of Definition 9 for each component of* $\boldsymbol{a}_3$. *In this notation, we have that*

$$\boldsymbol{x}_u \overset{\boldsymbol{a}_3}{\equiv} \boldsymbol{y}_u \Leftrightarrow x_u^{(j)} \overset{a_3^{(j)}}{\equiv} y_u^{(j)} \ \forall j \in \{1, 2, \ldots, k\}.$$

If $\boldsymbol{x}_1 \overset{\boldsymbol{a}_3}{\not\equiv} \boldsymbol{y}_1$, *then* $\phi_{(s_1,t)}(\boldsymbol{x}_1, \boldsymbol{a}_3) \neq \phi_{(s_1,t)}(\boldsymbol{y}_1, \boldsymbol{a}_3)$, *i.e., their* $\boldsymbol{Z}_1$ *labels must be different. An analogous statement can be seen to be true for the* $\boldsymbol{Z}_2$ *label as well.*

For either $u = 1$ or 2 and each $i \in \{1, 2, \ldots, k\}$, the equivalence classes under $\overset{a_3^{(i)}}{\equiv}|_u$ are denoted as $\mathrm{Cl}_u^{(j)}(a_3^{(i)})$, where the superscript $j \in \{1, 2, \ldots, V_u(a_3^{(i)})\}$ indexes the classes such that

$$|\mathrm{Cl}_u^{(1)}(a_3^{(i)})| \geq |\mathrm{Cl}_u^{(2)}(a_3^{(i)})| \geq \cdots \geq |\mathrm{Cl}_u^{(V_u(a_3^{(i)}))}(a_3^{(i)})|.$$

As described in the proof above, $\mathcal{A}^k$ can be partitioned into $V_u(\boldsymbol{a}_3) = \prod_{i=1}^k V_u(a_3^{(i)})$ partitions based on the value of $\boldsymbol{a}_3$ for each component. Thus the partitions of $\mathcal{A}^k$ under $\overset{\boldsymbol{a}_3}{\equiv}|_u$ can be represented using a index vector $\boldsymbol{v}$ having k components, each of which satisfies $v^{(i)} \in \{1, 2, \ldots, V_u(a_3^{(i)})\}$ and

$$\boldsymbol{x}_u \in \mathrm{Cl}_u^{(\boldsymbol{v})}(\boldsymbol{a}_3) \Leftrightarrow x_u^{(i)} \in \mathrm{Cl}_u^{(v^{(i)})}(a_3^{(i)}), \quad \forall i \in \{1, 2, \ldots, k\}.$$

Similar to the scalar case, we use a subscript $t \in \{1, 2, \ldots, V_u(\boldsymbol{a}_3)\}$ for the index vector $\boldsymbol{v}$ such that the equivalence classes under $\overset{\boldsymbol{a}_3}{\equiv}|_u$ satisfy

$$|\mathrm{Cl}_u^{(\boldsymbol{v}_1)}(\boldsymbol{a}_3)| \geq |\mathrm{Cl}_u^{(\boldsymbol{v}_2)}(\boldsymbol{a}_3)| \geq \cdots \geq |\mathrm{Cl}_u^{(\boldsymbol{v}_{V_u(\boldsymbol{a}_3)})}(\boldsymbol{a}_3)|.$$

From the definition, for every $t \in \{1, 2, \ldots, V_u(\boldsymbol{a}_3)\}$, we have that $\mathrm{Cl}_u^{(\boldsymbol{v}_t)}(\boldsymbol{a}_3) = \bigtimes_{i=1}^{k} \mathrm{Cl}_u^{(v_t^{(i)})}(a_3^{(i)})$, i.e., a cartesian product of k scalar equivalence classes and $|\mathrm{Cl}_u^{(\boldsymbol{v}_t)}(\boldsymbol{a}_3)| = \prod_{i=1}^{k} |\mathrm{Cl}_u^{(v_t^{(i)})}(a_3^{(i)})|$.

The following notation is useful in characterizing all valid probability mass functions for $\boldsymbol{Z}_u$ for both $u = 1, 2$ and also the pair label $(\boldsymbol{Z}_1, \boldsymbol{Z}_2)$. For any index i of a vector $\boldsymbol{p}$, we use $p^{[i]}$ to denote the ith component of $\boldsymbol{p}$ when it is arranged in non-increasing order. For two vectors $\boldsymbol{p}, \boldsymbol{q}$ of the same length l, the vector $\boldsymbol{p}$ is majorized by $\boldsymbol{q}$, denoted as $\boldsymbol{p} \prec \boldsymbol{q}$, if the following holds.

$$\sum_{i=1}^{t} p^{[i]} \leq \sum_{i=1}^{t} q^{[i]}, \text{ for all } t = 1, 2, \ldots, l-1, \text{ and}$$
$$\sum_{i=1}^{l} p^{[i]} = \sum_{i=1}^{l} q^{[i]}.$$

As an example, the vector [0.5 0.5] is majorized by [0.25 0.75]. Note that any vector $\boldsymbol{p}$ is majorized by itself.

Lemma 6 gives a lower bound on the number of distinct $\boldsymbol{Z}_1$ and $\boldsymbol{Z}_2$ labels that must be used by a source-network code for a particular realization of X_3^k. Using this and the uniform i.i.d. assumption on the messages, we can characterize the set of valid p.m.f.s for the conditional probability of the $\boldsymbol{Z}_u$ label given a realization $\boldsymbol{a}_3$ of the message X_3^k.

Lemma 7 *Let $\mathbb{Z}_{\geq 0}$ denote the set of non-negative integers and superscript $\top$ indicates the transpose operation. Consider the partitions of $\mathcal{A}^k$ induced by the set of relations $\overset{\boldsymbol{a}_3}{\equiv}|_u$ where $\boldsymbol{a}_3$ is a realization of X_3^k and $u = 1$ or 2. Let $L_u(\boldsymbol{a}_3) \geq V_u(\boldsymbol{a}_3)$ be the number of distinct $\boldsymbol{z}_u$ labels assigned by an encoding scheme to the $|\mathcal{A}|^k$ different $(X_u^k, \boldsymbol{a}_3)$ pairs. Define a vector $\boldsymbol{d}_u(\boldsymbol{a}_3) \in \mathbb{Z}_{\geq 0}^{L_u(\boldsymbol{a}_3)}$ as*

$$\boldsymbol{d}_u(\boldsymbol{a}_3) \triangleq \left[|\mathrm{Cl}_u^{(\boldsymbol{v}_1)}(\boldsymbol{a}_3)| \quad |\mathrm{Cl}_u^{(\boldsymbol{v}_2)}(\boldsymbol{a}_3)| \quad \cdots \quad |\mathrm{Cl}_u^{(\boldsymbol{v}_{V_u(\boldsymbol{a}_3)})}(\boldsymbol{a}_3)| \quad \boldsymbol{0}_{L_u(\boldsymbol{a}_3)-V_u(\boldsymbol{a}_3)} \right]^\top,$$

where $\boldsymbol{0}_{L_u(\boldsymbol{a}_3)-V_u(\boldsymbol{a}_3)}$ indicates a zero vector of length $L_u(\boldsymbol{a}_3) - V_u(\boldsymbol{a}_3)$. Then any valid conditional p.m.f. $\boldsymbol{p} \in \mathbb{R}^{L_u(\boldsymbol{a}_3)}$ for $\boldsymbol{Z}_u$ given the value of $X_3^k = \boldsymbol{a}_3$ satisfies $\boldsymbol{p} \prec \boldsymbol{d}_u(\boldsymbol{a}_3)/|\mathcal{A}|^k$.

Proof: We first note that $\boldsymbol{d}_u(\boldsymbol{a}_3)/|\mathcal{A}|^k$ is a valid p.m.f. as its components are non-negative and sum up to 1. Suppose that there is an encoding scheme for the $\boldsymbol{Z}_u$ label such that $\Pr(\boldsymbol{Z}_u | X_3^k = \boldsymbol{a}_3) \triangleq \boldsymbol{p} \not\prec \boldsymbol{d}_u(\boldsymbol{a}_3)/|\mathcal{A}|^k$. Furthermore let $\boldsymbol{p}$ be supported on $L_u(\boldsymbol{a}_3)$ components. Then the

assumption implies that there is a $t < L_u(\boldsymbol{a}_3)$ such that

$$\sum_{j=1}^{t} p^{[j]} > \frac{1}{|\mathcal{A}|^k} \sum_{j=1}^{t} d_u^{[j]} = \frac{1}{|\mathcal{A}|^k} \sum_{j=1}^{t} |\mathrm{Cl}_u^{(\boldsymbol{v}_j)}(\boldsymbol{a}_3)|.$$

Since each realization of X_u^k is equally likely, the RHS in the above equation is the conditional probability given the value of $X_3^k = \boldsymbol{a}_3$ that a realization $\boldsymbol{x}_u$ belongs to one of the t largest equivalence classes under $\overset{\boldsymbol{a}_3}{\equiv}|_u$. The LHS is equal to the conditional probability of observing the t most probable $\boldsymbol{Z}_u$ labels. Hence the encoding scheme gives a total of t distinct $\boldsymbol{Z}_u$ labels to at least as many $(X_u^k, \boldsymbol{a}_3)$ pairs for which the realization $\boldsymbol{x}_u$ belongs to $t+1$ different equivalence classes under $\overset{\boldsymbol{a}_3}{\equiv}|_u$. This contradicts lemma 6. ■

In order to obtain a lower bound on $H(\boldsymbol{Z}_u|X_3^k = \boldsymbol{a}_3)$, we use the order-preserving property of the entropy function with respect to the majorization relation between two vectors [48]. The entropy function $H : \mathbb{R}^{L_u(\boldsymbol{a}_3)} \to \mathbb{R}$ is a strictly Schur-concave function [48, Chap. 3], i.e., for two p.m.f.s $\boldsymbol{p}, \boldsymbol{q} \in \mathbb{R}^{L_u(\boldsymbol{a}_3)}$ that are not equal to each other under any permutation of their components, we have that

$$\boldsymbol{p} \prec \boldsymbol{q} \implies H(\boldsymbol{p}) > H(\boldsymbol{q}).$$

If $\boldsymbol{p}$ and $\boldsymbol{q}$ are equal to each other under some permutation of their components, then obviously $H(\boldsymbol{p}) = H(\boldsymbol{q})$. Thus from lemma 7, we have that $H_{|\mathcal{Z}|}(\boldsymbol{Z}_u|X_3^k = \boldsymbol{a}_3) \geq H_{|\mathcal{Z}|}(\boldsymbol{d}_u(\boldsymbol{a}_3)/|\mathcal{A}|^k)$. The value of γ in equation (3.3) can be found as follows.

$$\begin{aligned} \gamma &\triangleq \frac{1}{k} \sum_{\boldsymbol{a}_3 \in \mathcal{A}^k} \Pr(X_3^k = \boldsymbol{a}_3) H_{|\mathcal{Z}|}(\boldsymbol{d}_u(\boldsymbol{a}_3)/|\mathcal{A}|^k) \\ &\leq \frac{1}{k} \sum_{\boldsymbol{a}_3 \in \mathcal{A}^k} \Pr(X_3^k = \boldsymbol{a}_3) H_{|\mathcal{Z}|}(\boldsymbol{Z}_u|X_3^k = \boldsymbol{a}_3) = \frac{H_{|\mathcal{Z}|}(\boldsymbol{Z}_u|X_3^k)}{k}. \end{aligned} \tag{3.5}$$

Illustration 2 *We evaluate the vector $\boldsymbol{d}_1(\boldsymbol{a}_3)$ for the example function and consequently obtain a lower bound for $H(\boldsymbol{Z}_1|X_3^k)$ and R_1 in this case. As shown in the previous illustration, $V_1(0) = 3$ and $V_1(1) = V_1(2) = 2$. Suppose X_3^k takes the value $\boldsymbol{a}_3$, where $\boldsymbol{a}_3$ has m_t components with value t for $t \in \{0, 1, 2\}$ such that $m_0 + m_1 + m_2 = k$. For each component $a_3^{(i)} \in \mathcal{A}, i \in \{1, 2, \ldots, k\}$ the*

scalar equivalence classes under $\overset{a_3^{(i)}}{\equiv}|_u$ *are given in equations* (3.4a), (3.4b), *and are represented as below.*

$$\mathrm{Cl}_u^{(1)}(0)=\{0\},\ \mathrm{Cl}_u^{(2)}(0)=\{1\},\ \mathrm{Cl}_u^{(3)}(0)=\{2\},\ \textit{for both } u=1,2 \textit{ and}$$
$$\mathrm{Cl}_1^{(1)}(1)=\{0,1\},\ \mathrm{Cl}_1^{(2)}(1)=\{2\},\ \mathrm{Cl}_1^{(1)}(2)=\{1,2\},\ \mathrm{Cl}_1^{(2)}(2)=\{0\},$$
$$\mathrm{Cl}_2^{(1)}(1)=\{1,2\},\ \mathrm{Cl}_2^{(2)}(1)=\{0\},\ \mathrm{Cl}_2^{(1)}(2)=\{0,1\},\ \mathrm{Cl}_2^{(2)}(2)=\{2\}.$$

Accordingly, the total number of partitions

$$V_1(\boldsymbol{a}_3)=\prod_{i=1}^{k}V_1(a_3^{(i)})=\left(\prod_{i:a_3^{(i)}=0}V_1(0)\right)\left(\prod_{i:a_3^{(i)}=1}V_1(1)\right)\left(\prod_{i:a_3^{(i)}=2}V_1(2)\right)=3^{m_0}2^{m_1+m_2}.$$

To find the value of $|\mathrm{Cl}_1^{(\boldsymbol{v}_1)}(\boldsymbol{a}_3)|$, *we pick the scalar partitions for each component that have the largest number of elements under* $\overset{0}{\equiv}|_1,\overset{1}{\equiv}|_1$ *and* $\overset{2}{\equiv}|_1$. *From the above equations we get* $|\mathrm{Cl}_1^{(v_1^{(i)})}(a_3^{(i)})|=2$ *if* $a_3^{(i)}\in\{1,2\}$ *and* $|\mathrm{Cl}_1^{(v_1^{(i)})}(a_3^{(i)})|=1$ *if* $a_3^{(i)}=0$. *Thus* $|\mathrm{Cl}_1^{(\boldsymbol{v}_1)}(\boldsymbol{a}_3)|=2^{m_1+m_2}$. *Furthermore, since there are* 3 *different largest equivalence classes under* $\overset{0}{\equiv}|_u$, *we get that*

$$|\mathrm{Cl}_1^{(\boldsymbol{v}_1)}(\boldsymbol{a}_3)|=|\mathrm{Cl}_1^{(\boldsymbol{v}_2)}(\boldsymbol{a}_3)|=\cdots=|\mathrm{Cl}_1^{(\boldsymbol{v}_{(3^{m_0})})}(\boldsymbol{a}_3)|=2^{m_1+m_2}.$$

Now, we find the number of realizations $\boldsymbol{x}_1$ *whose components do not necessarily belong to the largest scalar equivalence class at each component. If* $t\le m_1+m_2$ *components of* $\boldsymbol{x}_1$ *are such that either* $x_1^{(i)}=0\in\mathrm{Cl}_1^{(2)}(a_3^{(i)})$ *if* $a_3^{(i)}=2$ *or* $\boldsymbol{x}_1^{(1)}=2\in\mathrm{Cl}_1^{(2)}(a_3^{(i)})$ *if* $a_3^{(i)}=1$, *then the total number of* X_1^k *realizations that belong to the same equivalence class as* $\boldsymbol{x}_1$ *is* $2^{m_1+m_2-t}$. *Thus we have that*

$$\textit{for } t=1\textit{:}\qquad |\mathrm{Cl}_1^{(\boldsymbol{v}_{3^{m_0}+1})}(\boldsymbol{a}_3)|=\cdots=|\mathrm{Cl}_1^{(\boldsymbol{v}_{3^{m_0}+\binom{m_1+m_2}{1}3^{m_0}})}(\boldsymbol{a}_3)|=2^{m_1+m_2-1},$$
$$\textit{for } t=2\textit{:}\qquad |\mathrm{Cl}_1^{(\boldsymbol{v}_{3^{m_0}+\binom{m_1+m_2}{1}3^{m_0}+1})}(\boldsymbol{a}_3)|=\cdots=|\mathrm{Cl}_1^{(\boldsymbol{v}_{3^{m_0}+\binom{m_1+m_2}{1}3^{m_0}+\binom{m_1+m_2}{2}3^{m_0}})}(\boldsymbol{a}_3)|=2^{m_1+m_2-2},$$
$$\vdots$$
$$\textit{for } t=m_1+m_2\textit{:}|\mathrm{Cl}_1^{(\boldsymbol{v}_{3^{m_0}\sum_{l=0}^{m_1+m_2-1}\binom{m_1+m_2}{l}})}(\boldsymbol{a_3})|=\cdots=|\mathrm{Cl}_1^{(\boldsymbol{v}_{3^{m_0}\sum_{l=0}^{m_1+m_2}\binom{m_1+m_2}{l}})}(\boldsymbol{a}_3)|=2^0=1.$$

The above equations determine the components of the vector $\boldsymbol{d}_1(\boldsymbol{a}_3)$. *We can then evaluate the value of* $H_{|\mathcal{Z}|}(\boldsymbol{d}_1(\boldsymbol{a}_3)/3^k)$ *as*

$$
\begin{aligned}
&3^{m_0}\cdot\frac{1}{3^{m_0}}\left(\frac{2}{3}\right)^{m_1+m_2}\log_{|\mathcal{Z}|}\frac{3^k}{2^{m_1+m_2}}+3^{m_0}\binom{m_1+m_2}{1}\cdot\frac{1}{2\cdot 3^{m_0}}\left(\frac{2}{3}\right)^{m_1+m_2}\log_{|\mathcal{Z}|}\frac{3^k}{2^{m_1+m_2-1}}\\
&+3^{m_0}\binom{m_1+m_2}{2}\cdot\frac{1}{2^2\cdot 3^{m_0}}\left(\frac{2}{3}\right)^{m_1+m_2}\log_{|\mathcal{Z}|}\frac{3^k}{2^{m_1+m_2-2}}\\
&+\cdots+3^{m_0}\binom{m_1+m_2}{m_1+m_2}\cdot\frac{1}{2^{m_1+m_2}\cdot 3^{m_0}}\left(\frac{2}{3}\right)^{m_1+m_2}\log_{|\mathcal{Z}|}\frac{3^k}{2^0}\\
&=(k\log_{|\mathcal{Z}|}3-(m_1+m_2)\log_{|\mathcal{Z}|}2)\left(\frac{2}{3}\right)^{m_1+m_2}\left(1+\binom{m_1+m_2}{1}2^{-1}+\cdots+\binom{m_1+m_2}{m_1+m_2}2^{-m_1-m_2}\right)\\
&+\log_{|\mathcal{Z}|}2\left(\frac{2}{3}\right)^{m_1+m_2}\left(\binom{m_1+m_2}{1}2^{-1}+2\binom{m_1+m_2}{2}2^{-2}+\cdots+(m_1+m_2)\binom{m_1+m_2}{m_1+m_2}2^{-m_1-m_2}\right)\\
&=(k\log_{|\mathcal{Z}|}3-(m_1+m_2)\log_{|\mathcal{Z}|}2)\left(\frac{2}{3}\right)^{m_1+m_2}\left(\frac{3}{2}\right)^{m_1+m_2}+\log_{|\mathcal{Z}|}2\left(\frac{2}{3}\right)^{m_1+m_2}\frac{m_1+m_2}{3}\left(\frac{3}{2}\right)^{m_1+m_2}\\
&=k\log_{|\mathcal{Z}|}3-\frac{2(m_1+m_2)}{3}\log_{|\mathcal{Z}|}2.
\end{aligned}
$$

It follows that

$$
\begin{aligned}
H_{|\mathcal{Z}|}(\boldsymbol{Z}_1|X_3^k)&=\sum_{\boldsymbol{a}_3}\Pr\{X_3^k=\boldsymbol{a}_3\}H_{|\mathcal{Z}|}(\boldsymbol{Z}_1|X_3^k=\boldsymbol{a}_3)\\
&=\sum_{m_1,m_2=0,m_1+m_2\le k}^{k}\ \sum_{\boldsymbol{a}_3\ has\ m_1 1's,m_2 2's}\Pr\{X_3^k=\boldsymbol{a}_3\}H_{|\mathcal{Z}|}(\boldsymbol{Z}_1|X_3^k=\boldsymbol{a}_3)\\
&\ge\sum_{m_1,m_2=0,m_1+m_2\le k}^{k}\frac{k!}{3^k m_1!m_2!(k-m_1-m_2)!}\left(k\log_{|\mathcal{Z}|}3-\frac{2(m_1+m_2)}{3}\log_{|\mathcal{Z}|}2\right)\\
&=k\log_{|\mathcal{Z}|}3-\frac{2\log_{|\mathcal{Z}|}2}{3}\frac{2k3^{k-1}}{3^k}=k(\log_{|\mathcal{Z}|}3-\frac{4}{9}\log_{|\mathcal{Z}|}2).
\end{aligned}
$$

Thus the value of γ *is* $(\log_{|\mathcal{Z}|}3-\frac{4}{9}\log_{|\mathcal{Z}|}2)$ *and from equation* (3.3), $R_1+\epsilon\ge\log_{|\mathcal{A}|}3-\frac{4}{9}\log_{|\mathcal{A}|}2\approx 0.7196$.

To obtain a lower bound on the conditional entropy of the pair of labels as in (3.2), we characterize the family of valid conditional p.m.f.s for the pair $(\boldsymbol{Z}_1,\boldsymbol{Z}_2)$ given the values of the demand function $f(X_1^k,X_2^k,X_3^k)$ and the realization of the message X_3^k. We first find the number of distinct $(\boldsymbol{Z}_1,\boldsymbol{Z}_2)$-labels that must be assigned by the network code to message tuples that result in a particular value, say, $\boldsymbol{b}\in\mathcal{B}^k$ of the demand function $f(X_1^k,X_2^k,X_3^k)$. The set $A_3(\boldsymbol{b})$ has all possible

realizations $\boldsymbol{a}_3$ of X_3^k that can result in the value of $\boldsymbol{b}$ for the demand function, i.e.,

$$A_3(\boldsymbol{b}) \triangleq \{\boldsymbol{a}_3 \in \mathcal{A}^k : \exists\, \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{A}^k \text{ such that } f(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{a}_3) = \boldsymbol{b}\}.$$

Let $M(\boldsymbol{a}_3, \boldsymbol{b})$ denote the number of distinct $(\boldsymbol{Z}_1, \boldsymbol{Z}_2)$ pair labels used for input message tuples that have $X_3^k = \boldsymbol{a}_3$ and $f(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{a}_3) = \boldsymbol{b}$. Consider two message tuples $(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{a}_3)$ and $(\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{a}_3)$ which satisfy $f(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{a}_3) = f(\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{a}_3) = \boldsymbol{b}$. If either $\boldsymbol{x}_1 \overset{\boldsymbol{a}_3}{\not\equiv} \boldsymbol{y}_1|_1$ or $\boldsymbol{x}_2 \overset{\boldsymbol{a}_3}{\not\equiv} \boldsymbol{y}_2|_2$, then the pair of labels $(\boldsymbol{Z}_1, \boldsymbol{Z}_2)$ assigned to the two message tuples must be different. This motivates us to define the pair index set:

$$\mathcal{V}_{12}(\boldsymbol{a}_3, \boldsymbol{b}) \triangleq \left\{ (\mathrm{Cl}_1^{(\boldsymbol{v}_j)}, \mathrm{Cl}_2^{(\boldsymbol{w}_t)}) : \begin{array}{c} \exists\, \boldsymbol{x}_1 \in \mathrm{Cl}_1^{(\boldsymbol{v}_j)}(\boldsymbol{a}_3), \boldsymbol{x}_2 \in \mathrm{Cl}_2^{(\boldsymbol{w}_t)}(\boldsymbol{a}_3) \text{ s.t. } f(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{a}_3) = \boldsymbol{b}, \\ \text{for all } 1 \le j \le V_1(\boldsymbol{a}_3), 1 \le t \le V_2(\boldsymbol{a}_3) \end{array} \right\}.$$

The above discussion then implies that $M(\boldsymbol{a}_3, \boldsymbol{b}) \ge |\mathcal{V}_{12}(\boldsymbol{a}_3, \boldsymbol{b})|$. By the definition, if $(\mathrm{Cl}_1^{(\boldsymbol{v})}, \mathrm{Cl}_2^{(\boldsymbol{w})}) \in \mathcal{V}_{12}(\boldsymbol{a}_3, \boldsymbol{b})$ then for every $i \in \{1, 2, \ldots, k\}$, the pair of scalar equivalence classes $(\mathrm{Cl}_1^{(v^{(i)})}, \mathrm{Cl}_2^{(w^{(i)})}) \in \mathcal{V}_{12}(a_3^{(i)}, b^{(i)})$.

Illustration 3 *Consider block size $k = 1$ and a realization $b = 0$ of the demand function of Table 3.1. In the previous illustration we evaluated that $\mathrm{Cl}_1^{(1)}(1) = \{0, 1\}$, $\mathrm{Cl}_1^{(2)}(1) = \{2\}$ and $\mathrm{Cl}_2^{(1)}(1) = \{1, 2\}$, $\mathrm{Cl}_2^{(2)}(1) = \{0\}$. Then we can evaluate that the pair index set $\mathcal{V}_{12}(1, 0) = \{(\mathrm{Cl}_1^{(1)}, \mathrm{Cl}_2^{(1)}), (\mathrm{Cl}_1^{(1)}, \mathrm{Cl}_2^{(2)}), (\mathrm{Cl}_1^{(2)}, \mathrm{Cl}_2^{(1)})\}$. Note that the pair $(\mathrm{Cl}_1^{(2)}, \mathrm{Cl}_2^{(2)}) \notin \mathcal{V}_{12}(1, 0)$ as the elements of that pair of equivalence classes do not result in the demand function value of 0, i.e.,*

$$\mathrm{Cl}_1^{(2)}(1) = \{2\},\ \mathrm{Cl}_2^{(2)}(1) = \{0\} \quad \textit{but for } \ x_1 = 2, x_2 = 0, x_3 = 1,\ f(x_1, x_2, x_3) = 1 \neq 0.$$

Thus in this case we have that $|\mathcal{V}_{12}(1, 0)| = 3$. The other pair index sets are given in Table 3.2.

We now explicitly derive a p.m.f. whose entropy is a lower bound to the conditional entropy $H(\boldsymbol{Z}_1, \boldsymbol{Z}_2 | f(X_1^k, X_2^k, X_3^k) = \boldsymbol{b}, X_3^k = \boldsymbol{a}_3)$. Let $A_{123}(\boldsymbol{b}) \subseteq \mathcal{A}^k \times \mathcal{A}^k \times \mathcal{A}^k$ contain all message tuples that are present in the pre-image of the demand function value of $\boldsymbol{b}$. We also let $A_{123}(\boldsymbol{b}, \boldsymbol{a}_3) \subseteq A_{123}(\boldsymbol{b})$ denote the subset of message tuples for which the realization of X_3^k is $\boldsymbol{a}_3$. Suppose that $(\mathrm{Cl}_1^{(\boldsymbol{v})}, \mathrm{Cl}_2^{(\boldsymbol{w})}) \in \mathcal{V}_{12}(\boldsymbol{a}_3, \boldsymbol{b})$. The number of different message tuples that cause the membership of

Table 3.2: The sets $\mathcal{V}_{12}(a_3, b)$ for different values of the demand function realization b and different values of a_3 in different rows.

a_3	$b = 0$	$b = 1$	$b = 2$
0	$\{(\mathrm{Cl}_1^{(1)}, \mathrm{Cl}_2^{(1)}), (\mathrm{Cl}_1^{(2)}, \mathrm{Cl}_2^{(2)}), (\mathrm{Cl}_1^{(3)}, \mathrm{Cl}_2^{(3)})\}$	$\{(\mathrm{Cl}_1^{(1)}, \mathrm{Cl}_2^{(3)}), (\mathrm{Cl}_1^{(2)}, \mathrm{Cl}_2^{(1)}), (\mathrm{Cl}_1^{(3)}, \mathrm{Cl}_2^{(2)})\}$	$\{(\mathrm{Cl}_1^{(1)}, \mathrm{Cl}_2^{(2)}), (\mathrm{Cl}_1^{(2)}, \mathrm{Cl}_2^{(3)}), (\mathrm{Cl}_1^{(3)}, \mathrm{Cl}_2^{(1)})\}$
1	$\{(\mathrm{Cl}_1^{(1)}, \mathrm{Cl}_2^{(1)}), (\mathrm{Cl}_1^{(1)}, \mathrm{Cl}_2^{(2)}), (\mathrm{Cl}_1^{(2)}, \mathrm{Cl}_2^{(1)})\}$	$\{(\mathrm{Cl}_1^{(2)}, \mathrm{Cl}_2^{(2)})\}$	$\emptyset$
2	$\{(\mathrm{Cl}_1^{(2)}, \mathrm{Cl}_2^{(2)})\}$	$\{(\mathrm{Cl}_1^{(1)}, \mathrm{Cl}_1^{(1)}), (\mathrm{Cl}_1^{(1)}, \mathrm{Cl}_2^{(2)}), (\mathrm{Cl}_1^{(2)}, \mathrm{Cl}_2^{(1)})\}$	$\emptyset$

the equivalence class pair $(\mathrm{Cl}_1^{(\boldsymbol{v})}, \mathrm{Cl}_2^{(\boldsymbol{w})}) \in \mathcal{V}_{12}(\boldsymbol{a}_3, \boldsymbol{b})$ is denoted as follows.

$$h_{\boldsymbol{a}_3}(\boldsymbol{v}, \boldsymbol{w}) \triangleq \left| \{(\boldsymbol{x}_1, \boldsymbol{x}_2) : \boldsymbol{x}_1 \in \mathrm{Cl}_1^{(\boldsymbol{v})}(\boldsymbol{a}_3), \boldsymbol{x}_2 \in \mathrm{Cl}_2^{(\boldsymbol{w})}(\boldsymbol{a}_3), f(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{a}_3) = \boldsymbol{b}\} \right|, \tag{3.6}$$

$$= |\mathrm{Cl}_1^{(\boldsymbol{v})}(\boldsymbol{a}_3)| \cdot |\mathrm{Cl}_2^{(\boldsymbol{w})}(\boldsymbol{a}_3)|, \tag{3.7}$$

$$= \prod_{i=1}^{k} |\mathrm{Cl}_1^{(v^{(i)})}(a_3^{(i)})| \cdot |\mathrm{Cl}_2^{(w^{(i)})}(a_3^{(i)})| = \prod_{i=1}^{k} h_{a_3^{(i)}}(v^{(i)}, w^{(i)}). \tag{3.8}$$

Equality (3.7) is true above as by Definition 9 every element of an equivalence class under $\overset{\boldsymbol{a}_3}{\equiv}|_1$ results in the same demand function value (while the other message $\boldsymbol{x}_2$ is held constant), and since $(\mathrm{Cl}_1^{(\boldsymbol{v})}, \mathrm{Cl}_2^{(\boldsymbol{w})}) \in \mathcal{V}_{12}(\boldsymbol{a}_3, \boldsymbol{b})$, there is at least one $\boldsymbol{x}_1 \in \mathrm{Cl}_1^{(\boldsymbol{v})}(\boldsymbol{a}_3)$ and one $\boldsymbol{x}_2 \in \mathrm{Cl}_2^{(\boldsymbol{w})}(\boldsymbol{a}_3)$ such that $f(\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{a}_3) = \boldsymbol{b}$. Hence every other pair of elements in $\mathrm{Cl}_1^{(\boldsymbol{v})}(\boldsymbol{a}_3) \times \mathrm{Cl}_2^{(\boldsymbol{w})}(\boldsymbol{a}_3)$ would also result in the same demand function value with $X_3^k = \boldsymbol{a}_3$.

Illustration 4 *For block size $k = 1$ and demand function realization $b = 0$, we can check that $A_3(0) = \{0, 1, 2\}$. Following the indexing of the equivalence partitions and Table 3.2 we have that $h_1(1, 1) = |\mathrm{Cl}_1^{(1)}(1)| \cdot |\mathrm{Cl}_2^{(1)}(1)| = 4$ as $\mathrm{Cl}_1^{(1)}(1) = \{0, 1\}$ and $\mathrm{Cl}_2^{(1)}(1) = \{1, 2\}$. One can similarly check that $h_1(1, 2) = h_1(2, 1) = 2$ and for other values of a_3, that $h_0(1, 1) = h_0(2, 2) = h_0(3, 3) = 1$ and $h_2(2, 2) = 1$.*

For block size $k = 3$ and $\boldsymbol{b} = (1, 2, 1)$ we can check that $\boldsymbol{a}_3 = (0, 0, 1) \in A_3(\boldsymbol{b})$. Then the equivalence class pairs under $\boldsymbol{a}_3$ that result in this $\boldsymbol{b}$ can be obtained in the following manner. If $(\mathrm{Cl}_1^{(\boldsymbol{v})}, \mathrm{Cl}_2^{(\boldsymbol{w})}) \in \mathcal{V}_{12}(\boldsymbol{a}_3, \boldsymbol{b})$, then for each component, using Table 3.2 we have that

- $(\mathrm{Cl}_1^{(v^{(1)})}, \mathrm{Cl}_2^{(w^{(1)})}) \in \mathcal{V}_{12}(a_3^{(1)}, b^{(1)}) = \mathcal{V}_{12}(0, 1) = \{(\mathrm{Cl}_1^{(1)}, \mathrm{Cl}_2^{(3)}), (\mathrm{Cl}_1^{(2)}, \mathrm{Cl}_2^{(1)}), (\mathrm{Cl}_1^{(3)}, \mathrm{Cl}_2^{(2)})\}$,
- $(\mathrm{Cl}_1^{(v^{(2)})}, \mathrm{Cl}_2^{(w^{(2)})}) \in \mathcal{V}_{12}(a_3^{(2)}, b^{(2)}) = \mathcal{V}_{12}(0, 2) = \{(\mathrm{Cl}_1^{(1)}, \mathrm{Cl}_2^{(2)}), (\mathrm{Cl}_1^{(2)}, \mathrm{Cl}_2^{(3)}), (\mathrm{Cl}_1^{(3)}, \mathrm{Cl}_2^{(1)})\}$,
- $(\mathrm{Cl}_1^{(v^{(3)})}, \mathrm{Cl}_2^{(w^{(3)})}) \in \mathcal{V}_{12}(a_3^{(3)}, b^{(3)}) = \mathcal{V}_{12}(1, 1) = \{(\mathrm{Cl}_1^{(2)}, \mathrm{Cl}_2^{(2)})\}$.

As $(\mathrm{Cl}_1^{(\boldsymbol{v})}, \mathrm{Cl}_2^{(\boldsymbol{w})}) = (\bigtimes_{i=1}^{3} \mathrm{Cl}_1^{(v^{(i)})}, \bigtimes_{i=1}^{3} \mathrm{Cl}_2^{(w^{(i)})})$, we get that $|\mathcal{V}_{12}(\boldsymbol{a}_3, \boldsymbol{b})| = 9$. One of these nine equivalence class pairs is the pair $(\mathrm{Cl}_1^{((1,1,2))}, \mathrm{Cl}_2^{((3,2,2))})$. Then using equation (3.8) we have that

$$h_{(0,0,1)}((1, 1, 2), (3, 2, 2)) = h_0(1, 3) \cdot h_0(1, 2) \cdot h_1(2, 2) = 1 \cdot 1 \cdot 1 = 1.$$

The proof of the following lemma is similar in spirit to that of Lemma 7.

Lemma 8 *For any $\boldsymbol{a}_3 \in A_3(\boldsymbol{b})$, define the vector*

$$\boldsymbol{h}_{\boldsymbol{b},\boldsymbol{a}_3} \triangleq \left[h_{\boldsymbol{a}_3}\left((\boldsymbol{v},\boldsymbol{w})_1\right) \quad h_{\boldsymbol{a}_3}\left((\boldsymbol{v},\boldsymbol{w})_2\right) \quad \cdots \quad h_{\boldsymbol{a}_3}\left((\boldsymbol{v},\boldsymbol{w})_{|\mathcal{V}_{12}(\boldsymbol{a}_3,\boldsymbol{b})|}\right) \quad \boldsymbol{0}_{M(\boldsymbol{a}_3,\boldsymbol{b})-|\mathcal{V}_{12}(\boldsymbol{a}_3,\boldsymbol{b})|} \right]^\top,$$

where $\boldsymbol{0}_t$ indicates a vector of zeros of length t and subscript j in $(\boldsymbol{v},\boldsymbol{w})_j$ indexes all the equivalence class pairs such that

$$h_{\boldsymbol{a}_3}\left((\boldsymbol{v},\boldsymbol{w})_1\right) \geq h_{\boldsymbol{a}_3}\left((\boldsymbol{v},\boldsymbol{w})_2\right) \geq \cdots \geq h_{\boldsymbol{a}_3}\left((\boldsymbol{v},\boldsymbol{w})_{|\mathcal{V}_{12}(\boldsymbol{a}_3,\boldsymbol{b})|}\right).$$

Then all conditional probability mass functions $\boldsymbol{p} \in \mathbb{R}_{\geq 0}^{M(\boldsymbol{a}_3,\boldsymbol{b})}$ on $M(\boldsymbol{a}_3,\boldsymbol{b})$ valid $(\boldsymbol{z}_1,\boldsymbol{z}_2)$-labels given the value $\boldsymbol{b}$ of the demand function and the realization $\boldsymbol{a}_3$ of X_3^k satisfy $\boldsymbol{p} \prec \boldsymbol{h}_{\boldsymbol{b},\boldsymbol{a}_3}/|A_{123}(\boldsymbol{b},\boldsymbol{a}_3)|$.

It will be useful for analysis to define the following index sets of a demand function realization $\boldsymbol{b} \in \mathcal{B}^k$ and the message realization $\boldsymbol{a}_3 \in A_3(\boldsymbol{b})$.

Definition 10 *For every $p \in \mathcal{B}$, let $\mathcal{I}_p(\boldsymbol{b}) \subseteq \{1,2,\ldots,k\}$ be the index set of components of the demand function realization $\boldsymbol{b}$ that are equal to p, i.e.,*

$$\mathcal{I}_p(\boldsymbol{b}) = \{i : b^{(i)} = p, i \in \{1,2,\ldots,k\}\}.$$

We can similarly define the index set $\mathcal{J}_q(\boldsymbol{x}_3)$ for every $q \in \mathcal{A}$.

Illustration 5 *We evaluate $H(\boldsymbol{h}_{\boldsymbol{b},\boldsymbol{a}_3}/|A_{123}(\boldsymbol{b},\boldsymbol{a}_3)|)$ for a general block length k of the example demand function. To specify $\boldsymbol{h}_{\boldsymbol{b},\boldsymbol{a}_3}$, we need to find $|\mathcal{V}_{12}(\boldsymbol{a}_3,\boldsymbol{b})|$ and $h_{\boldsymbol{a}_3}((\boldsymbol{v},\boldsymbol{w})_j)$ for all $j \in \{1,2,\ldots,|\mathcal{V}_{12}(\boldsymbol{a}_3,\boldsymbol{b})|\}$. Suppose the realization $\boldsymbol{b} \in \{0,1,2\}^k$ has m_1 1's, m_2 2's and $k-m_1-m_2$ 0's, and the realization of X_3^k is some $\boldsymbol{a}_3 \in A_3(\boldsymbol{b})$. Let $t_{p,q}$ for any $p \in \mathcal{B}, q \in \mathcal{A}$ be defined as $t_{p,q} \triangleq |\mathcal{I}_p(\boldsymbol{b}) \cap \mathcal{J}_q(\boldsymbol{a}_3)|$. Note that $t_{2,1} = t_{2,2} = 0$ for any choice of $\boldsymbol{b}$ and $\boldsymbol{a}_3$.*

$\boxed{|\mathcal{V}_{12}(\boldsymbol{a}_3,\boldsymbol{b})|}$ *We find the number of non-zero components of $\boldsymbol{h}_{\boldsymbol{b},\boldsymbol{a}_3}$, i.e. $|\mathcal{V}_{12}(\boldsymbol{b},\boldsymbol{a}_3)|$, as follows. For every $(\mathrm{Cl}_1^{(\boldsymbol{v})},\mathrm{Cl}_2^{(\boldsymbol{w})}) \in \mathcal{V}_{12}(\boldsymbol{a}_3,\boldsymbol{b})$, we have from Table 3.2 that for the index i,*

- *if $i \in (\mathcal{I}_0(\boldsymbol{b}) \cap \mathcal{J}_2(\boldsymbol{a}_3)) \cup (\mathcal{I}_1(\boldsymbol{b}) \cap \mathcal{J}_1(\boldsymbol{a}_3))$, there is only one possible choice for the scalar equivalence class pair $(\mathrm{Cl}_1^{(v^{(i)})},\mathrm{Cl}_2^{(w^{(i)})})$,*

- *else if i is not in the previous index set, then there are three possible choices for the scalar equivalence class pair* $(\mathrm{Cl}_1^{(v^{(i)})}, \mathrm{Cl}_2^{(w^{(i)})})$.

Thus the total number of equivalence class pairs for for the choice of $\boldsymbol{b}$ *and* $\boldsymbol{a}_3$ *is* $|\mathcal{V}_{12}(\boldsymbol{a}_3, \boldsymbol{b})| = 3^{k-t_{0,2}-t_{1,1}}$. *Next we evaluate the components of the vector* $\boldsymbol{h}_{\boldsymbol{b},\boldsymbol{a}_3}$.

$\boxed{h_{\boldsymbol{a}_3}((\boldsymbol{v},\boldsymbol{w})_1)}$ *Based on the subscript indexing of the pair* $(\boldsymbol{v},\boldsymbol{w})$ *and equation (3.8), the value of* $h_{\boldsymbol{a}_3}((\boldsymbol{v},\boldsymbol{w})_1)$ *is obtained by counting the number of message tuples in the equivalence class pair that has the largest scalar equivalence class pair for each component. As evaluated in the previous illustration, the largest equivalence class pair for every* $i \in \mathcal{I}_0(\boldsymbol{b}) \cap \mathcal{J}_1(\boldsymbol{a}_3)$ *has* $h_1(1,1) = 4$ *message tuples, and for every* $i \in \mathcal{I}_1(\boldsymbol{b}) \cap \mathcal{J}_2(\boldsymbol{a}_3)$ *the largest equivalence class again has* $h_2(1,1) = 4$ *message tuples. For all other values of* i, *the largest equivalence class has a single input tuple. Hence we have that* $h_{\boldsymbol{a}_3}((\boldsymbol{v},\boldsymbol{w})_1) = 4^{t_{0,1}+t_{1,2}}$. *Note that there are three different choices for the largest scalar equivalence class when* i *belongs to one of the following sets.*

- $i \in \mathcal{I}_0(\boldsymbol{b}) \cap \mathcal{J}_0(\boldsymbol{a}_3)$*: there are* $t_{0,0} = k - m_1 - m_2 - t_{0,1} - t_{0,2}$ *such components,*
- $i \in \mathcal{I}_1(\boldsymbol{b}) \cap \mathcal{J}_0(\boldsymbol{a}_3)$*: there are* $t_{1,0} = m_1 - t_{1,1} - t_{1,2}$ *such components, and*
- $i \in \mathcal{I}_2(\boldsymbol{b}) \cap \mathcal{J}_0(\boldsymbol{a}_3)$*: there are* $t_{2,0} = m_2$ *such components.*

Thus there are $3^{k-m_1-m_2-t_{0,1}-t_{0,2}} \cdot 3^{m_1-t_{1,1}-t_{1,2}} \cdot 3^{m_2} = 3^{k-t_{0,1}-t_{0,2}-t_{1,1}-t_{1,2}}$ *components of* $h_{\boldsymbol{b},\boldsymbol{a}_3}$ *which have the same value. Let* $k' \triangleq k - t_{0,1} - t_{0,2} - t_{1,1} - t_{1,2}$. *Hence we have that*

$$h_{\boldsymbol{a}_3}((\boldsymbol{v},\boldsymbol{w})_1) = h_{\boldsymbol{a}_3}((\boldsymbol{v},\boldsymbol{w})_2) = \cdots = h_{\boldsymbol{a}_3}((\boldsymbol{v},\boldsymbol{w})_{3^{k'}}) = 4^{t_{0,1}+t_{1,2}}.$$

$\boxed{h_{\boldsymbol{a}_3}((\boldsymbol{v},\boldsymbol{w})_c) \text{ for } c > 3^{k'}}$ *Next we consider the case when not every component of a message tuple* $(\boldsymbol{x}_1, \boldsymbol{x}_2) \in \mathrm{Cl}_1^{(\boldsymbol{v})} \times \mathrm{Cl}_2^{(\boldsymbol{w})}$ *is present in the largest scalar equivalence class pair. Suppose that* $(\mathrm{Cl}_1^{(\boldsymbol{v})}, \mathrm{Cl}_2^{(\boldsymbol{w})})$ *is such that*

- *at* u_0 *indices from* $\mathcal{I}_0(\boldsymbol{b}) \cup \mathcal{J}_1(\boldsymbol{a}_3)$, *the equivalence class pair is either* $(\mathrm{Cl}_1^{(1)}, \mathrm{Cl}_2^{(2)})$ *or* $(\mathrm{Cl}_1^{(2)}, \mathrm{Cl}_2^{(1)})$, *and*
- *at* u_1 *indices from* $\mathcal{I}_1(\boldsymbol{b}) \cup \mathcal{J}_2(\boldsymbol{a}_3)$, *the equivalence class pair is either* $(\mathrm{Cl}_1^{(1)}, \mathrm{Cl}_2^{(2)})$ *or* $(\mathrm{Cl}_1^{(2)}, \mathrm{Cl}_2^{(1)})$.

Let $c > 3^{k'}$ be the index in $\boldsymbol{h}_{\boldsymbol{b},\boldsymbol{a}_3}$ corresponding to this equivalence class pair. We have that $h_1(1,2) = h_1(2,1) = 2$ and $h_2(1,2) = h_2(2,1) = 2$. Then we get that

$$h^{(c)}_{\boldsymbol{b},\boldsymbol{a}_3} = h_{\boldsymbol{a}_3}((\boldsymbol{v},\boldsymbol{w})_c) = 4^{t_{0,1}-u_0+t_{1,2}-u_1} \cdot 2^{u_0+u_1} = \frac{4^{t_{0,1}+t_{1,2}}}{2^{u_0+u_1}}.$$

Thus the number of components of $\boldsymbol{h}_{\boldsymbol{b},\boldsymbol{a}_3}$ that have the same value as $h^{(c)}_{\boldsymbol{b},\boldsymbol{a}_3}$ are

$$\binom{t_{0,1}}{u_0} 2^{u_0} \binom{t_{1,2}}{u_1} 2^{u_1} 3^{k'}.$$

Thus, the vector $\boldsymbol{h}_{\boldsymbol{b},\boldsymbol{a}_3}$ is as follows, where u_0 and u_1 are indices satisfying $1 \le u_0 \le t_{0,1}$ and $1 \le u_1 \le t_{1,2}$.

$$\boldsymbol{h}_{\boldsymbol{b},\boldsymbol{a}_3} = \Big[\underbrace{4^{t_{0,1}+t_{1,2}} \ \dots \ 4^{t_{0,1}+t_{1,2}}}_{3^{k'}} \ \dots \ \underbrace{\frac{4^{t_{0,1}+t_{1,2}}}{2^{u_0+u_1}} \ \dots \ \frac{4^{t_{0,1}+t_{1,2}}}{2^{u_0+u_1}}}_{\binom{t_{0,1}}{u_0}2^{u_0}\binom{t_{1,2}}{u_1}2^{u_1}3^{k'}} \ \dots \ \underbrace{\frac{4^{t_{0,1}+t_{1,2}}}{2^{t_{0,1}+t_{1,2}}} \ \dots \ \frac{4^{t_{0,1}+t_{1,2}}}{2^{t_{0,1}+t_{1,2}}}}_{2^{t_{0,1}}2^{t_{1,2}}3^{k'}}\Big]^\top. \tag{3.9}$$

Using Table 3.1, the cardinality of the pre-image set $|A_{123}(\boldsymbol{b},\boldsymbol{a}_3)| = 3^{t_{0,0}}8^{t_{0,1}}1^{t_{0,2}}3^{t_{1,0}}1^{t_{1,1}}8^{t_{1,2}}3^{t_{2,0}} = 3^{k'}8^{t_{0,1}+t_{1,2}}$. Using this, we can find the value of the entropy as follows.

$$\begin{aligned}
&H_{|\mathcal{Z}|}(\boldsymbol{h}_{\boldsymbol{b},\boldsymbol{a}_3}/|A_{123}(\boldsymbol{b},\boldsymbol{a}_3)|) \\
&= \sum_{u_0=0}^{t_{0,1}} \sum_{u_1=0}^{t_{1,2}} \binom{t_{0,1}}{u_0} 2^{u_0} \binom{t_{1,2}}{u_1} 2^{u_1} 3^{k'} \frac{4^{t_{0,1}+t_{1,2}}}{2^{u_0+u_1}3^{k'}8^{t_{0,1}+t_{1,2}}} \log_{|\mathcal{Z}|}\left(\frac{2^{u_0+u_1}3^{k'}8^{t_{0,1}+t_{1,2}}}{4^{t_{0,1}+t_{1,2}}}\right) \\
&= \frac{(t_{0,1}+t_{1,2})\log_{|\mathcal{Z}|}2 + k'\log_{|\mathcal{Z}|}3}{2^{t_{0,1}+t_{1,2}}} \sum_{u_0=0}^{t_{0,1}} \sum_{u_1=0}^{t_{1,2}} \binom{t_{0,1}}{u_0}\binom{t_{1,2}}{u_1} + \frac{\log_{|\mathcal{Z}|}2}{2^{t_{0,1}+t_{1,2}}} \sum_{u_0=0}^{t_{0,1}} \sum_{u_1=0}^{t_{1,2}} \binom{t_{0,1}}{u_0} u_0 \binom{t_{1,2}}{u_1} \\
&\quad + \frac{\log_{|\mathcal{Z}|}2}{2^{t_{0,1}+t_{1,2}}} \sum_{u_0=0}^{t_{0,1}} \sum_{u_1=0}^{t_{1,2}} \binom{t_{0,1}}{u_0} u_1 \binom{t_{1,2}}{u_1} \\
&= k'\log_{|\mathcal{Z}|}3 + 1.5(t_{0,1}+t_{1,2})\log_{|\mathcal{Z}|}2 \\
&= k\log_{|\mathcal{Z}|}3 + (1.5\log_{|\mathcal{Z}|}2 - \log_{|\mathcal{Z}|}3)(t_{0,1}+t_{1,2}) - (t_{0,2}+t_{1,1})\log_{|\mathcal{Z}|}3.
\end{aligned} \tag{3.10}$$

3.3.2 Value of α

Having evaluated $H(\boldsymbol{h}_{\boldsymbol{b},\boldsymbol{a}_3}/|A_{123}(\boldsymbol{b},\boldsymbol{a}_3)|)$, we can find the value of α as used in equation (3.2) in the following manner.

$$
\begin{aligned}
&H(\boldsymbol{Z}_1,\boldsymbol{Z}_2|f(X_1^k,X_2^k,X_3^k),X_3^k)\\
&=\sum_{\boldsymbol{b}}\Pr\{f(X_1^k,X_2^k,X_3^k)=\boldsymbol{b}\}\sum_{\boldsymbol{x}_3}\Pr\{X_3^k=\boldsymbol{x}_3|f(X_1^k,X_2^k,X_3^k)=\boldsymbol{b}\}H(\boldsymbol{Z}_1,\boldsymbol{Z}_2|f(X_1^k,X_2^k,X_3^k)=\boldsymbol{b},X_3^k=\boldsymbol{x}_3)\\
&\geq\sum_{\boldsymbol{b}}\Pr\{f(X_1^k,X_2^k,X_3^k)=\boldsymbol{b}\}\sum_{\boldsymbol{x}_3}\Pr\{X_3^k=\boldsymbol{x}_3|f(X_1^k,X_2^k,X_3^k)=\boldsymbol{b}\}H(\boldsymbol{h}_{\boldsymbol{b},\boldsymbol{a}_3}/|A_{123}(\boldsymbol{b},\boldsymbol{a}_3)|)\triangleq\alpha k.
\end{aligned}
\tag{3.11}
$$

The value of $\Pr\{f(X_1^k,X_2^k,X_3^k)=\boldsymbol{b}\}$ can be found from Table 3.1 and the i.i.d. uniform assumption on the message tuples. The value of $\Pr\{X_3^k=\boldsymbol{x}_3|f(X_1^k,X_2^k,X_3^k)=\boldsymbol{b}\}$ can be evaluated by finding the ratio of the cardinalities of two pre-image sets, i.e., $|A_{123}(\boldsymbol{b},\boldsymbol{x}_3)|/|A_{123}(\boldsymbol{b})|$. We carry out this computation for the illustrated demand function below.

Illustration 6 *Consider a realization $\boldsymbol{b}$ with m_1 1's, m_2 2's and $k-m_1-m_2$ 0's. Then we have that $\Pr\{f(X_1^k,X_2^k,X_3^k)=\boldsymbol{b}\}=(4/9)^{m_1}(1/9)^{m_2}(4/9)^{k-m_1-m_2}$. The number of different demand function realizations which have the same number of 1's, 2's and 0's is $\binom{k}{m_1,m_2,k-m_1-m_2}$. Consider a realization $\boldsymbol{a}_3\in A_3(\boldsymbol{b})$ of the message X_3^k. The number of message tuples that result in the demand function value $\boldsymbol{b}$ and have their X_3^k realization as $\boldsymbol{a}_3$ is $|A_{123}(\boldsymbol{b},\boldsymbol{a}_3)|=3^{k'}8^{t_{0,1}+t_{1,2}}$, as evaluated in the previous illustration. The number of message tuples in the pre-image set $A_{123}(\boldsymbol{b})$ can be evaluated using Table 3.1 as $|A_{123}(\boldsymbol{b})|=12^{m_1}3^{m_2}12^{k-m_1-m_2}=3^k4^{k-m_2}$. Because of the uniform i.i.d. assumption, we have that*

$$
\begin{aligned}
\Pr\{X_3^k=\boldsymbol{x}_3|f(X_1^k,X_2^k,X_3^k)=\boldsymbol{b}\}&=\frac{|A_{123}(\boldsymbol{b},\boldsymbol{a}_3)|}{|A_{123}(\boldsymbol{b})|}=\frac{3^{k'}8^{t_{0,1}+t_{1,2}}}{3^k4^{k-m_2}}\\
&=(1/4)^{k-m_1-m_2-t_{0,1}-t_{0,2}}(2/3)^{t_{0,1}}(1/12)^{t_{0,2}}(1/4)^{m_1-t_{1,1}-t_{1,2}}(2/3)^{t_{1,2}}(1/12)^{t_{1,1}}.
\end{aligned}
$$

Using equations (3.11), (3.10) and the probabilities computed above, the value of α is found in Appendix C to be

$$\alpha=\frac{8}{9}\log_{|\mathcal{Z}|}2+\frac{4}{12}\log_{|\mathcal{Z}|}3. \tag{3.12}$$

Using this value of α in equation (3.2), we get that

$$\frac{R_1+R_2}{2}+\epsilon \geq \frac{\alpha+\left(\log_{|\mathcal{Z}|} 9-\frac{8}{9}\log_{|\mathcal{Z}|} 4\right)}{2\log_{|\mathcal{Z}|}|\mathcal{A}|} = \frac{\frac{8}{9}\log_{|\mathcal{A}|} 2+\frac{1}{3}\log_{|\mathcal{A}|} 3+2\log_{|\mathcal{A}|} 3-\frac{16}{9}\log_{|\mathcal{A}|} 2}{2} \approx \frac{1.7725}{2}.$$

3.3.3 Example demand function: Arithmetic sum

Suppose the message alphabet is $\mathcal{A}=\{0,1\}$, such that the messages X_1, X_2, X_3 are independent bits each equally likely to be 0 or 1. The demand function $f(X_1,X_2,X_3)=X_1+X_2+X_3$ is the sum of the messages over the real numbers, such that $\mathcal{B}=\{0,1,2,3\}$. This case of arithmetic sum computation in the variable-length network code framework was considered in [49] and we recover the results there in our general framework. For a given value of the arithmetic sum $\boldsymbol{b}\in\{0,1,2,3\}^k$ the set $A_3(\boldsymbol{b})$ of valid realizations for X_3^k can be described as

$$A_3(\boldsymbol{b})=\{\boldsymbol{a}_3\in\{0,1\}^k : a_3^{(i)}=0 \text{ if } \boldsymbol{b}^{(i)}=0 \text{ and } a_3^{(j)}=1 \text{ if } \boldsymbol{b}^{(j)}=3 \text{ for all } i,j\in\{1,2,\ldots,k\}\}.$$

We consider the equivalence relation $\overset{\boldsymbol{a}_3}{\equiv}|_1$ for the arithmetic sum demand function. Then

$$\boldsymbol{x}_1 \overset{\boldsymbol{a}_3}{\equiv} \boldsymbol{y}_1|_1 \Leftrightarrow \boldsymbol{x}_1=\boldsymbol{y}_1,$$

because if $\boldsymbol{x}_1\neq\boldsymbol{y}_1$ then for all $\boldsymbol{x}_2\in\{0,1\}^k$ we have that $\boldsymbol{x}_1+\boldsymbol{x}_2+\boldsymbol{a}_3\neq\boldsymbol{y}_1+\boldsymbol{x}_2+\boldsymbol{a}_3$. A similar conclusion also holds true for the $\overset{\boldsymbol{a}_3}{\equiv}|_2$ relation. Thus for both $u=1$ and 2, we have that $|\mathrm{Cl}_u^{(\boldsymbol{v})}(\boldsymbol{a}_3)|=1$ for any class index $\boldsymbol{v}$ and $V_u(\boldsymbol{a}_3)=2^k$. We use the alphabet $\mathcal{Z}=\{0,1\}$ for communication. The vector $\boldsymbol{d}_u(\boldsymbol{a}_3)$ defined in Lemma 7 satisfies $\boldsymbol{d}_u(\boldsymbol{a}_3)=\mathbf{1}_{2^k}$ in this case, where $\mathbf{1}_t$ indicates a vector of ones with length t. Using this in equation (3.5), we obtain the value of γ as

$$\gamma=\frac{1}{k}\sum_{\boldsymbol{a}_3\in\mathcal{A}^k}\Pr\{X_3^k=\boldsymbol{a}_3\}H_{|\mathcal{Z}|}\Big(\frac{\boldsymbol{d}_u(\boldsymbol{a}_3)}{|\mathcal{A}|^k}\Big)=\frac{1}{k}\cdot k=1,$$

and thus $R_u+\epsilon\geq 1/\log_{|\mathcal{Z}|}|\mathcal{A}|=1$.

For any value of X_1^k, X_3^k and $f(X_1^k,X_2^k,X_3^k)$ the value of X_2^k is fixed by $X_2^k=f(X_1^k,X_2^k,X_3^k)-X_1^k-X_3^k$. Hence, for every $(\mathrm{Cl}_1^{(\boldsymbol{v})},\mathrm{Cl}_2^{(\boldsymbol{w})})\in\mathcal{V}_{12}(\boldsymbol{a}_3)$ there is exactly one message tuple whose X_1^k and X_2^k belong to the equivalence classes $\mathrm{Cl}_1^{(\boldsymbol{v})}(\boldsymbol{a}_3)$ and $\mathrm{Cl}_2^{(\boldsymbol{w})}(\boldsymbol{a}_3)$ respectively. Thus we have that

$$h_{\boldsymbol{a}_3}(\boldsymbol{v},\boldsymbol{w})=1 \text{ for every } \boldsymbol{a}_3\in A_3(\boldsymbol{b}) \quad \text{and} \quad (\mathrm{Cl}_1^{(\boldsymbol{v})},\mathrm{Cl}_2^{(\boldsymbol{w})})\in\mathcal{V}_{12}(\boldsymbol{a}_3,\boldsymbol{b}).$$

Consider an arithmetic sum realization $\boldsymbol{b}$ with m_0 0's, m_1 1's, m_2 2's and $k-m_0-m_1-m_2$ 3's. Define $t_{p,q} \triangleq |\mathcal{I}_p(\boldsymbol{b}) \cap \mathcal{J}_q(\boldsymbol{a}_3)|$ for every $p \in \{0,1,2,3\}$ and $q \in \{0,1\}$. Then for any choice of $\boldsymbol{b}$ and $\boldsymbol{a}_3 \in A_3(\boldsymbol{b})$, $t_{0,1} = t_{3,0} = 0$. The cardinality of the pre-image set $|A_{123}(\boldsymbol{b},\boldsymbol{a}_3)| = 2^{t_{1,0}+t_{2,1}} = 2^{t_{1,0}+m_2-t_{2,0}}$. The value of the entropy $H(\boldsymbol{h}_{\boldsymbol{b},\boldsymbol{a}_3}/|A_{123}(\boldsymbol{b},\boldsymbol{a}_3)|) = t_{1,0}+m_2-t_{2,0}$. From the function definition, we can check that $|A_{123}(\boldsymbol{b})| = 3^{m_1+m_2}$. Thus we have that

$$\Pr\{X_3^k = \boldsymbol{x}_3 | f(X_1^k, X_2^k, X_3^k) = \boldsymbol{b}\} = \frac{|A_{123}(\boldsymbol{b},\boldsymbol{a}_3)|}{|A_{123}(\boldsymbol{b})|} = \frac{2^{t_{1,0}+m_2-t_{2,0}}}{3^{m_1+m_2}}.$$

Then the value of α can be found as follows.

$$\begin{aligned}
\alpha k &= \sum_{\boldsymbol{b}} \Pr\{f(X_1^k, X_2^k, X_3^k) = \boldsymbol{b}\} \sum_{\boldsymbol{a}_3 \in A_3(\boldsymbol{b})} \Pr\{X_3^k = \boldsymbol{a}_3 | f(X_1^k, X_2^k, X_3^k) = \boldsymbol{b}\} H(\boldsymbol{h}_{\boldsymbol{b},\boldsymbol{a}_3}/|A_{123}(\boldsymbol{b},\boldsymbol{a}_3)|) \\
&= \sum_{\boldsymbol{b}} \Pr\{f(X_1^k, X_2^k, X_3^k) = \boldsymbol{b}\} \sum_{t_{1,0}=0}^{m_1} \sum_{t_{2,0}=0}^{m_2} \frac{m_1!(2/3)^{t_{1,0}}(1/3)^{m_1-t_{1,0}}}{t_{1,0}!(m_1-t_{1,0})!} \frac{m_2!(2/3)^{m_2-t_{2,0}}(1/3)^{t_{2,0}}}{t_{2,0}!(m_2-t_{2,0})!} (t_{1,0}+m_2-t_{2,0}) \\
&= \sum_{m_1=0}^{k} \sum_{m_2=0}^{k-m_1} \frac{k! 2^{k-m_1-m_2}}{m_1! m_2! (k-m_1-m_2)!} \left(\frac{1}{8}\right)^{k-m_1-m_2} \left(\frac{3}{8}\right)^{m_1+m_2} \frac{2(m_1+m_2)}{3} \\
&= \frac{2/3}{4^k} \sum_{m_1=0}^{k} \sum_{m_2=0}^{k-m_1} \frac{k!(m_1+m_2)}{m_1! m_2! (k-m_1-m_2)!} \left(\frac{3}{2}\right)^{m_1+m_2} = \frac{2/3}{4^k} 2 \cdot \frac{3}{2} k 4^{k-1} = 0.5k.
\end{aligned}$$

Putting this value of α in equation (3.2), we get that

$$\frac{R_1+R_2}{2} + \epsilon \geq \frac{0.5 + 3 - 0.75\log 3}{2} \approx \frac{2.31128}{2}.$$

We note that the lower bound for the sum rate shown above is tighter than the bound $R_1+R_2 > 2.25$ obtained in [49] for the same problem. The inequalities considered there were similar to those used in arriving at the sum rate lower bound in equation (3.2), however, in [49] they did not include X_3^k in the conditioning while lower bounding $H_{|\mathcal{Z}|}(\boldsymbol{Z}_1, \boldsymbol{Z}_2 | f(X_1^k, X_2^k, X_3^k))$ as done here. Instead, they directly lower bounded $H_{|\mathcal{Z}|}(\boldsymbol{Z}_1, \boldsymbol{Z}_2 | f(X_1^k, X_2^k, X_3^k))$ using a so-called *clumpy* distribution, which is a p.m.f. that majorizes any valid conditional p.m.f. of the pair $(\boldsymbol{Z}_1, \boldsymbol{Z}_2)$ given $f(X_1^k, X_2^k, X_3^k)$. Since only a lower bound to the entropy of the clumpy distribution was obtained in [49], the corresponding lower bound for the sum rate ends up being looser than the value we obtain here. In the following, we show that for $k \to \infty$, conditioning on X_3^k does not cause the bound to be any more looser than

that obtained by directly bounding $H_{|\mathcal{Z}|}(\boldsymbol{Z}_1, \boldsymbol{Z}_2 | f(X_1^k, X_2^k, X_3^k))$ using the entropy of the clumpy distribution.

We first describe a family $\mathcal{P}$ of p.m.f.s that must contain the conditional p.m.f. $\Pr(\boldsymbol{Z}_1, \boldsymbol{Z}_2 | f(X_1^k, X_2^k, X_3^k))$ for any valid labelling scheme.

Lemma 9 *Let $L_{12}(\boldsymbol{b})$ be the number of distinct $(\boldsymbol{z}_1, \boldsymbol{z}_2)$ labels used by a valid labelling scheme when X_1^k, X_2^k, X_3^k are such that $f(X_1^k, X_2^k, X_3^k) = \boldsymbol{b}$. That implies $L_{12}(\boldsymbol{b}) \geq M(\boldsymbol{a}_3, \boldsymbol{b})$ and we define $\bar{\boldsymbol{h}}_{\boldsymbol{b},\boldsymbol{a}_3} \triangleq \left[\boldsymbol{h}_{\boldsymbol{b},\boldsymbol{a}_3}^\top \ \boldsymbol{0}_{L_{12}(\boldsymbol{b})-M(\boldsymbol{a}_3,\boldsymbol{b})}\right]^\top$ for every $\boldsymbol{a}_3 \in A_3(\boldsymbol{b})$. The set of all conditional probability mass functions on $L_{12}(\boldsymbol{b})$ labels given the function value $\boldsymbol{b}$ can be represented by a family $\mathcal{P}$ of vectors over non-negative reals as below.*

$$\mathcal{P} = \left\{ \boldsymbol{p} \in \mathbb{R}_{\geq 0}^{L_{12}(\boldsymbol{b})} : \boldsymbol{p} = \frac{1}{|A_{123}(\boldsymbol{b})|} \sum_{\boldsymbol{a}_3 \in A_3(\boldsymbol{b})} \boldsymbol{g}_{\boldsymbol{a}_3}, \ \text{where each } \boldsymbol{g}_{\boldsymbol{a}_3} \in \mathbb{Z}_{\geq 0}^{L_{12}(\boldsymbol{b})} \text{ and } \boldsymbol{g}_{\boldsymbol{a}_3} \prec \bar{\boldsymbol{h}}_{\boldsymbol{b},\boldsymbol{a}_3} \right\}.$$

Proof: In what follows, by distinct labels we mean distinct $(\boldsymbol{z}_1, \boldsymbol{z}_2)$-pair labels. By definitions of the relevant quantities, there are $L_{12}(\boldsymbol{b})$ distinct labels that are assigned to the message tuples in $A_{123}(\boldsymbol{b})$ and message tuples in $A_{123}(\boldsymbol{b}, \boldsymbol{a}_3)$ for any $\boldsymbol{a}_3 \in A_3(\boldsymbol{b})$ are assigned $M(\boldsymbol{a}_3, \boldsymbol{b})$ distinct labels. However, for two different $\boldsymbol{a}_3, \boldsymbol{a}_3' \in A_3(\boldsymbol{b})$, a label assigned to a message tuple in $A_{123}(\boldsymbol{b}, \boldsymbol{a}_3)$ can potentially be also assigned to a message tuple in $A_{123}(\boldsymbol{b}, \boldsymbol{a}_3')$. Since all messages are equally likely, the conditional probability of observing a particular label then depends on how many message tuples the label is assigned to across all $A_{123}(\boldsymbol{b}, \boldsymbol{a}_3)$-partitions (for all $\boldsymbol{a}_3 \in A_3(\boldsymbol{b})$) of $A_{123}(\boldsymbol{b})$.

Each component of the vector $\boldsymbol{p}$ denotes the conditional probability of a distinct $(\boldsymbol{z}_1, \boldsymbol{z}_2)$-label given the value $\boldsymbol{b}$ of the demand function. We define the non-negative integer vector $\boldsymbol{g}_{\boldsymbol{a}_3}$ as follows. The lth component of $\boldsymbol{g}_{\boldsymbol{a}_3}$, i.e., $g_{\boldsymbol{a}_3}^{(l)}$ records the number of message tuples in $A_{123}(\boldsymbol{b}, \boldsymbol{a}_3)$ that are assigned the $(\boldsymbol{z}_1, \boldsymbol{z}_2)$ label corresponding to the lth component of $\boldsymbol{p}$. Thus each component of the sum $\sum_{\boldsymbol{a}_3 \in A_3(\boldsymbol{b})} \boldsymbol{g}_{\boldsymbol{a}_3}$ describes the number of message tuples in the pre-image set $A_{123}(\boldsymbol{b})$ that are assigned the label corresponding to that component. The conditional p.m.f. of $(\boldsymbol{Z}_1, \boldsymbol{Z}_2)$ given the function value $\boldsymbol{b}$ is obtained by dividing the sum by the cardinality of the pre-image set, i.e., $|A_{123}(\boldsymbol{b})|$.

We now show that for any valid labelling scheme, the vector $\boldsymbol{g}_{\boldsymbol{a}_3} \prec \bar{\boldsymbol{h}}_{\boldsymbol{b},\boldsymbol{a}_3}$ for all $\boldsymbol{a}_3 \in A_3(\boldsymbol{b})$. We can verify that

$$\sum_{j=1}^{L_{12}(\boldsymbol{b})} g_{\boldsymbol{a}_3}^{(j)} = |A_{123}(\boldsymbol{b}, \boldsymbol{a}_3)| = \sum_{l=1}^{M(\boldsymbol{a}_3,\boldsymbol{b})} h_{\boldsymbol{b},\boldsymbol{a}_3}^{(l)} = \sum_{j=1}^{L_{12}(\boldsymbol{b})} \bar{h}_{\boldsymbol{b},\boldsymbol{a}_3}^{(j)},$$

where the first equality is true because each message tuple in $A_{123}(\boldsymbol{b}, \boldsymbol{a}_3)$ is assigned exactly one of the $L_{12}(\boldsymbol{b})$ labels and the other equalities are because of the definitions of the quantities. Now suppose that there is a valid labelling scheme, in which for a $\boldsymbol{a}_3 \in A_3(\boldsymbol{b})$ the vector $\boldsymbol{g} \not\prec \bar{\boldsymbol{h}}_{\boldsymbol{b},\boldsymbol{a}_3}$, i.e., there is an integer $t < L_{12}(\boldsymbol{b})$ such that

$$\sum_{j=1}^{t} g_{\boldsymbol{a}_3}^{[j]} > \sum_{j=1}^{t} \bar{h}_{\boldsymbol{b},\boldsymbol{a}_3}^{[j]}.$$

The LHS counts the maximum number of message tuples in $A_{123}(\boldsymbol{b}, \boldsymbol{a}_3)$ that are assigned t distinct labels among them. The RHS above counts the number of message tuples that are present in the t largest equivalence class pairs in $\mathcal{V}_{12}(\boldsymbol{a}_3, \boldsymbol{b})$. Thus if $\boldsymbol{g}_{\boldsymbol{a}_3} \not\prec \bar{\boldsymbol{h}}_{\boldsymbol{b},\boldsymbol{a}_3}$ then there are message tuples from at least $t+1$ different equivalence class pairs that are assigned t distinct labels. This contradicts Lemma 6. ■

We use Schur-concavity of the entropy function to find the entropy-minimizing clumpy p.m.f. in the family $\mathcal{P}$ next.

Lemma 10 *For $\bar{\boldsymbol{h}}_{\boldsymbol{b},\boldsymbol{a}_3}$ and $\mathcal{P}$ defined in Lemma 9, consider the p.m.f. $\boldsymbol{p}_\star \triangleq \sum_{\boldsymbol{a}_3 \in A_3(\boldsymbol{b})} \bar{\boldsymbol{h}}_{\boldsymbol{b},\boldsymbol{a}_3} / |A_{123}(\boldsymbol{b})|$ with parameters $M^\star(\boldsymbol{a}_3, \boldsymbol{b}) = \mathcal{V}_{12}(\boldsymbol{a}_3, \boldsymbol{b})$ for every $\boldsymbol{a}_3 \in A_3(\boldsymbol{b})$ and $L_{12}^\star(\boldsymbol{b}) = \max_{\boldsymbol{a}_3 \in A_3(\boldsymbol{b})} M^\star(\boldsymbol{a}_3, \boldsymbol{b})$. Then for all $\boldsymbol{p} \in \mathcal{P}$, $H(\boldsymbol{p}) \geq H(\boldsymbol{p}_\star)$. We call $\boldsymbol{p}_\star$ the clumpy distribution.*

Proof: We note that $\boldsymbol{p}_\star \in \mathcal{P}$. The fact that $\boldsymbol{p}_\star \succ \boldsymbol{p}$ for all $\boldsymbol{p} \in \mathcal{P}$ can be seen to be true by [48, Prop. 6.A.1]. We give the proof here for completeness. From Lemma 9 any $\boldsymbol{p} \in \mathcal{P}$ can be expressed as $\sum_{\boldsymbol{a}_3 \in A_3(\boldsymbol{b})} \frac{\boldsymbol{g}_{\boldsymbol{a}_3}}{|A_{123}(\boldsymbol{b})|}$ with $\boldsymbol{g}_{\boldsymbol{a}_3} \prec \bar{\boldsymbol{h}}_{\boldsymbol{b},\boldsymbol{a}_3}$. Let π be the permutation that arranges all components of $\boldsymbol{p} \in \mathcal{P}$ in non-increasing order with increasing index. Let $\boldsymbol{p}_\downarrow \triangleq \pi(\boldsymbol{p})$. Then

$$\boldsymbol{p}_\downarrow = \pi(\boldsymbol{p}) = \pi\left(\frac{\sum_{\boldsymbol{a}_3 \in A_3(\boldsymbol{b})} \boldsymbol{g}_{\boldsymbol{a}_3}}{|A_{123}(\boldsymbol{b})|}\right) = \frac{\sum_{\boldsymbol{a}_3 \in A_3(\boldsymbol{b})} \pi(\boldsymbol{g}_{\boldsymbol{a}_3})}{|A_{123}(\boldsymbol{b})|},$$

so $\boldsymbol{p}_\downarrow \in \mathcal{P}$ as $\boldsymbol{g}_{\boldsymbol{a}_3} \prec \bar{\boldsymbol{h}}_{\boldsymbol{b},\boldsymbol{a}_3}$ implies that $\pi(\boldsymbol{g}_{\boldsymbol{a}_3}) \prec \bar{\boldsymbol{h}}_{\boldsymbol{b},\boldsymbol{a}_3}$ for any permutation π. Then $\boldsymbol{p}_\star \succ \boldsymbol{p}_\downarrow$, as for any $l \in L_{12}(\boldsymbol{b})$,

$$\sum_{i=1}^{l}\left(p_\star^{[i]} - p_\downarrow^{[i]}\right) = \sum_{i=1}^{l}\left(p_\star^{(i)} - p_\downarrow^{(i)}\right) = \frac{1}{|A_{123}(\boldsymbol{b})|}\sum_{i=1}^{l}\left(\sum_{\boldsymbol{a}_3\in A_3(\boldsymbol{b})}\bar{h}_{\boldsymbol{b},\boldsymbol{a}_3}^{(i)} - \sum_{\boldsymbol{a}_3\in A_3(\boldsymbol{b})} g_{\boldsymbol{a}_3}^{(\pi^{-1}(i))}\right),$$

$$= \frac{1}{|A_{123}(\boldsymbol{b})|}\sum_{i=1}^{l}\left(\sum_{\boldsymbol{a}_3\in A_3(\boldsymbol{b})}\left\{\bar{h}_{\boldsymbol{b},\boldsymbol{a}_3}^{(i)} - g_{\boldsymbol{a}_3}^{(\pi^{-1}(i))}\right\}\right) = \frac{1}{|A_{123}(\boldsymbol{b})|}\sum_{\boldsymbol{a}_3\in A_3(\boldsymbol{b})}\left(\sum_{i=1}^{l}\left\{h_{\boldsymbol{b},\boldsymbol{a}_3}^{(i)} - g_{\boldsymbol{a}_3}^{(\pi^{-1}(i))}\right\}\right) \geq 0,$$

where the inequality above is again due to the fact that $\boldsymbol{g}_{\boldsymbol{a}_3} \prec \bar{\boldsymbol{h}}_{\boldsymbol{b},\boldsymbol{a}_3}$ implies that $\pi(\boldsymbol{g}_{\boldsymbol{a}_3}) \prec \bar{\boldsymbol{h}}_{\boldsymbol{b},\boldsymbol{a}_3}$ for any permutation π. Since the majorization relation is a preordering and $\boldsymbol{p}_\star \succ \boldsymbol{p}_\downarrow \succ \boldsymbol{p}$, we get that $\boldsymbol{p}_\star \succ \boldsymbol{p}$ for any $\boldsymbol{p} \in \mathcal{P}$. As the entropy function is strictly Schur-concave, $H(\boldsymbol{p}) \geq H(\boldsymbol{p}_\star)$, with equality if and only if $\boldsymbol{p}$ is a permutation of $\boldsymbol{p}_\star$.

The number of distinct labels used for message tuples in $A_{123}(\boldsymbol{b},\boldsymbol{a}_3)$ is equal to the number of positive components of $\boldsymbol{h}_{\boldsymbol{b},\boldsymbol{a}_3}$. Thus $M^\star(\boldsymbol{a}_3,\boldsymbol{b}) = |\mathcal{V}_{12}(\boldsymbol{a}_3,\boldsymbol{b})|$ for all $\boldsymbol{a}_3 \in A_3(\boldsymbol{b})$ and the total number of non-zero entries of $\boldsymbol{p}_\star$, which is the same as the number of distinct labels used for all the message tuples in $A_{123}(\boldsymbol{b})$ is $L_{12}^\star(\boldsymbol{b}) = \max_{\boldsymbol{a}_3\in A_3(\boldsymbol{b})} M^\star(\boldsymbol{a}_3,\boldsymbol{b})$. ■

We use $(\boldsymbol{C}_1,\boldsymbol{C}_2)$ to denote the codewords whose conditional p.m.f. given the demand function value $\boldsymbol{b}$ is the clumpy distribution, i.e., $\Pr\{\boldsymbol{C}_1,\boldsymbol{C}_2|f(X_1^k,X_2^k,X_3^k)=\boldsymbol{b}\} = \boldsymbol{p}_\star$. From the definition of $\boldsymbol{p}_\star$ and $\mathcal{P}$, we have that $H(\boldsymbol{C}_1,\boldsymbol{C}_2|f(X_1^k,X_2^k,X_3^k)=\boldsymbol{b},X_3^k=\boldsymbol{a}_3) = H(\bar{\boldsymbol{h}}_{\boldsymbol{b},\boldsymbol{a}_3}/|A_{123}(\boldsymbol{b},\boldsymbol{a}_3)|)$. Using the definition of α in Eq. (3.11) and noting that $H(\bar{\boldsymbol{h}}_{\boldsymbol{b},\boldsymbol{a}_3}/|A_{123}(\boldsymbol{b})|) = H(\boldsymbol{h}_{\boldsymbol{b},\boldsymbol{a}_3}/|A_{123}(\boldsymbol{b})|)$, we get that $H(\boldsymbol{C}_1,\boldsymbol{C}_2|f(X_1^k,X_2^k,X_3^k),X_3^k) = \alpha k$. Then we have the following.

$$\begin{aligned}
(H_{|\mathcal{Z}|}(\boldsymbol{Z}_1) + H_{|\mathcal{Z}|}(\boldsymbol{Z}_2))/k &\geq H_{|\mathcal{Z}|}(f(X_1^k,X_2^k,X_3^k))/k + H_{|\mathcal{Z}|}(\boldsymbol{Z}_1,\boldsymbol{Z}_2|f(X_1^k,X_2^k,X_3^k))/k \\
&\geq H_{|\mathcal{Z}|}(f(X_1^k,X_2^k,X_3^k))/k + H_{|\mathcal{Z}|}(\boldsymbol{C}_1,\boldsymbol{C}_2|f(X_1^k,X_2^k,X_3^k))/k \\
&= H_{|\mathcal{Z}|}(f(X_1^k,X_2^k,X_3^k))/k + H_{|\mathcal{Z}|}(\boldsymbol{C}_1,\boldsymbol{C}_2|f(X_1^k,X_2^k,X_3^k),X_3^k)/k \\
&\quad + I_{|\mathcal{Z}|}(\boldsymbol{C}_1,\boldsymbol{C}_2;X_3^k|f(X_1^k,X_2^k,X_3^k))/k \\
&\to H_{|\mathcal{Z}|}(f(X_1^k,X_2^k,X_3^k))/k + \alpha + I_{|\mathcal{Z}|}(\boldsymbol{C}_1,\boldsymbol{C}_2;X_3^k|f(X_1^k,X_2^k,X_3^k))/k \\
&\quad \text{as } k\to\infty.
\end{aligned}$$

Next, we show that when $f(X_1^k,X_2^k,X_3^k)$ is the arithmetic sum function,

$I_{|\mathcal{Z}|}(\boldsymbol{C}_1,\boldsymbol{C}_2;X_3^k|f(X_1^k,X_2^k,X_3^k))/k \to 0$ as $k\to\infty$.

Definition 11 *For $p \in \{0,1,2,3\}$ and $q \in \{0,1\}$, define the random variable*

$$T_{p,q}(X_3^k, f(X_1^k, X_2^k, X_3^k)) = |\mathcal{I}_p(f(X_1^k, X_2^k, X_3^k)) \cap \mathcal{J}_q(X_3^k)|.$$

For arithmetic sum, we have that $T_{0,1}(X_3^k, f(X_1^k, X_2^k, X_3^k)) = T_{3,0}(X_3^k, f(X_1^k, X_2^k, X_3^k)) = 0$.

Conditioned on a particular value of the arithmetic sum, say $\boldsymbol{b}$, both $T_{1,0}(\cdot)$ and $T_{2,1}(\cdot)$ follow the Binomial distribution. Suppose $\boldsymbol{b}$ has m_1 1's and m_2 2's. Then we have that

$$\begin{aligned}
&\Pr\{T_{1,0}(X_3^k, f(X_1^k, X_2^k, X_3^k)) = t_{1,0} | f(X_1^k, X_2^k, X_3^k) = \boldsymbol{b}\} \\
&= \binom{m_1}{t_{1,0}} \Pr(X_3 = 0 | f(X_1, X_2, X_3) = 1)^{t_{1,0}} \Pr(X_3 = 1 | f(X_1, X_2, X_3) = 1)^{m_1 - t_{1,0}} \\
&= \binom{m_1}{t_{1,0}} \left(\frac{2}{3}\right)^{t_{1,0}} \left(\frac{1}{3}\right)^{m_1 - t_{1,0}}, \\
&\Pr\{T_{2,1}(X_3^k, f(X_1^k, X_2^k, X_3^k)) = t_{2,1} | f(X_1^k, X_2^k, X_3^k) = \boldsymbol{b}\} \\
&= \binom{m_2}{t_{2,1}} \Pr(X_3 = 1 | f(X_1, X_2, X_3) = 2)^{t_{2,1}} \Pr(X_3 = 0 | f(X_1, X_2, X_3) = 2)^{m_2 - t_{2,1}} \\
&= \binom{m_2}{t_{2,1}} \left(\frac{2}{3}\right)^{t_{2,1}} \left(\frac{1}{3}\right)^{m_2 - t_{2,1}}.
\end{aligned}$$

For either $p = 1$ or 2, consider the random source $\{X_3^{(j)} \in \mathcal{A} : j \in \mathcal{I}_p(\boldsymbol{b}), f(X_1^k, X_2^k, X_3^k) = \boldsymbol{b}\}$, and define its strongly δ-typical set [54, Chap. 5] as containing all vectors $\boldsymbol{x}_3 \in \mathcal{A}^{|\mathcal{I}_p(\boldsymbol{b})|}$ which satisfy the following:

$$\begin{aligned}
\text{if } p = 1: \quad & \left|\frac{\#(0;\boldsymbol{x}_3)}{m_1} - \frac{2}{3}\right| + \left|\frac{\#(1;\boldsymbol{x}_3)}{m_1} - \frac{1}{3}\right| \le \delta \\
& \implies \left|\frac{\#(0;\boldsymbol{x}_3)}{m_1} - \frac{2}{3}\right| + \left|\frac{m_1 - \#(0;\boldsymbol{x}_3)}{m_1} - \frac{1}{3}\right| = 2\left|\frac{\#(0;\boldsymbol{x}_3)}{m_1} - \frac{2}{3}\right| \le \delta, \qquad (3.13a) \\
\text{or if } p = 2: \quad & \left|\frac{\#(0;\boldsymbol{x}_3)}{m_2} - \frac{1}{3}\right| + \left|\frac{\#(1;\boldsymbol{x}_3)}{m_2} - \frac{2}{3}\right| \le \delta \\
& \implies \left|\frac{m_2 - \#(1;\boldsymbol{x}_3)}{m_2} - \frac{1}{3}\right| + \left|\frac{\#(1;\boldsymbol{x}_3)}{m_2} - \frac{2}{3}\right| = 2\left|\frac{\#(1;\boldsymbol{x}_3)}{m_2} - \frac{2}{3}\right| \le \delta, \qquad (3.13b)
\end{aligned}$$

where $\#(q;\boldsymbol{x}_3)$ for $q \in \{0,1\}$ counts the components in the vector $\boldsymbol{x}_3$ that are equal to q.

Lemma 11 *For the arithmetic sum function, $I(\boldsymbol{C}_1, \boldsymbol{C}_2; X_3^k | f(X_1^k, X_2^k, X_3^k))/k \to 0$ as $k \to \infty$.*

Proof: Let E denote the event in which the sources $\{X_3^{(j)} \in \mathcal{A} : j \in \mathcal{I}_p(\boldsymbol{b}), f(X_1^k, X_2^k, X_3^k) = \boldsymbol{b}\}$ for $p = 1$ and 2 both belong to their respective strongly δ-typical sets, and $\mathbf{1}\{E\}$ be its indicator

random variable. Let $\mathcal{T}_\beta(f)$ denote the strongly β-typical set of the arithmetic sum $f(X_1^k, X_2^k, X_3^k)$. Then we have the following.

$$
\begin{aligned}
&I(X_3^k; \boldsymbol{C}_1, \boldsymbol{C}_2 | f(X_1^k, X_2^k, X_3^k)) = I(X_3^k; \boldsymbol{C}_1, \boldsymbol{C}_2 | f(X_1^k, X_2^k, X_3^k)) + I(\mathbf{1}\{E\}; \boldsymbol{C}_1, \boldsymbol{C}_2 | f(X_1^k, X_2^k, X_3^k), X_3^k) \\
&= I(X_3^k, \mathbf{1}\{E\}; \boldsymbol{C}_1, \boldsymbol{C}_2 | f(X_1^k, X_2^k, X_3^k)) \\
&= I(\mathbf{1}\{E\}; \boldsymbol{C}_1, \boldsymbol{C}_2 | f(X_1^k, X_2^k, X_3^k)) + I(X_3^k; \boldsymbol{C}_1, \boldsymbol{C}_2 | f(X_1^k, X_2^k, X_3^k), \mathbf{1}\{E\}) \\
&\le H(\mathbf{1}\{E\}) + \sum_{\boldsymbol{b} \notin \mathcal{T}_\beta(f)} \Pr\{f(X_1^k, X_2^k, X_3^k) = \boldsymbol{b}\} I(\boldsymbol{C}_1, \boldsymbol{C}_2; X_3^k | \mathbf{1}\{E\}, f(X_1^k, X_2^k, X_3^k) = \boldsymbol{b}) \\
&\quad + \sum_{\boldsymbol{b} \in \mathcal{T}_\beta(f)} \Pr\{f(X_1^k, X_2^k, X_3^k) = \boldsymbol{b}\} I(\boldsymbol{C}_1, \boldsymbol{C}_2; X_3^k | \mathbf{1}\{E\}, f(X_1^k, X_2^k, X_3^k) = \boldsymbol{b}) \\
&\le 1 + \sum_{\boldsymbol{b} \notin \mathcal{T}_\beta(f)} \Pr\{f(X_1^k, X_2^k, X_3^k) = \boldsymbol{b}\} H(X_3^k) \\
&\quad + \sum_{\boldsymbol{b} \in \mathcal{T}_\beta(f)} \Pr\{f(X_1^k, X_2^k, X_3^k) = \boldsymbol{b}\} \Pr\{\mathbf{1}\{E\} = 0 | f(X_1^k, X_2^k, X_3^k) = \boldsymbol{b}\} \\
&\qquad\qquad \cdot I(\boldsymbol{C}_1, \boldsymbol{C}_2; X_3^k | \mathbf{1}\{E\} = 0, f(X_1^k, X_2^k, X_3^k) = \boldsymbol{b}) \\
&\quad + \sum_{\boldsymbol{b} \in \mathcal{T}_\beta(f)} \Pr\{f(X_1^k, X_2^k, X_3^k) = \boldsymbol{b}\} \Pr\{\mathbf{1}\{E\} = 1 | f(X_1^k, X_2^k, X_3^k) = \boldsymbol{b}\} \\
&\qquad\qquad \cdot I(\boldsymbol{C}_1, \boldsymbol{C}_2; X_3^k | \mathbf{1}\{E\} = 1, f(X_1^k, X_2^k, X_3^k) = \boldsymbol{b}).
\end{aligned}
$$

For a realization $\boldsymbol{b} \in \mathcal{T}_\beta(f)$, we have by definition of the strongly β-typical set

$$
\begin{aligned}
&\left|\frac{|\mathcal{I}_0(\boldsymbol{b})|}{k} - \frac{1}{8}\right| + \left|\frac{m_1}{k} - \frac{3}{8}\right| + \left|\frac{m_2}{k} - \frac{3}{8}\right| + \left|\frac{|\mathcal{I}_3(\boldsymbol{b})|}{k} - \frac{1}{8}\right| \le \beta, \\
&\implies k(3/8 - \beta) \le m_1 \le k(3/8 + \beta),\ k(3/8 - \beta) \le m_2 \le k(3/8 + \beta) \ \text{ and } \ m_1 + m_2 \le 2k(3/8 + \beta).
\end{aligned}
$$

Then for $\boldsymbol{b} \in \mathcal{T}_\beta(f)$ as $k \to \infty$, both $m_1 \to \infty$ and $m_2 \to \infty$ and hence $\Pr(\mathbf{1}\{E\} = 0 | f(X_1^k, X_2^k, X_3^k) = \boldsymbol{b}) \le \delta^2 + 2(1-\delta)\delta \le 2\delta$. Then we have

$$
\begin{aligned}
&I(X_3^k; \boldsymbol{C}_1, \boldsymbol{C}_2 | f(X_1^k, X_2^k, X_3^k)) \\
&\le 1 + \sum_{\boldsymbol{b} \notin \mathcal{T}_\beta(f)} \Pr\{f(X_1^k, X_2^k, X_3^k) = \boldsymbol{b}\} H(X_3^k) + \sum_{\boldsymbol{b} \in \mathcal{T}_\beta(f)} \Pr\{f(X_1^k, X_2^k, X_3^k) = \boldsymbol{b}\} 2\delta H(X_3^k) \\
&\quad + \sum_{\boldsymbol{b} \in \mathcal{T}_\beta(f)} \Pr\{f(X_1^k, X_2^k, X_3^k) = \boldsymbol{b}, \mathbf{1}\{E\} = 1\} I(\boldsymbol{C}_1, \boldsymbol{C}_2; X_3^k | \mathbf{1}\{E\} = 1, f(X_1^k, X_2^k, X_3^k) = \boldsymbol{b})
\end{aligned}
$$

$$\leq 1 + \beta k \log|\mathcal{A}| + 2\delta k \log|\mathcal{A}|$$
$$+ \sum_{\boldsymbol{b}\in\mathcal{T}_\beta(f)} \Pr\{\mathbf{1}\{E\}=1, f(X_1^k,X_2^k,X_3^k)=\boldsymbol{b}\}\Big[H(\boldsymbol{C}_1,\boldsymbol{C}_2|\mathbf{1}\{E\}=1, f(X_1^k,X_2^k,X_3^k)=\boldsymbol{b})$$
$$- H(\boldsymbol{C}_1,\boldsymbol{C}_2|X_3^k,\mathbf{1}\{E\}=1, f(X_1^k,X_2^k,X_3^k)=\boldsymbol{b})\Big].$$

Let $A_3^E(\boldsymbol{b}) \subseteq A_3(\boldsymbol{b})$ denote the set of realizations of X_3^k which result in the event E occurring. Then for the clumpy distribution, we have that

$$H(\boldsymbol{C}_1,\boldsymbol{C}_2|\mathbf{1}\{E\}=1, f(X_1^k,X_2^k,X_3^k)=\boldsymbol{b}) = H\left(\frac{\sum_{\boldsymbol{x}_3\in A_3^E(\boldsymbol{b})}\boldsymbol{h}_{\boldsymbol{b},\boldsymbol{x}_3}}{\sum_{\boldsymbol{x}_3\in A_3^E(\boldsymbol{b})}|A_{123}(\boldsymbol{b},\boldsymbol{x}_3)|}\right),$$
$$H(\boldsymbol{C}_1,\boldsymbol{C}_2|X_3^k,\mathbf{1}\{E\}=1, f(X_1^k,X_2^k,X_3^k)=\boldsymbol{b}) = \sum_{\boldsymbol{x}_3\in A_3^E(\boldsymbol{b})}\frac{|A_{123}(\boldsymbol{b},\boldsymbol{x}_3)|}{(\sum_{\boldsymbol{x}_3\in A_3^E(\boldsymbol{b})}|A_{123}(\boldsymbol{b},\boldsymbol{x}_3)|)}H\left(\frac{\boldsymbol{h}_{\boldsymbol{b},\boldsymbol{x}_3}}{|A_{123}(\boldsymbol{b},\boldsymbol{x}_3)|}\right).$$

For the arithmetic sum, we have that $|A_{123}(\boldsymbol{b},\boldsymbol{a}_3)| = 2^{t_{1,0}}2^{t_{2,1}}$ and $\boldsymbol{h}_{\boldsymbol{b},\boldsymbol{a}_3} = \mathbf{1}_{|A_{123}(\boldsymbol{b},\boldsymbol{a}_3)|}$. Hence we get that

$$H\left(\frac{\boldsymbol{h}_{\boldsymbol{b},\boldsymbol{x}_3}}{|A_{123}(\boldsymbol{b},\boldsymbol{x}_3)|}\right) = \log|A_{123}(\boldsymbol{b},\boldsymbol{x}_3)| = t_{1,0}+t_{2,1}.$$

By equations (3.13a), (3.13b), for any $\boldsymbol{b}\in\mathcal{T}_\beta(f)$ and $\boldsymbol{x}_3\in A_3^E(\boldsymbol{b})$, we have that $2|t_{1,0}-2m_1/3|\leq\delta m_1$ and $2|t_{2,1}-2m_1/3|\leq\delta m_2$. Then we have that

$$H\left(\frac{\sum_{\boldsymbol{x}_3\in A_3^E(\boldsymbol{b})}|A_{123}(\boldsymbol{b},\boldsymbol{x}_3)|(\boldsymbol{h}_{\boldsymbol{b},\boldsymbol{x}_3}/|A_{123}(\boldsymbol{b},\boldsymbol{x}_3)|)}{\sum_{\boldsymbol{x}_3\in A_3^E(\boldsymbol{b})}|A_{123}(\boldsymbol{b},\boldsymbol{x}_3)|}\right) \leq \max_{\boldsymbol{x}_3\in A_3^E(\boldsymbol{b})} H\left(\frac{\boldsymbol{h}_{\boldsymbol{b},\boldsymbol{x}_3}}{|A_{123}(\boldsymbol{b},\boldsymbol{x}_3)|}\right)$$
$$\leq (m_1+m_2)(2/3+\delta/2),$$
$$\sum_{\boldsymbol{x}_3\in A_3^E(\boldsymbol{b})}\frac{|A_{123}(\boldsymbol{b},\boldsymbol{x}_3)|}{(\sum_{\boldsymbol{x}_3\in A_3^E(\boldsymbol{b})}|A_{123}(\boldsymbol{b},\boldsymbol{x}_3)|)}H\left(\frac{\boldsymbol{h}_{\boldsymbol{b},\boldsymbol{x}_3}}{|A_{123}(\boldsymbol{b},\boldsymbol{x}_3)|}\right) \geq \min_{\boldsymbol{x}_3\in A_3^E(\boldsymbol{b})} H\left(\frac{\boldsymbol{h}_{\boldsymbol{b},\boldsymbol{x}_3}}{|A_{123}(\boldsymbol{b},\boldsymbol{x}_3)|}\right)$$
$$\geq (m_1+m_2)(2/3-\delta/2).$$

Thus we have that

$$I(X_3^k;\boldsymbol{C}_1,\boldsymbol{C}_2|f(X_1^k,X_2^k,X_3^k))$$
$$\leq 1+\beta k\log|\mathcal{A}| + 2\delta k\log|\mathcal{A}| + \sum_{\boldsymbol{b}\in\mathcal{T}_\beta(f)}\Pr(\mathbf{1}\{E\}=1, f(X_1^k,X_2^k,X_3^k)=\boldsymbol{b})(m_1+m_2)\delta$$
$$\leq 1+\beta k\log|\mathcal{A}| + 2\delta k\log|\mathcal{A}| + \sum_{\boldsymbol{b}\in\mathcal{T}_\beta(f)}\Pr(\mathbf{1}\{E\}=1, f(X_1^k,X_2^k,X_3^k)=\boldsymbol{b})2k(3/8+\beta)\delta$$

$$\implies I(X_3^k; \boldsymbol{C}_1, \boldsymbol{C}_2 | f(X_1^k, X_2^k, X_3^k))/k \leq 1/k + \beta \log |\mathcal{A}| + 2\delta \log |\mathcal{A}| + 2(3/8 + \beta)\delta.$$

Choosing $\beta \to 0$ and $\delta \to 0$ and consequently $k \to \infty$, gives us the result. ■

3.3.4 Example demand function: Sum over $GF(2)$

Suppose the messages $X_1, X_2, X_3 \in \{0, 1\}$ and the terminal wants to compute their finite field sum over $GF(2)$. For this demand function, we demonstrate that the outer bound to the rate region is tight. We assume that the alphabet used for communication is $\mathcal{Z} = \{0, 1\}$.

For X_3 we have that $0 \not\equiv 1$, and thus for a function $g(X_3^k)$ that returns the equivalence class that X_3^k belongs to, we have that $H(g(X_3^k)) = k$. Thus from Lemma 4 we get that $R_{31} + R_{32} \geq 1$.

For X_u, $u = 1$ or 2, using the values of the finite field sum, we obtain the following partitions

$$\overset{0}{\equiv}|_u : \{0\} \cup \{1\} \quad \text{and} \quad \overset{1}{\equiv}|_u : \{0\} \cup \{1\}.$$

Thus, similar to the case of arithmetic sum (*c.f.* Section 3.3.3), for a $GF(2)$-sum realization $\boldsymbol{b}$ and $\boldsymbol{a}_3 \in A_3(\boldsymbol{b})$, the equivalence classes satisfy $|\mathrm{Cl}_1^{(\boldsymbol{v})}(\boldsymbol{a}_3)| = |\mathrm{Cl}_2^{(\boldsymbol{w})}(\boldsymbol{x}_3)| = 1$ for any class index $\boldsymbol{v}$ or $\boldsymbol{w}$ and $V_1(\boldsymbol{a}_3) = V_2(\boldsymbol{a}_3) = 2^k$. Thus the vector $\boldsymbol{d}_u(\boldsymbol{a}_3)$ defined in Lemma 7 is $\mathbf{1}_{2^k}$, and hence $H(\boldsymbol{d}_u(\boldsymbol{a}_3)/2^k) = k$, giving us the value of γ as

$$\gamma = 1 \implies R_u + \epsilon \geq 1.$$

For any value of X_1^k, X_3^k and $f(X_1^k, X_2^k, X_3^k)$ the value of X_2^k is fixed by $X_2^k = f(X_1^k, X_2^k, X_3^k) - X_1^k - X_3^k$, where the subtraction operations are also over $GF(2)$. Hence we have that

$$h_{\boldsymbol{a}_3}(\boldsymbol{v}, \boldsymbol{w}) = 1 \text{ for every } \boldsymbol{a}_3 \in A_3(\boldsymbol{b}) \quad \text{and} \quad (\mathrm{Cl}_1^{(\boldsymbol{v})}, \mathrm{Cl}_2^{(\boldsymbol{w})}) \in \mathcal{V}_{12}(\boldsymbol{a}_3, \boldsymbol{b}).$$

We enumerate the different (X_1, X_2) pairs that result in $b^{(i)}$ for the realization $a_3^{(i)}$ in different cases as below.

- If $i \in \mathcal{I}_0(\boldsymbol{b}) \cap \mathcal{J}_0(\boldsymbol{a}_3)$: $(X_1, X_2) \in \{(0, 0), (1, 1)\}$.
- If $i \in \mathcal{I}_0(\boldsymbol{b}) \cap \mathcal{J}_1(\boldsymbol{a}_3)$: $(X_1, X_2) \in \{(0, 1), (1, 0)\}$.

- If $i \in \mathcal{I}_1(\boldsymbol{b}) \cap \mathcal{J}_0(\boldsymbol{a}_3)$: $(X_1, X_2) \in \{(0,1),(1,0)\}$.
- If $i \in \mathcal{I}_1(\boldsymbol{b}) \cap \mathcal{J}_1(\boldsymbol{a}_3)$: $(X_1, X_2) \in \{(0,0),(1,1)\}$.

Since there are two choices in each case, we have that $|A_{123}(\boldsymbol{b}, \boldsymbol{a}_3)| = 2^k$. Thus we have that $\boldsymbol{h}_{\boldsymbol{b},\boldsymbol{a}_3}/|A_{123}(\boldsymbol{b}, \boldsymbol{a}_3)| = \mathbf{1}_{2^k}$ which gives the value of α as

$$\alpha = 1 \implies (R_1 + R_2)/2 + \epsilon \geq (1 + H(f(X_1^k, X_2^k, X_3^k))/k)/2 = (1 + k/k)/2 = 1.$$

We describe simple network code that allows t to compute the $GF(2)$-sum by carrying out the operation $\boldsymbol{Z}_1(X_1^k, \boldsymbol{Z}_{31}(X_3^k)) + \boldsymbol{Z}_2(X_2^k, \boldsymbol{Z}_{32}(X_3^k))$. The codewords used are

$$\boldsymbol{Z}_{31}(X_3^k) = X_3^k,\ \boldsymbol{Z}_{32}(X_3^k) = 0.$$
$$\boldsymbol{Z}_1(X_1^k, \boldsymbol{Z}_{31}(X_3^k)) = X_1^k + X_3^k,\ \boldsymbol{Z}_2(X_2^k, \boldsymbol{Z}_{32}(X_3^k)) = X_2^k.$$

The sums computed above are component-wise over $GF(2)$. Then $\mathbb{E}\ell(\boldsymbol{Z}_{31}) = k, \mathbb{E}\ell(\boldsymbol{Z}_{32}) = 0$ and $\mathbb{E}\ell(\boldsymbol{Z}_2) = k$. If $X_1, X_3 \sim \text{Unif}\{0,1\}$ then $X_1 + X_3 \sim \text{Unif}\{0,1\}$. Hence $\mathbb{E}\ell(\boldsymbol{Z}_1) = k$. These imply that the rate tuple achieved is

$$(R_{31}, R_{32}, R_1, R_2) = (1, 0, 1, 1),$$

which match the lower bounds derived above.

3.4 Conclusions and future work

In this paper, we have described a procedure to obtain an outer bound for the rate region (in the setup of [50]) for computing a function with zero-error over a simple DAG network. The demand function can be an arbitrary discrete-valued function and only needs to be specified as a function table. For computing the arithmetic sum of three bits, we show that the outer bound obtained is tighter than the one in [49]. For computing the $GF(2)$-sum of three bits, we show that the lower bounds for the rate tuple obtained using our procedure can also be achieved by a simple network code. Our method uses the equivalence relations defined in [47] as adapted to the specific DAG network considered here. Assuming a independent, uniform probability distribution for each

of the messages, we compute the probability that the messages belong to a particular equivalence class. These are used in obtaining a lower bound to the conditional entropy of the descriptions transmitted on the edges, which imply an outer bound to the rate region.

There are many opportunities for future work. An immediate question is to find the exact rate region for computing the arithmetic sum over the DAG network considered here. This function computation problem has been instructive in characterizing the worst-case communication necessary for computation, and we expect it to play a similar role in finding the rate region for average-case communication scenario. Progress in this direction should give us insights in obtaining outer bounds for the rate region for computing functions other than the arithmetic sum on this network. A different direction is to consider function computation over other directed acyclic graph networks. An interesting question is whether we can 'overlay' the probability information, in a manner similar to as done here, on the various equivalence classes as given in [47] for an arbitrary DAG network and demand function and obtain corresponding outer bounds on the rate region.

CHAPTER 4. CONCLUSIONS AND FUTURE WORK

In our study on computing functions over networks, we have demonstrated certain characteristics of the general problem. We have done this by focusing on certain tractable problem instances.

Sum-networks are function computation instances in which we are interested in computing the finite field sum of messages observed in a network. The significance of the capacity problem for a sum-network is underscored by a known reduction in the literature, which relates whether a multiple-unicast problem can be solved at rate = 1 to whether the computation capacity of an appropriate sum-network is 1. Multiple-unicast is a very general communication problem in which several source-terminal pairs communicate their respective messages over a shared network. Finding the computation capacity of a general sum-network is a well-known hard problem. We were able to find the computation capacity of an infinite family of sum-networks that were obtained using incidence structures. Depending on the structure of the sum-network, the computation capacity of a sum-network was seen to greatly depend on the characteristic of the finite field over which the sum is computed.

Computing the arithmetic sum of three bits over the directed acyclic graph network considered in Chapter 3 has served as an illuminating example for various reasons. It is the smallest network which does not have a tree structure. The computation capacity of networks with a tree structure is known in the literature. Nevertheless, the *worst-case* computation capacity for computing the arithmetic sum over this specific non-tree network was known in the literature through a combinatorial argument. However, as we have shown, the *average-case* computation capacity for this problem is still unknown. For the average-case, we worked in the framework of a *source-network* code and consequently the quantity of interest is a *rate-region* $\subseteq \mathbb{R}^4_{\geq 0}$. The inner bound to the rate region demonstrates that we can compute the arithmetic sum by communicating less than what is required in the worst-case scenario, as one would expect. The technique by which we arrive at the

outer bound to the rate region can also be applied to functions other than the arithmetic sum, as was demonstrated.

Key ideas used in Chapter 2

- We used incidence structures to construct our sum-network problem instances. The symmetry in the incidence structure then translated into a symmetry in the network structure.
- The incidence between points and blocks of the incidence structure allowed us to determine the minimum amount of independent information that must be transmitted across the bottleneck edges in the sum-network. It was equal to the rank of a particular 0-1 matrix, where the rank was evaluated over the finite field over which the sum is to be computed. This gave us a finite-field dependent upper bound to the computation capacity of the sum-networks.
- We used certain non-negative integer matrices to obtain network codes whose rate matched the upper bound. The existence of these matrices was shown using necessary and sufficient conditions for the existence of integral flows over a flow-network. For several families of incidence structures, we could verify that they satisfied these conditions by double-counting the number of 1's in various submatrices of the associated incidence matrix.
- There are sum-networks whose computation capacity is 1 over a finite field but close to 0 over a different finite field. This was established using existence results for balanced incomplete block designs known in the literature.

Key ideas used in Chapter 3

- It is well-known that one can describe a sequence of independent and identically distributed random variables using codeword whose expected length per symbol is asymptotically close to the entropy of the random variable. We use this idea to find the length of the codewords that are generated by the encoders at s_1 and s_2 in terms of the entropy of the codewords. This allows us to reduce the average amount of communication required.

- It is known in the literature that communicating which equivalence class a message belongs to is sufficient for correctly computing the function. Using this idea and applying the pigeon-hole principle, we can specify a family of probability mass functions that must contain the conditional probability mass function of the codewords given the function value and the message observed at s_3.

- A lower bound to the conditional entropy is found by evaluating the entropy of a probability mass function that majorizes every other probability mass function in the family. This is because of the Schur concave property of the entropy function. This lower bound is used to get an outer bound to the rate region.

4.1 Future work

There are many avenues for future work within the general problem setup of this thesis for computing functions over networks. Some of them were highlighted in the conclusion sections of Chapters 2 and 3. Here we outline some other general directions of future work. For example, in the case of computing the finite field sum, what methods could be used to find the computation capacity of other sum-networks whose structure is not as symmetric as those obtained using incidence structures. For the class of sum-networks constructed using our procedure, there exist sum-networks for which we have an upper bound to the computation capacity but no network code with a matching rate.

For the particular directed acyclic graph considered in Chapter 3, an immediate question is to find the exact rate region for the computing arithmetic sum. This problem has been instructive for finding the worst-case communication necessary for computation, and we expect it to play a similar role in finding the average-case communication scenario. Progress in this direction should give us insights for obtaining outer bounds for the rate region for computing functions other than the arithmetic sum on this network.

BIBLIOGRAPHY

[1] J. Körner and K. Marton, "How to encode the modulo-2 sum of binary sources," *IEEE Trans. on Info. Th.*, vol. 25, no. 2, pp. 219–221, 1979.

[2] A. Orlitsky and J. Roche, "Coding for computing," *IEEE Trans. on Info. Th.*, vol. 47, no. 3, pp. 903–917, Mar 2001.

[3] V. Doshi, D. Shah, M. Médard, and M. Effros, "Functional compression through graph coloring," *IEEE Trans. on Info. Th.*, vol. 56, no. 8, pp. 3901–3917, Aug 2010.

[4] R. Ahlswede, N. Cai, S.-Y. Li, and R. W. Yeung, "Network Information Flow," *IEEE Trans. on Info. Th.*, vol. 46, no. 4, pp. 1204–1216, 2000.

[5] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 782–795, Oct 2003.

[6] S.-Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. on Info. Th.*, vol. 49, no. 2, pp. 371–381, Feb 2003.

[7] R. Dougherty, C. Freiling, and K. Zeger, "Insufficiency of linear coding in network information flow," *IEEE Trans. on Info. Th.*, vol. 51, no. 8, pp. 2745–2759, Aug. 2005.

[8] J. Cannons, R. Dougherty, C. Freiling, and K. Zeger, "Network routing capacity," *IEEE Trans. on Info. Th.*, vol. 52, no. 3, pp. 777–788, Mar. 2006.

[9] R. Dougherty, C. Freiling, and K. Zeger, "Unachievability of network coding capacity," *IEEE Trans. on Info. Th.*, vol. 52, no. 6, pp. 2365–2372, June 2006.

[10] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. M. G. M. Tolhuizen, "Polynomial time algorithms for multicast network code construction," *IEEE Trans. on Info. Th.*, vol. 51, no. 6, pp. 1973–1982, June 2005.

[11] S. Huang and A. Ramamoorthy, "An achievable region for the double unicast problem based on a minimum cut analysis," *IEEE Trans. on Comm.*, vol. 61, no. 7, pp. 2890–2899, 2013.

[12] S. Huang and A. Ramamoorthy, "On the multiple unicast capacity of 3-source, 3-terminal directed acyclic networks," *IEEE/ACM Trans. on Networking*, vol. 22, no.1, pp. 285–299, 2014.

[13] A. R. Lehman and E. Lehman, "Complexity classification of network information flow problems," in *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, Jan 2004, pp. 142–150.

[14] S. Kamath, D. N. C. Tse, and C. C. Wang, "Two-unicast is hard," in *IEEE Intl. Symposium on Info. Th.*, June 2014, pp. 2147–2151.

[15] R. Appuswamy, M. Franceschetti, N. Karamchandani, and K. Zeger, "Network coding for computing: Cut-set bounds," *IEEE Trans. on Info. Th.*, vol. 57, no. 2, pp. 1015–1030, Feb 2011.

[16] C. Huang, Z. Tan, and S. Yang, "Upper bound on function computation in directed acyclic networks," in *IEEE Info. Th. Workshop*, April 2015, pp. 1–5.

[17] R. Appuswamy, M. Franceschetti, N. Karamchandani, and K. Zeger, "Linear codes, target function classes, and network computing capacity," *IEEE Trans. on Info. Th.*, vol. 59, no. 9, pp. 5741–5753, Sept 2013.

[18] A. Ramamoorthy, "Communicating the sum of sources over a network," in *IEEE Intl. Symposium on Info. Th.*, July 2008, pp. 1646–1650.

[19] B. K. Rai and B. K. Dey, "On network coding for sum-networks," *IEEE Trans. on Info. Th.*, vol. 58, no. 1, pp. 50 –63, 2012.

[20] A. Ramamoorthy and M. Langberg, "Communicating the sum of sources over a network," *IEEE J. Select. Areas Comm.*, vol. 31, no. 4, pp. 655–665, 2013.

[21] D. R. Stinson, *Combinatorial Designs: Construction and Analysis*. Springer, 2003.

[22] O. Olmez and A. Ramamoorthy, "Fractional repetition codes with flexible repair from combinatorial designs," *IEEE Trans. on Info. Th.*, vol. 62, no. 4, pp. 1565 –1591, 2016.

[23] S. E. Rouayheb and K. Ramchandran, "Fractional repetition codes for repair in distributed storage systems," in 48^{th} *Annual Allerton Conference on Comm., Control & Computing*, Sept 2010, pp. 1510 –1517.

[24] L. Tang and A. Ramamoorthy, "Coded caching with low subpacketization levels," in *Intl. Symp. on Network Coding (NetCod)*, Dec 2016, pp. 1–6.

[25] L. Tang and A. Ramamoorthy, "Coded Caching Schemes with Reduced Subpacketization from Linear Block Codes," *IEEE Trans. on Info. Th.*, vol. 64, no. 4, pp. 3099–3120, 2018.

[26] L. Tang and A. Ramamoorthy, "Coded caching for networks with the resolvability property," in *IEEE Intl. Symposium on Info. Th.*, July 2016, pp. 420-424.

[27] L. Tang and A. Ramamoorthy, "Low subpacketization schemes for coded caching," in *IEEE Intl. Symposium on Info. Th.*, June 2017, pp. 2790-2794.

[28] B. K. Rai and N. Das, "On the capacity of ms/3t and 3s/nt sum-networks," in *IEEE Info. Th. Workshop*, Sep. 2013, pp. 1–5.

[29] B. Rai and N. Das, "On the capacity of sum-networks," in 51^{st} *Annual Allerton Conference on Comm., Control & Computing*, Oct 2013, pp. 1545–1552.

[30] A. Tripathy and A. Ramamoorthy, "Sum-networks from undirected graphs: Construction and capacity analysis," in 52^{nd} *Annual Allerton Conference on Comm., Control & Computing*, Sept 2014, pp. 651–658.

[31] N. Das and B. K. Rai, "On the number of sources and terminals of sum-networks with capacity p/q," in 21^{st} *National Conference on Communications (NCC)*, Feb 2015, pp. 1–6.

[32] A. Tripathy and A. Ramamoorthy, "Capacity of sum-networks for different message alphabets," in *IEEE Intl. Symposium on Info. Th.*, June 2015, pp. 606–610.

[33] R. A. Brualdi, *Combinatorial matrix classes*. Cambridge University Press, 2006, vol. 13.

[34] L. Mirsky, "Combinatorial theorems and integral matrices," *Journal of Combinatorial Theory*, vol. 5, no. 1, pp. 30–44, 1968.

[35] L. Teirlinck, "Non-trivial t-designs without repeated blocks exist for all t," *Discrete Mathematics*, vol. 65, no. 3, pp. 301–311, 1987.

[36] C. J. Colbourn and J. H. Dinitz, *Handbook of combinatorial designs*. CRC press, 2006.

[37] A. Giridhar and P. R. Kumar, "Computing and communicating functions over sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 4, pp. 755-764, April 2005.

[38] A. E. Gamal and Y.-H. Kim, *Network Information Theory*. Cambridge University Press, Jan 2012.

[39] H. Witsenhausen, "The zero-error side information problem and chromatic numbers (Corresp.)," *IEEE Trans. on Info. Th.*, vol. 22, no. 5, pp. 592-593, September 1976.

[40] N. Alon and A. Orlitsky, "Source coding and graph entropies," *IEEE Trans. on Info. Th.*, vol. 42, no. 5, pp. 1329-1339, Sep 1996.

[41] J. Körner and A. Orlitsky, "Zero-error information theory," *IEEE Trans. on Info. Th.*, vol. 44, no. 6, pp. 2207-2229, Oct 1998.

[42] T. S. Han and K. Kobayashi, "A dichotomy of functions $F(X, Y)$ of correlated sources (X, Y) from the viewpoint of the achievable region," *IEEE Trans. on Info. Th.*, vol. 33, no. 1, pp. 69-76, January 1987.

[43] S. Feizi and M. Médard, "On Network Functional Compression," *IEEE Trans. on Info. Th.*, vol. 60, no. 9, pp. 5387-5401, Sept. 2014.

[44] M. Sefidgaran and A. Tchamkerten, "Distributed Function Computation Over a Rooted Directed Tree," *IEEE Trans. on Info. Th.*, vol. 62, no. 12, pp. 7135-7152, Dec. 2016.

[45] H. Kowshik and P. R. Kumar, "Optimal Function Computation in Directed and Undirected Graphs," *IEEE Trans. on Info. Th.*, vol. 58, no. 6, pp. 3407–3418, June 2012.

[46] T. Cover and J. Thomas, *Elements of Information Theory*, ser. Wiley Series in Telecommunications and Signal Processing. Wiley-Interscience, 2006.

[47] X. Guang, S. Yang, and C. Li, "An improved upper bound on network function computation using cut-set partition," in *IEEE Info. Th. Workshop*, Sept 2016, pp. 11–15.

[48] A. W. Marshall, I. Olkin, and B. C. Arnold, *Inequalities: Theory of Majorization and Its Applications*, ser. Springer Series in Statistics. Springer New York, 2011.

[49] A. Tripathy and A. Ramamoorthy, "On computation rates for arithmetic sum," in *IEEE Intl. Symposium on Info. Th.*, July 2016, pp. 2354–2358.

[50] L. Song, R. W. Yeung and N. Cai, "Zero-error network coding for acyclic networks," in *IEEE Trans. on Info. Th.*, vol. 49, no. 12, pp. 3129–3139, Dec. 2003.

[51] S. Leung-Yan-Cheong and T. Cover, "Some equivalences between Shannon entropy and Kolmogorov complexity," *IEEE Trans. on Info. Th.*, vol. 24, no. 3, pp. 331–338, May 1978.

[52] X. Guang, R. W. Yeung, S. Yang and C. Li, "Improved upper bound on the network function computing capacity," online at http://arxiv.org/abs/1710.02252.

[53] K. Konstantinidis and A. Ramamoorthy, "Leveraging Coding Techniques for Speeding up Distributed Computing," online at http://arxiv.org/abs/1802.03049.

[54] R. W. Yeung, *A First Course in Information Theory*, ser. Information Technology: Transmission, Processing and Storage. Springer US, 2002.

[55] A. Tripathy and A. Ramamoorthy, "Sum-networks from incidence structures: construction and capacity analysis," *IEEE Trans. on Info. Th.*, vol. 64, no. 5, pp. 3461–3480, May 2018.

APPENDIX A. NON-APPLICABILITY OF THEOREM VI.5 IN [8] FOR SUM-NETWORKS

The capacity of multiple-unicast networks is known to be independent of the alphabet chosen for communication [8, Theorem VI.5]. The core idea there was this. Consider alphabets $\mathcal{F}_1, \mathcal{F}_2$ of different cardinality. Suppose there exists a (m_1, n_1) network code over $\mathcal{F}_1$ that satisfies the demands of every terminal in the network. Then for any $\epsilon > 0$, [8] described a procedure to simulate a (m_2, n_2) network code over $\mathcal{F}_2$ using the (m_1, n_1) network code over $\mathcal{F}_1$ such that $m_2/n_2 \geq m_1/n_1 - \epsilon$. The values of the parameters m_1, n_1, m_2, n_2 are determined by the value of ϵ and $|\mathcal{F}_1|, |\mathcal{F}_2|$. The simulation procedure uses two one-to-one functions $\boldsymbol{h_0} : \mathcal{F}_2^{m_2} \to \mathcal{F}_1^{m_1}$ and $\boldsymbol{h} : \mathcal{F}_1^{n_1} \to \mathcal{F}_2^{n_2}$. We informally describe the simulation procedure as applied to every component in a multiple-unicast network below.

- At each source node with no incoming edges: A message in $\mathcal{F}_2^{m_2}$ is observed at a source node. This message is mapped to a symbol in $\mathcal{F}_1^{m_1}$ using the function $\boldsymbol{h_0}$. This symbol is used as an argument for the encoding function of this node in the (m_1, n_1) network code; the value returned belongs to the set $\mathcal{F}_1^{n_1}$. The value returned by the encoding function is then mapped by $\boldsymbol{h}$ to an element in $\mathcal{F}_2^{n_2}$, which is transmitted along the outgoing edge.

- At each intermediate node in the network: Each intermediate node observes as many values from $\mathcal{F}_2^{n_2}$ as the number of its incoming edges. Since $\boldsymbol{h}$ is a one-to-one function, for each received symbol in $\mathcal{F}_2^{n_2}$ the node can obtain its pre-image under $\boldsymbol{h}$ in $\mathcal{F}_1^{n_1}$. After obtaining the pre-images for each received value, the node can use them as arguments for its encoding function in the (m_1, n_1) network code and obtain the values that must be transmitted along its outgoing edges. These returned values are in $\mathcal{F}_1^{n_1}$ and they are mapped to symbols in $\mathcal{F}_2^{n_2}$ by $\boldsymbol{h}$ before transmission.

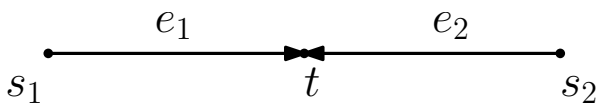


Figure A.1: A simple sum-network. Both edges can transmit one symbol in $\mathcal{F}_1$ from tail to head in one channel use.

- Decoding at each terminal node in the network: At each terminal node, the received values in $\mathcal{F}_2^{n_2}$ are mapped to their pre-images in $\mathcal{F}_1^{n_1}$ under $\boldsymbol{h}$. These pre-images are used as arguments for the decoding function of this terminal in the (m_1, n_1) network code. The value returned by the decoding function is an element of $\mathcal{F}_1^{m_1}$ that is the image under $\boldsymbol{h_0}$ of the demanded message at this terminal. Since $\boldsymbol{h_0}$ is also a one-to-one function, each terminal can recover its required message.

This simulation procedure however cannot be applied to sum-networks as is illustrated by the example below.

Example 13 *Consider a simple sum-network shown in Figure A.1, terminal t wants to evaluate $X_1 + X_2$ where $X_1, X_2 \in \mathcal{F}_1$ are random variables observed at source nodes s_1, s_2 respectively. We have a scalar network code (rate $= 1$) that satisfies the problem, described as follows.*

1. *Edge functions:*

$$\phi_{e_1}(X_1) = X_1, \quad \phi_{e_2}(X_2) = X_2.$$

2. *Decoding function:*

$$\psi(\phi_{e_1}(X_1), \phi_{e_2}(X_2)) = \phi_{e_1}(X_1) + \phi_{e_2}(X_2) = X_1 + X_2$$

and $X_1 + X_2$ is the only value terminal t is interested in decoding.

We use the procedure outlined in [8] to extend the network code for another alphabet $\mathcal{F}_2$. Let $\mathcal{F}_1 = GF(3), \mathcal{F}_2 = GF(2)$. Setting $\epsilon = 2^{1-\gamma}/\log_2 3$ where $\gamma > 1$, we obtain the following values

$$m_1 = n_1 = \lceil 2^\gamma \rceil, n_2 = \left\lceil \frac{2^\gamma}{\log_2 3} \right\rceil \text{ and } m_2 = n_2 - 1.$$

Let $h_0 : \mathcal{F}_2 \to \mathcal{F}_1$ *be such that*

$$h_0(x) = \begin{cases} 0 & \text{if } x = 0, \\ 1 & \text{if } x = 1. \end{cases}$$

and let $\hat{h}_0 : \mathcal{F}_1 \to \mathcal{F}_2$ *be such that* $\hat{h}_0(h_0(x)) = x$ *for all* $x \in \mathcal{F}_2$ *and arbitrary otherwise. Then we can define an injection* $\boldsymbol{h_0} : \mathcal{F}_2^{m_2} \to \mathcal{F}_1^{m_1}$ *as the component-wise application of* h_0 *to each of the elements in the argument.*

$$\boldsymbol{h_0}(b_1, b_2, \ldots, b_{m_2}) = \begin{bmatrix} h_0(b_1) & h_0(b_2) & \ldots & h_0(b_{m_2}) & \mathbf{0}_{m_1 - m_2} \end{bmatrix}$$

where $b_1, b_2, \ldots b_{m_2} \in \mathcal{F}_2$ *and* $\mathbf{0}_{m_1 - m_2}$ *is a zero vector with* $m_1 - m_2$ *components. We define* $\hat{\boldsymbol{h}}_{\mathbf{0}} : \mathcal{F}_1^{m_1} \to \mathcal{F}_2^{m_2}$ *as*

$$\hat{\boldsymbol{h}}_{\mathbf{0}}(a_1, a_2, \ldots, a_{m_1}) = \begin{bmatrix} \hat{h}_0(a_1) & \hat{h}_0(a_2) & \ldots & \hat{h}_0(a_{m_2}) \end{bmatrix}$$

where $a_1, a_2, \ldots, a_{m_1} \in \mathcal{F}_1$.

Also we let $\boldsymbol{h} : \mathcal{F}_1^{n_1} \to \mathcal{F}_2^{n_2}$ *be an arbitrary injection and* $\hat{\boldsymbol{h}} : \mathcal{F}_2^{n_2} \to \mathcal{F}_1^{n_1}$ *is such that* $\hat{\boldsymbol{h}}(\boldsymbol{h}(\boldsymbol{x})) = \boldsymbol{x}$ *for all* $\boldsymbol{x} \in \mathcal{F}_1^{n_1}$ *and arbitrary otherwise. This is possible because* $3^{\lceil 2\gamma \rceil} \geq 2^{\lceil 2\gamma / \log_2 3 \rceil}$ *for any* $\gamma > 1$. *We now use the extended network code to satisfy the sum network for when the source random variables take values in the alphabet* $\mathcal{F}_2^{m_2}$. *Suppose a particular realization of* $X_1 \in \mathcal{F}_2^{m_2}$ *and* $X_2 \in \mathcal{F}_2^{m_2}$ *is such that*

$$\boldsymbol{x_1} = (1, 1, \ldots, 1) = \mathbf{1}_{m_2} \text{ and } \boldsymbol{x_2} = (1, 1, \ldots, 1) = \mathbf{1}_{m_2}.$$

Following steps in [8] for the decoding function we get that terminal t *carries out the following operation to obtain the value of* $\boldsymbol{x_1} + \boldsymbol{x_2}$

$$\begin{aligned} \hat{\boldsymbol{h}}_{\mathbf{0}}(\psi_t(\phi_{e_1}(\boldsymbol{h_0}(\boldsymbol{x_1})), \phi_{e_2}(\boldsymbol{h_0}(\boldsymbol{x_2})))) &= \hat{\boldsymbol{h}}_{\mathbf{0}}(\boldsymbol{h_0}(\boldsymbol{x_1}) + \boldsymbol{h_0}(\boldsymbol{x_2})) \\ &= \hat{\boldsymbol{h}}_{\mathbf{0}}([\mathbf{1}_{m_2} \ \mathbf{0}_{m_1 - m_2}] + [\mathbf{1}_{m_2} \ \mathbf{0}_{m_1 - m_2}]) \\ &= \hat{\boldsymbol{h}}_{\mathbf{0}}([\mathbf{2}_{m_2} \ \mathbf{0}_{m_1 - m_2}]) \end{aligned}$$

where $\mathbf{2}_{m_2}$ *is a vector of* m_2 2*'s.*

Since $\hat{h}_0(2)$ *is arbitrarily assigned,* $\hat{\boldsymbol{h}}_{\mathbf{0}}([\mathbf{2}_{m_2}\ \mathbf{0}_{m_1-m_2}])$ *need not equal* $\mathbf{0}_{m_2}$ *which is the right value of* $\boldsymbol{x_1}+\boldsymbol{x_2}$*. Thus the simulated* (m_2, n_2) *network code over* $\mathcal{F}_2$ *does not correctly evaluate the required sum.*

In fact, the computation capacity of sum-networks explicitly depends on the alphabet used in message generation and communication, as described in Sections 2.5, 2.6. Moreover, the choice of alphabet can significantly reduce the computation capacity of the same sum-network as discussed in Section 2.7.

APPENDIX B. PROOF OF LEMMA 6 IN CHAPTER 3

Proof: We use the same argument for $u = 1$ and 2. Partition the set $\mathcal{A}^k$ into $\prod_{i=1}^{k} V_u(a_3^{(i)})$ disjoint subsets based on the equivalence relation $\overset{a_3^{(i)}}{\equiv}|_u$ in each component $i \in \{1, 2, \ldots, k\}$. Then every element $(x_u^{(1)}, x_u^{(2)}, \ldots, x_u^{(k)}) \in \mathcal{A}^k$ belongs to a corresponding subset in the partition. Suppose the number of distinct $\boldsymbol{Z}_u$-labels transmitted on (s_u, t) is strictly less than $\prod_{i=1}^{k} V_u(a_3^{(i)})$. Then by the pigeon-hole principle, there exist two elements $\boldsymbol{x}_u \triangleq (x_u^{(1)}, x_u^{(2)}, \ldots, x_u^{(k)})$ and $\boldsymbol{y}_u \triangleq (y_u^{(1)}, y_u^{(2)}, \ldots, y_u^{(k)})$ that belong to different equivalence relation subsets of $\mathcal{A}^k$ but are given the same $\boldsymbol{Z}_u$-label. Let $J \subseteq \{1, 2, \ldots, k\}$ be the index set collecting all indices j such that $x_u^{(j)} \overset{a_3^{(j)}}{\not\equiv} y_u^{(j)}|_u$ Since $\boldsymbol{x}_u$ and $\boldsymbol{y}_u$ belong to different equivalence relation partitions, $J \neq \emptyset$ and for every $j \in J$ there exists, by definition, a $a_v^{(j)} \in \mathcal{A}, v \in \{1, 2\} \setminus u$ such that

$$f\left(x_u^{(j)}, a_v^{(j)}, a_3^{(j)}\right) \neq f\left(y_u^{(j)}, a_v^{(j)}, a_3^{(j)}\right).$$

Then consider the following two different scenarios of k independent messages.

$$\text{(I)} \quad X_u^k = \boldsymbol{x}_u, X_v^k = \boldsymbol{x}_v, X_3^k = \boldsymbol{a}_3$$

$$\text{(II)} \quad X_u^k = \boldsymbol{y}_u, X_v^k = \boldsymbol{x}_v, X_3^k = \boldsymbol{a}_3,$$

where $\boldsymbol{x}_v$ is such that $x_v^{(j)} = a_v^{(j)}$ for every $j \in J$. Note that the realizations of X_v^k and X_3^k are the same in both cases, and hence so is the label transmitted on the (s_v, t) edge. On the other hand, by assumption we have that the label transmitted on the edge (s_u, t) is the same in both cases as well. Then the terminal cannot recover the correct value of the demand function for the components in the set J, as the $(\boldsymbol{Z}_1, \boldsymbol{Z}_2)$-labels received are the same but the function values are different by choice of $\boldsymbol{x}_u, \boldsymbol{y}_u$. This contradicts the fact that the network code allows t to recover $f(X_1^k, X_2^k, X_3^k)$ with zero error. The two scenarios considered above also give a proof for the second statement of the lemma. ■

APPENDIX C. CALCULATION FOR EQUATION (3.12)

Using equations (3.11), (3.10) and the probabilities computed in illustration 6, we can find the value of α as follows.

$$
\begin{aligned}
&\alpha k \\
&= \sum_{\boldsymbol{b}} \Pr\{f(X_1^k, X_2^k, X_3^k) = \boldsymbol{b}\} \sum_{\boldsymbol{x}_3 \in A_3(\boldsymbol{b})} \Pr\{X_3^k = \boldsymbol{x}_3 | f(X_1^k, X_2^k, X_3^k) = \boldsymbol{b}\} \Big(k \log_{|\mathcal{Z}|} 3 - (t_{0,2} + t_{1,1}) \log_{|\mathcal{Z}|} 3 \\
&\qquad + (1.5 \log_{|\mathcal{Z}|} 2 - \log_{|\mathcal{Z}|} 3)(t_{0,1} + t_{1,2}) \Big) \\
&= \sum_{\boldsymbol{b}} \Pr\{f(X_1^k, X_2^k, X_3^k) = \boldsymbol{b}\} \Bigg[k \log_{|\mathcal{Z}|} 3 \\
&+ \sum_{t_{1,1}=0}^{m_1} \sum_{t_{1,2}=0}^{m_1 - t_{1,1}} \binom{m_1}{t_{1,1}, t_{1,2}, m_1 - t_{1,1} - t_{1,2}} (\tfrac{1}{4})^{m_1 - t_{1,1} - t_{1,2}} (\tfrac{2}{3})^{t_{1,2}} (\tfrac{1}{12})^{t_{1,1}} ((1.5 - \log_{|\mathcal{Z}|} 3) t_{1,2} - t_{1,1} \log_{|\mathcal{Z}|} 3) \\
&\quad \cdot \sum_{t_{0,1}=0}^{k - m_1 - m_2} \sum_{t_{0,2}=0}^{k - m_1 - m_2 - t_{0,1}} \binom{k - m_1 - m_2}{t_{0,1}, t_{0,2}, k - m_1 - m_2 - t_{0,1} - t_{0,2}} (\tfrac{1}{4})^{k - m_1 - m_2 - t_{0,1} - t_{0,2}} (\tfrac{2}{3})^{t_{0,1}} (\tfrac{1}{12})^{t_{0,2}} \\
&+ \sum_{t_{0,1}=0}^{k - m_1 - m_2} \sum_{t_{0,2}=0}^{k - m_1 - m_2 - t_{0,1}} \binom{k - m_1 - m_2}{t_{0,1}, t_{0,2}, m_1 - t_{0,1} - t_{0,2}} (\tfrac{1}{4})^{k - m_1 - m_2 - t_{0,1} - t_{0,2}} (\tfrac{2}{3})^{t_{0,1}} (\tfrac{1}{12})^{t_{0,2}} \\
&\qquad \cdot ((1.5 \log_{|\mathcal{Z}|} 2 - \log_{|\mathcal{Z}|} 3) t_{0,1} - t_{0,2} \log_{|\mathcal{Z}|} 3) \\
&\qquad \cdot \sum_{t_{1,1}=0}^{m_1} \sum_{t_{1,2}=0}^{m_1 - t_{1,1}} \binom{m_1}{t_{1,1}, t_{1,2}, m_1 - t_{1,1} - t_{1,2}} (\tfrac{1}{4})^{m_1 - t_{1,1} - t_{1,2}} (\tfrac{2}{3})^{t_{1,2}} (\tfrac{1}{12})^{t_{1,1}} \Bigg] \\
&= \sum_{\boldsymbol{b}} \Pr\{f(X_1^k, X_2^k, X_3^k) = \boldsymbol{b}\} \left[k \left(\log_{|\mathcal{Z}|} 2 + \frac{\log_{|\mathcal{Z}|} 3}{4} \right) + m_2 \left(- \log_{|\mathcal{Z}|} 2 + \frac{9}{12} \log 3 \right) \right] \\
&= \sum_{m_1=0}^{k} \sum_{m_2=0}^{k - m_1} \binom{k}{m_1, m_2, k - m_1 - m_2} (\tfrac{4}{9})^{k - m_1 - m_2} (\tfrac{4}{9})^{m_1} (\tfrac{1}{9})^{m_2} \left[k \left(\log_{|\mathcal{Z}|} 2 + \frac{\log_{|\mathcal{Z}|} 3}{4} \right) \right. \\
&\qquad \left. + m_2 \left(- \log_{|\mathcal{Z}|} 2 + \frac{9}{12} \log_{|\mathcal{Z}|} 3 \right) \right] \\
&= k \left(\log_{|\mathcal{Z}|} 2 + \frac{\log_{|\mathcal{Z}|} 3}{4} \right)
\end{aligned}
$$

$$+\left(-\log_{|\mathcal{Z}|}2+\frac{9}{12}\log_{|\mathcal{Z}|}3\right)\sum_{m_1=0}^{k}\sum_{m_2=0}^{k-m_1}\binom{k}{m_1,m_2,k-m_1-m_2}m_2(\frac{1}{9})^{m_2}(\frac{4}{9})^{k-m_2}$$

$$=k\left(\log_{|\mathcal{Z}|}2+\frac{\log_{|\mathcal{Z}|}3}{4}\right)+\left(-\log_{|\mathcal{Z}|}2+\frac{9}{12}\log_{|\mathcal{Z}|}3\right)\frac{k}{9}.$$

APPENDIX D. LIST OF PUBLICATIONS FROM DISSERTATION

- A. Tripathy and A. Ramamoorthy, "Sum-Networks From Incidence Structures: Construction and Capacity Analysis," in IEEE Transactions on Information Theory, vol. 64, no. 5, pp. 3461-3480, May 2018. doi: 10.1109/TIT.2017.2765661

- Ardhendu Tripathy and Aditya Ramamoorthy, "On computation rates for arithmetic sum," 2016 IEEE International Symposium on Information Theory (ISIT), Barcelona, 2016, pp. 2354-2358. doi: 10.1109/ISIT.2016.7541720

- Ardhendu Tripathy and Aditya Ramamoorthy, "Capacity of sum-networks for different message alphabets," 2015 IEEE International Symposium on Information Theory (ISIT), Hong Kong, 2015, pp. 606-610. doi: 10.1109/ISIT.2015.7282526

- Ardhendu Tripathy and Aditya Ramamoorthy, "Sum-networks from undirected graphs: Construction and capacity analysis," 2014 52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton), Monticello, IL, 2014, pp. 651-658. doi: 10.1109/ALLERTON.2014.7028517