UNIVERSITY OF CALIFORNIA, SAN DIEGO

Network coding for function computation

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Electrical Engineering (Communication Theory and Systems)

by

Rathinakumar Appuswamy

Committee in charge:

Professor Massimo Franceschetti, Chair
Professor Kenneth Zeger, Co-Chair
Professor Larry Carter
Professor William Hodgkiss
Professor Alex Vardy

2011

Copyright
Rathinakumar Appuswamy, 2011
All rights reserved.

The dissertation of Rathinakumar Appuswamy is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____
Co-Chair

_____
Chair

University of California, San Diego

2011

TABLE OF CONTENTS

Signature Page . . . iii

Table of Contents . . . iv

List of Figures . . . v

List of Tables . . . vi

Acknowledgements . . . vii

Vita . . . viii

Abstract of the Dissertation . . . ix

Chapter 1 Introduction . . . 1

Chapter 2 Network computing: Cut-set bounds . . . 4
- 2.1 Introduction . . . 4
  - 2.1.1 Network model and definitions . . . 5
  - 2.1.2 Classes of target functions . . . 9
  - 2.1.3 Contributions . . . 11
- 2.2 Min-cut upper bound on computing capacity . . . 12
- 2.3 Lower bounds on the computing capacity . . . 12
- 2.4 On the tightness of the min-cut upper bound . . . 19
- 2.5 An example network . . . 23
- 2.6 Conclusions . . . 26
- 2.7 Appendix . . . 27

Chapter 3 Linear Codes, Target Function Classes, and Network Computing Capacity 34
- 3.1 Introduction . . . 34
  - 3.1.1 Contributions . . . 35
- 3.2 Network model and definitions . . . 38
  - 3.2.1 Target functions . . . 38
  - 3.2.2 Network computing and capacity . . . 39
- 3.3 Linear coding over different ring alphabets . . . 43
- 3.4 Linear network codes for computing target functions . . . 46
  - 3.4.1 Non-reducible target functions . . . 46
  - 3.4.2 Reducible target functions . . . 52
- 3.5 Computing linear target functions . . . 56
- 3.6 The reverse butterfly network . . . 61

Chapter 4 Computing linear functions by linear coding over networks . . . 66
- 4.1 Introduction . . . 66
  - 4.1.1 Network model and preliminaries . . . 67
- 4.2 Algebraic framework . . . 69
  - 4.2.1 An algebraic test for the existence of a linear solution . . . 69
  - 4.2.2 Minimum cut condition . . . 71
- 4.3 Computing linear functions . . . 72
- 4.4 Appendix . . . 83

Bibliography . . . 86

LIST OF FIGURES

Figure 2.1: Illustration of equivalence classes 8
Figure 2.2: An example of a multi-edge tree. 9
Figure 2.3: The reverse butterfly Network and a line network 18
Figure 2.4: Computing arithmetic sum in the reverse butterfly network 23

Figure 3.1: Decomposition of the space of all target functions into various classes. . . 37
Figure 3.2: A network illustrating coding over different rings 43
Figure 3.3: A network with s sources to illustrating coding gain 51
Figure 3.4: A network with no coding gain 54
Figure 3.5: The butterfly network and its reverse $\mathcal{N}_3$. 61
Figure 3.6: Computing mod q sum in the reverse butterfly network 63
Figure 3.7: Illustration a solution for computing mod $2q-1$ sum in the reverse butterfly 64

Figure 4.1: Network on which there is no linear solution for computing f_1. 78
Figure 4.2: A network that does not have a linear solution 80
Figure 4.3: A network that computes a given linear function 82

LIST OF TABLES

Table 3.1: Summary of our main results for certain classes of target functions 37
Table 3.2: Definitions of some target functions. 38
Table 3.3: Definition of the 4-ary map f. 43

ACKNOWLEDGEMENTS

I would have not been able to complete this work without help. I would like to thank my advisors Professor Massimo Franceschetti and Professor Ken Zeger for their guidance, patience and persistent efforts to help me become a better writer and researcher, and for his many helpful suggestions and insights. I want to thank Professors Larry Carter, William Hodgkiss, Alexander Vardy, and Jack Wolf for serving on my committee. From my lab, I am indebted to Ehsan Ardestanizadeh, Lorenzo Coviello, Nikhil Karamchandani, and Prof. Paolo Minero for the help they gave me in many different forms, as well as their sympathy and companionship. Also in the Electrical and Computer Engineering department, I would like to thank Mlissa Michelson, Karol Previte, Robert Rome, John Minan, and Bernadette Villaluz for their kind and efficient support of all my efforts at UCSD. In addition, I would like to thank my family. Even from a distance, they were always there for me. I'm grateful for their support, love, and patience.

The text of Chapter 2, in full, is a reprint of the material as it appears in: R. Appuswamy, M. Franceschetti, N. Karamchandani, and K. Zeger, "Network coding for computing: cut-set bounds," *IEEE Transactions on Information Theory*, vol. 57 (2), pp. 1015-1030, Feb. 2011. The text of Chapter 3, in full, has been submitted for publication as: R. Appuswamy, M. Franceschetti, N. Karamchandani, and K. Zeger, "Linear Codes, Target Function Classes, and Network Computing Capacity," submitted to the *IEEE Transactions on Information Theory*, May 2011. The text of Chapter 4, in full, has been submitted for publication as: R.Appuswamy, and M.Franceschetti, "Computing linear functions by linear coding over networks," submitted to the *IEEE Transactions on Information Theory*, Feb. 2011. I was the primary researcher of all the three papers listed above and Nikhil Karamchandani, Professors Massimo Franceschetti, and Ken Zeger who are listed as co-authors have contributed with insights and helpful discussions.

VITA

2002	B. Tech. in Electronics, Anna University
2004	M. Tech. in Electrical Engineering, Indian Institute of Technology, Kanpur
2010	M. A. in Mathematics, University of California, San Diego
2011	Ph. D. in Electrical Engineering (Communication Theory and Sysmtems), University of California, San Diego

PUBLICATIONS

R. Appuswamy and A. K. Chaturvedi, "Mutually orthogonal sets of ZCZ sequences," *Electronics Letters*, vol. 40(18), pp. 1133-1134, Sept. 2004.

R. Appuswamy and A. K. Chaturvedi,"A New Framework for Constructing ZCZ and Complementary Sequences with Applications," *IEEE Transactions on Information Theory*, vol. 52(8), pp. 3817-3826, Aug. 2006.

Q. Qu, R. Appuswamy, and Y.S. Chan, "QoS guarantee and provisioning for realtime digital video over mobile ad hoc cdma networks with cross-layer design," *IEEE Wireless Communications*, pp. 82-88, Oct. 2006.

R. Appuswamy and A. K. Chaturvedi, "Complete Mutually Orthogonal Golay Complementary Sets From Reed- Muller Codes," *IEEE Transactions on Information Theory*, vol. 54(3), pp. 1339-1346, March 2008.

R. Appuswamy, M. Franceschetti, N. Karamchandani, and K. Zeger, "Network coding for computing: cut-set bounds," *IEEE Transactions on Information Theory*, vol. 57 (2), pp. 1015-1030, Feb. 2011.

N.Karamchandani, R.Appuswamy, and M.Franceschetti, "Distributed Computation of Symmetric Functions with Binary Inputs," submitted to the *IEEE Transactions on Information Theory*, Sep. 2009.

R.Appuswamy, and M.Franceschetti, "Computing linear functions by linear coding over networks," submitted to the *IEEE Transactions on Information Theory*, Feb. 2011.

R. Appuswamy, M. Franceschetti, N. Karamchandani, and K. Zeger, "Linear Codes, Target Function Classes, and Network Computing Capacity," submitted to the *IEEE Transactions on Information Theory*, May 2011.

ABSTRACT OF THE DISSERTATION

Network coding for function computation

by

Rathinakumar Appuswamy

Doctor of Philosophy in Electrical Engineering (Communication Theory and Systems)

University of California, San Diego, 2011

Professor Massimo Franceschetti, Chair
Professor Kenneth Zeger, Co-Chair

In this dissertation, the following *network computing* problem is considered. Source nodes in a directed acyclic network generate independent messages and a single receiver node computes a target function f of the messages. The objective is to maximize the average number of times f can be computed per network usage, i.e., the "computing capacity". The *network coding* problem for a single-receiver network is a special case of the network computing problem in which all of the source messages must be reproduced at the receiver. For network coding with a single receiver, routing is known to achieve the capacity by achieving the network *min-cut* upper bound. First we extend the definition of min-cut to the network computing problem and show that the generalized min-cut is an upper bound on the maximum achievable rate and is tight for computing (using coding) any target function in multi-edge tree networks and for computing linear target functions in any network. We also study the bound's tightness for different classes of target functions. In particular, we give a lower bound on the computing capacity in terms of the Steiner tree packing number and a different bound for symmetric functions. We also show that for certain networks and target functions, the computing capacity can be less than an arbitrarily small fraction of the min-cut bound.

Next, we study the use of linear codes for network computing in single-receiver networks with various classes of target functions of the source messages. Such classes include reducible, injective, semi-injective, and linear target functions over finite fields. Computing capacity bounds are given with respect to these target function classes for network codes that use routing, linear coding, or nonlinear coding.

Lastly, we consider the scenario in which a set of sources generate messages in a network over a finite field alphabet and a receiver node demands an arbitrary *linear function* of these messages. We formulate an algebraic test to determine whether an arbitrary network can compute linear functions using *linear codes*. We identify a class of linear functions that can be computed using linear codes in every network that satisfies a natural cut-based condition. Conversely, for another class of linear functions, we show that the cut-based condition does not guarantee the existence of a linear coding solution. For linear functions over the binary field, the two classes are complements of each other.

Chapter 1

Introduction

With great progress in electronics technology, some systems including automobiles, sensors, and portable personal devices are radio enabled and communicate and cooperate with other devices. In such networks, receivers may need to compute a *target function* of the messages generated by the sources rather than obtain the messages themselves. Examples of such *target functions* that can arise in applications include average, histogram, maximum, and minimum, etc. In this setting, we use the term 'network computing' to refer to problems where a subset of nodes in a network generate messages and a receiver node wants to compute a given function of these messages. My dissertation focuses on investigating the information theoretic limits of computing in networks and on designing codes to achieve such limits.

Network computing in its most general form includes the problem of communicating possibly correlated messages to a specific destination, at a desired fidelity with respect to a joint distortion criterion dependent on the given target function. The overwhelming complexity of this general problem suggests that simplifications be examined in order to obtain some understanding of the field.

As part of my dissertation, I propose a natural model of network computing that is closely related to the *network coding* model of Ahlswede, Cai, Li, and Yeung [Ahlswede 00]. Unlike routing, network coding allows every network node to send an arbitrary function of its received symbols on its out-edges. A *network code* provides an assignment of such functions for each of the network edges. It is known [Ahlswede 00, Harvey 06, Ngai 04] that network coding provides some advantages over routing including increased throughput, robustness against link failures, etc. Network coding considers networks with multiple sources and multiple receivers and studies the special case when each of the receivers wants a subset of the source messages. In contrast, in network computing, we consider the setting where source nodes generate independent messages and a single receiver node computes a target function f of these messages. The objective of the receiver node is to compute f as often as possible. We introduce the notion

of *achievable computing rate* which captures how often the target function is computed by the receiver by a given network code and define the *computing capacity* to be the supremum of all the achievable computing rates over all possible codes. Similarly, we define *linear computing capacity* and *routing computing capacity* by restricting the node operations to be linear and routing, respectively.

The notion of *min-cut* was studied in graph theory by Menger in 1927 [Menger 27] and it was later incorporated in to the study of single-source single-receiver networks by Ford and Fulkerson [Ford 56]. This idea was later generalized to multiple-source multiple-receiver networks by Hu [Hu 63] and Leighton and Rao [Leighton 99].The min-cut is also referred to as "sparsity" by some authors, such as Harvey, Kleinberg, and Lehman [Harvey 06] and Vazirani [Vazirani 04]. In Chapter 2, we define a generalized min-cut to the network computing problem and show that this *new min-cut* is an upper bound on the computing capacity for arbitrary target functions and networks. In addition, a lower bound on the computing capacity is given in terms of the Steiner tree packing number for any network and any target function. Our next objective is to understand if and when the min-cut upper bound is achievable i.e., given a network and a target function, does there exist a network code that achieves a computing rate that is close to the min-cut? Our approach toward answering this question is twofold. First, we restrict the set of allowed networks and show that within this restricted set of networks, the min-cut is equal to the computing capacity for any target function. Next, by grouping target functions into *divisible, symmetric,* λ*-exponential*, and λ*-bounded* functions, we present several coding schemes for computing these groups of functions and show that the min-cut upper bound and lower bounds are within a constant factor of each other for λ-exponential and λ-bounded target functions. On the other hand, we also show that if the target function is the *arithmetic sum*, then the gap between the computing capacity and the min-cut may grow with the number of source nodes [Appuswamy 11c].

Linear codes are of fundamental importance in communication systems since they are easy to design and exhibit low encoding and decoding complexity. For network computing, it is of interest to know if linear codes are sufficient to achieve the computing capacity, or how much capacity gain they may provide over routing. In Chapter 3, we investigate the performance of linear network codes for computing different types of target functions. We compare the linear computing capacity with the (non-linear) computing capacity and the routing computing capacity for different classes of target functions. Our results show that whether or not linear network codes provide a capacity advantage depends on the target function computed at the receiver. It is also shown that non-linear codes are necessary in general to achieve the computing capacity and that for computing scalar linear functions over finite fields, linear codes are capacity achieving. One important contribution of our research is that we provide a characterization of the class of target functions for which the use of linear codes may provide a computing capacity advantage over routing [Appuswamy 11b].

Finally, we consider the following problem: Given a network, an arbitrary target function f, and a rational number r, does there exists a linear code that computes f and achieves a computing rate of r? There are, in general, infinitely many linear codes that need to be examined before we can answer this question. In Chapter 4, we formulate a more 'efficient' algebraic test to answer this question by restricting the set of allowable target functions to be linear. We further show that for a small subclass of linear target functions, a linear code that computes the target function exists whenever the computing rate r is less than the corresponding min-cut defined in Chapter 2. Conversely, we also show that for another large class of linear target functions, there always exist a network with min-cut larger than r but the network does not have a linear code that achieves a computing rate of r [Appuswamy 11a].

Each of the three Chapters 2-4 in this dissertation is a copy of a published or submitted co-authored journal paper. These are as follows:

Chapter 2	R. Appuswamy, M. Franceschetti, N. Karamchandani, and K. Zeger, "Network coding for computing: cut-set bounds," *IEEE Transactions on Information Theory*, vol. 57 (2), pp. 1015-1030, Feb. 2011.
Chapter 3	R. Appuswamy, M. Franceschetti, N. Karamchandani, and K. Zeger, "Linear Codes, Target Function Classes, and Network Computing Capacity," submitted to the *IEEE Transactions on Information Theory*, May 2011.
Chapter 4	R.Appuswamy, and M.Franceschetti, "Computing linear functions by linear coding over networks," submitted to the *IEEE Transactions on Information Theory*, Feb. 2011.

Chapter 2

Network computing: Cut-set bounds

2.1 Introduction

We consider networks where source nodes generate independent messages and a single receiver node computes a target function f of these messages. The objective is to characterize the maximum rate of computation, that is the maximum number of times f can be computed per network usage.

Giridhar and Kumar [Giridhar 05] have recently stated:

> "In its most general form, computing a function in a network involves communicating possibly correlated messages, to a specific destination, at a desired fidelity with respect to a joint distortion criterion dependent on the given function of interest. This combines the complexity of source coding of correlated sources, with rate distortion, different possible network collaborative strategies for computing and communication, and the inapplicability of the separation theorem demarcating source and channel coding."

The overwhelming complexity of network computing suggests that simplifications be examined in order to obtain some understanding of the field.

We present a natural model of network computing that is closely related to the network coding model of Ahlswede, Cai, Li, and Yeung [Ahlswede 00, Yeung 02]. Network coding is a widely studied communication mechanism in the context of network information theory. In network coding, some nodes in the network are labeled as sources and some as receivers. Each receiver needs to reproduce a subset of the messages generated by the source nodes, and all nodes can act as relays and encode the information they receive on in-edges, together with the information they generate if they are sources, into codewords which are sent on their out-edges. In existing computer networks, the encoding operations are purely routing: at each node, the

codeword sent over an out-edge consists of a symbol either received by the node, or generated by it if is a source. It is known that allowing more complex encoding than routing can in general be advantageous in terms of communication rate [Ahlswede 00, Harvey 06, Ngai 04]. Network coding with a single receiver is equivalent to a special case of our function computing problem, namely when the function to be computed is the identity, that is when the receiver wants to reproduce all the messages generated by the sources. In this paper, we study network computation for target functions different than the identity.

Some other approaches to network computation have also appeared in the literature. In [Körner 79, Orlitsky 01, Doshi 06, Doshi 07b, Ma 08, Cuff 09] network computing was considered as an extension of distributed source coding, allowing the sources to have a joint distribution and requiring that a function be computed with small error probability. A rate-distortion approach to the problem has been studied in [Yamamoto 82, Feng 04, Doshi 07a]. However, the complexity of network computing has restricted prior work to the analysis of elementary networks. Networks with noisy links were studied in [Gamal 87, Gallager 88, Ying 07, Goyal 08, Dutta 08, Karamchandani 09, Ayaso 07, Nazer 07, Ma 09] and distributed computation in networks using gossip algorithms was studied in [Kempe 03, Boyd 06, Mosk-Aoyama 08, Ayaso 08, Dimakis 06, Benezit 07].

In the present paper, our approach is somewhat (tangentially) related to the field of communication complexity [Kushilevitz 97, Yao 79] which studies the minimum number of messages that two nodes need to exchange in order to compute a function of their inputs with zero error. Other studies of computing in networks have been considered in [Giridhar 05, Subramanian 07], but these were restricted to the wireless communication protocol model of Gupta and Kumar [Gupta 00].

In contrast, our approach is more closely associated with wired networks with independent noiseless links. Our work is closest in spirit to the recent work of [Ramamoorthy 08, Rai 10b, Rai 09] on computing the sum (over a finite field) of source messages in networks. We note that in independent work, Kowshik and Kumar[Kowshik 09] obtain the asymptotic maximum rate of computation in tree networks and present bounds for computation in networks where all nodes are sources.

Our main contributions are summarized in Section 2.1.3, after formally introducing the network model.

2.1.1 Network model and definitions

In this paper, a *network* $\mathcal{N}$ consists of a finite, directed acyclic multigraph $G = (\mathcal{V}, \mathcal{E})$, a set of *source nodes* $S = \{\sigma_1, \ldots, \sigma_s\} \subseteq \mathcal{V}$, and a *receiver* $\rho \in \mathcal{V}$. Such a network is denoted by $\mathcal{N} = (G, S, \rho)$. We will assume that $\rho \notin S$ and that the graph[1] G contains a directed path

[1]Throughout the paper, we will use "graph" to mean a directed acyclic multigraph, and "network" to mean a single-receiver network. We may sometimes write $\mathcal{E}(G)$ to denote the edges of graph G.

from every node in $\mathcal{V}$ to the receiver ρ. For each node $u \in \mathcal{V}$, let $\mathcal{E}_i(u)$ and $\mathcal{E}_o(u)$ denote the set of in-edges and out-edges of u respectively. We will also assume (without loss of generality) that if a network node has no in-edges, then it is a source node.

An *alphabet* $\mathcal{A}$ is a finite set of size at least two. For any positive integer m, any vector $x \in \mathcal{A}^m$, and any $i \in \{1, 2, \ldots, m\}$, let x_i denote the i-th component of x. For any index set $I = \{i_1, i_2, \ldots, i_q\} \subseteq \{1, 2, \ldots, m\}$ with $i_1 < i_2 < \ldots < i_q$, let x_I denote the vector $(x_{i_1}, x_{i_2}, \ldots, x_{i_q}) \in \mathcal{A}^{|I|}$.

The *network computing* problem consists of a network $\mathcal{N}$ and a *target function* f of the form

$$f : \mathcal{A}^s \longrightarrow \mathcal{B}$$

(see Definition 2.1.4 for some examples). We will also assume that any target function depends on all network sources (i.e. they cannot be constant functions of any one of their arguments). Let k and n be positive integers. Given a network $\mathcal{N}$ with source set S and alphabet $\mathcal{A}$, a *message generator* is any mapping

$$\alpha \ : \ S \longrightarrow \mathcal{A}^k.$$

For each source σ_i, $\alpha(\sigma_i)$ is called a *message vector* and its components $\alpha(\sigma_i)_1, \ldots, \alpha(\sigma_i)_k$ are called *messages*.[2]

Definition 2.1.1. A (k, n) *network code* for computing a target function f in a network $\mathcal{N}$ consists of the following:

(i) For any node $v \in \mathcal{V} - \rho$ and any out-edge $e \in \mathcal{E}_o(v)$, an *encoding function*:

$$h^{(e)} : \begin{cases} \left(\prod_{\hat{e} \in \mathcal{E}_i(v)} \mathcal{A}^n \right) \times \mathcal{A}^k \longrightarrow \mathcal{A}^n & \text{if } v \text{ is a source node} \\ \prod_{\hat{e} \in \mathcal{E}_i(v)} \mathcal{A}^n \longrightarrow \mathcal{A}^n & \text{otherwise} \end{cases}$$

(ii) A *decoding function*:

$$\psi : \prod_{j=1}^{|\mathcal{E}_i(\rho)|} \mathcal{A}^n \longrightarrow \mathcal{B}^k.$$

Given a (k, n) network code, every edge $e \in \mathcal{E}$ *carries a vector* z_e of at most n alphabet symbols,[3] which is obtained by evaluating the encoding function $h^{(e)}$ on the set of vectors carried by the in-edges to the node and the node's message vector if it is a source. The objective of the receiver is to compute the target function f of the source messages, for any arbitrary message generator α. More precisely, the receiver constructs a vector of k alphabet symbols such that for each $i \in \{1, 2, \ldots, k\}$, the i-th component of the receiver's computed vector equals the value

[2] For simplicity, we assume that each source has exactly one message vector associated with it, but all of the results in this paper can readily be extended to the more general case.

[3] By default, we will assume that edges carry exactly n symbols.

of the desired target function f applied to the i-th components of the source message vectors, for any choice of message generator α. Let $e_1, e_2, \ldots, e_{|\mathcal{E}_i(\rho)|}$ denote the in-edges of the receiver.

Definition 2.1.2. A (k, n) network code is called *a solution for computing f in $\mathcal{N}$* (or simply *a (k, n) solution*) if the decoding function ψ is such that for each $j \in \{1, 2, \ldots, k\}$ and for every message generator α, we have

$$\psi\left(z_{e_1}, \cdots, z_{e_{|\mathcal{E}_i(\rho)|}}\right)_j = f\left(\alpha(\sigma_1)_j, \cdots, \alpha(\sigma_s)_j\right). \tag{2.1}$$

If there exists a (k, n) solution, we say the rational number k/n is an *achievable computing rate*.

In the network coding literature, one definition of the *coding capacity* of a network is the supremum of all achievable coding rates [Cannons 06, Dougherty 06]. We adopt an analogous definition for computing capacity.

Definition 2.1.3. The *computing capacity* of a network $\mathcal{N}$ with respect to target function f is

$$\mathcal{C}_{\text{cod}}(\mathcal{N}, f) \;=\; \sup\left\{\frac{k}{n} \;:\; \exists\ (k, n) \text{ network code for computing } f \text{ in } \mathcal{N}\right\}.$$

Thus, the computing capacity is the supremum of all achievable computing rates for a given network $\mathcal{N}$ and a target function f. Some example target functions are defined below.

Definition 2.1.4.

Target function f	Alphabet $\mathcal{A}$	$f(x_1, \ldots, x_s)$	Comments
identity	arbitrary	$(x_1, \ldots, x_s)$	
arithmetic sum	$\{0, 1, \ldots, q-1\}$	$x_1 + x_2 + \cdots + x_s$	'+' is integer addition
mod r sum	$\{0, 1, \ldots, q-1\}$	$x_1 \oplus x_2 \oplus \ldots \oplus x_s$	$\oplus$ is mod r addition
histogram	$\{0, 1, \ldots, q-1\}$	$(c_0, c_1, \ldots, c_{q-1})$	$c_i = \lvert\{j : x_j = i\}\rvert$
linear	any finite field	$a_1x_1 + a_2x_2 + \ldots + a_sx_s$	arithmetic over the field
maximum	any ordered set	$\max\{x_1, \ldots, x_s\}$	

Definition 2.1.5. For any target function $f : \mathcal{A}^s \longrightarrow \mathcal{B}$, any index set $I \subseteq \{1, 2, \ldots, s\}$, and any $a, b \in \mathcal{A}^{|I|}$, we write $a \equiv b$ if for every $x, y \in \mathcal{A}^s$, we have $f(x) = f(y)$ whenever $x_I = a$, $y_I = b$, and $x_j = y_j$ for all $j \notin I$.

It can be verified that $\equiv$ is an equivalence relation[4] for every f and I.

[4] Witsenhausen [Witsenhausen 76] represented this equivalence relation in terms of the independent sets of a characteristic graph and his representation has been used in various problems related to function computation [Doshi 06, Doshi 07b, Orlitsky 01]. Although $\equiv$ is defined with respect to a particular index set I and a function f, we do not make this dependence explicit – the values of I and f will be clear from the context.

Definition 2.1.6. For every f and I, let $R_{I,f}$ denote the total number of equivalence classes induced by $\equiv$ and let

$$\Phi_{I,f} : \mathcal{A}^{|I|} \longrightarrow \{1, 2, \ldots, R_{I,f}\}$$

be any function such that $\Phi_{I,f}(a) = \Phi_{I,f}(b)$ iff $a \equiv b$.

That is, $\Phi_{I,f}$ assigns a unique index to each equivalence class, and

$$R_{I,f} = \left| \left\{ \Phi_{I,f}(a) : a \in \mathcal{A}^{|I|} \right\} \right|.$$

The value of $R_{I,f}$ is independent of the choice of $\Phi_{I,f}$. We call $R_{I,f}$ the *footprint size* of f with respect to I.

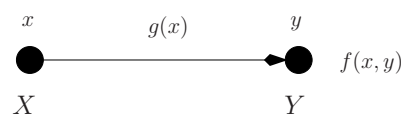


Figure 2.1: X, Y are two sources with messages x and y respectively. X communicates $g(x)$ to Y so that Y can compute a function f of x and y.

Remark 2.1.7. Let $I^c = \{1, 2, \ldots, s\} - I$. The footprint size $R_{I,f}$ has the following interpretation (see Figure 2.1). Suppose a network has two nodes, X and Y, and both are sources. A single directed edge connects X to Y. Let X generate $x \in \mathcal{A}^{|I|}$ and Y generate $y \in \mathcal{A}^{|I^c|}$. X communicates a function $g(x)$ of its input, to Y so that Y can compute $f(a)$ where $a \in \mathcal{A}^s$, $a_I = x$, and $a_{I^c} = y$. Then for any $x, \hat{x} \in \mathcal{A}^{|I|}$ such that $x \not\equiv \hat{x}$, we need $g(x) \neq g(\hat{x})$. Thus $\left| g\left(\mathcal{A}^{|I|}\right) \right| \geq R_{I,f}$, which implies a lower bound on a certain amount of "information" that X needs to send to Y to ensure that it can compute the function f. Note that $g = \Phi_{I,f}$ achieves the lower bound. We will use this intuition to establish a cut-based upper bound on the computing capacity $\mathcal{C}_{\text{cod}}(\mathcal{N}, f)$ of any network $\mathcal{N}$ with respect to any target function f, and to devise a capacity-achieving scheme for computing any target function in multi-edge tree networks.

Definition 2.1.8. A set of edges $C \subseteq \mathcal{E}$ in network $\mathcal{N}$ is said to *separate* sources $\sigma_{m_1}, \ldots, \sigma_{m_d}$ from the receiver ρ, if for each $i \in \{1, 2, \ldots, d\}$, every directed path from σ_{m_i} to ρ contains at least one edge in C. The set C is said to be a *cut* in $\mathcal{N}$ if it separates at least one source from the receiver. For any network $\mathcal{N}$, define $\Lambda(\mathcal{N})$ to be the collection of all cuts in $\mathcal{N}$. For any cut $C \in \Lambda(\mathcal{N})$ and any target function f, define

$$\begin{aligned} I_C &= \{i : C \text{ separates } \sigma_i \text{ from the receiver}\} \\ R_{C,f} &= R_{I_C,f}. \end{aligned} \tag{2.2}$$

Since target functions depend on all sources, we have $R_{C,f} \geq 2$ for any cut C and any target function f. The footprint sizes $R_{C,f}$ for some example target functions are computed below.

A *multi-edge tree* is a graph such that for every node $v \in \mathcal{V}$, there exists a node u such that all the out-edges of v are in-edges to u, i.e., $\mathcal{E}_o(v) \subseteq \mathcal{E}_i(u)$ (e.g. see Figure 2.2).

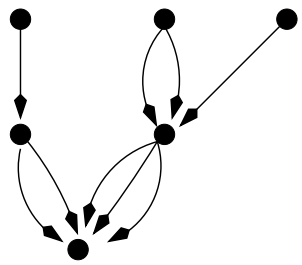


Figure 2.2: An example of a multi-edge tree.

2.1.2 Classes of target functions

We study the following four classes of target functions: (1) divisible, (2) symmetric, (3) λ-exponential, (4) λ-bounded.

Definition 2.1.9. A target function $f : \mathcal{A}^s \longrightarrow \mathcal{B}$ is *divisible* if for every index set $I \subseteq \{1, \ldots, s\}$, there exists a finite set $\mathcal{B}_I$ and a function $f^I : \mathcal{A}^{|I|} \longrightarrow \mathcal{B}_I$ such that the following hold:

(1) $f^{\{1,\ldots,s\}} = f$

(2) $\left|f^I\left(\mathcal{A}^{|I|}\right)\right| \leq |f\left(\mathcal{A}^s\right)|$

(3) For every partition $\{I_1, \ldots, I_\gamma\}$ of I, there exists a function $g : \mathcal{B}_{I_1} \times \cdots \times \mathcal{B}_{I_\gamma} \longrightarrow \mathcal{B}_I$ such that for every $x \in \mathcal{A}^{|I|}$, we have $f^I(x) = g\left(f^{I_1}(x_{I_1}), \ldots, f^{I_\gamma}\left(x_{I_\gamma}\right)\right)$.

Examples of divisible target functions include the identity, maximum, mod r sum, and arithmetic sum.

Divisible functions have been studied previously[5] by Giridhar and Kumar [Giridhar 05] and Subramanian, Gupta, and Shakkottai [Subramanian 07]. Divisible target functions can be computed in networks in a divide-and-conquer fashion as follows. For any arbitrary partition $\{I_1, \ldots, I_\gamma\}$ of the source indices $\{1, \ldots, s\}$, the receiver ρ can evaluate the target function f by combining evaluations of $f^{I_1}, \ldots, f^{I_\gamma}$. Furthermore, for every $i = 1, \ldots, \gamma$, the target function f^{I_i} can be evaluated similarly by partitioning I_i and this process can be repeated until the function value is obtained.

[5]The definitions in [Giridhar 05, Subramanian 07] are similar to ours but slightly more restrictive.

Definition 2.1.10. A target function $f : \mathcal{A}^s \longrightarrow \mathcal{B}$ is *symmetric* if for any permutation π of $\{1, 2, \ldots, s\}$ and any vector $x \in \mathcal{A}^s$,

$$f(x_1, x_2, \ldots, x_s) = f(x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(s)}).$$

That is, the value of a symmetric target function is invariant with respect to the order of its arguments and hence, it suffices to evaluate the histogram target function for computing any symmetric target function. Examples of symmetric functions include the arithmetic sum, maximum, and mod r sum. Symmetric functions have been studied in the context of computing in networks by Giridhar and Kumar [Giridhar 05], Subramanian, Gupta, and Shakkottai [Subramanian 07], Ying, Srikant, and Dullerud [Ying 07], and [Karamchandani 09].

Definition 2.1.11. Let $\lambda \in (0, 1]$. A target function $f : \mathcal{A}^s \longrightarrow \mathcal{B}$ is said to be λ-*exponential* if its footprint size satisfies

$$R_{I,f} \geq |\mathcal{A}|^{\lambda|I|} \text{ for every } I \subseteq \{1, 2, \ldots, s\}.$$

Let $\lambda \in (0, \infty)$. A target function $f : \mathcal{A}^s \longrightarrow \mathcal{B}$ is said to be λ-*bounded* if its footprint size satisfies

$$R_{I,f} \leq |\mathcal{A}|^{\lambda} \text{ for every } I \subseteq \{1, 2, \ldots, s\}.$$

Example 2.1.12. The following facts are easy to verify:

- The identity function is 1-exponential.
- Let $\mathcal{A}$ be an ordered set. The maximum (or minimum) function is 1-bounded.
- Let $\mathcal{A} = \{0, 1, \ldots, q-1\}$ where $q \geq 2$. The mod r sum target function with $q \geq r \geq 2$ is $\log_q r$-bounded.

Remark 2.1.13. Giridhar and Kumar [Giridhar 05] defined two classes of functions: *type-threshold* and *type-sensitive* functions. Both are sub-classes of symmetric functions. In addition, type-threshold functions are also divisible and c-bounded, for some constant c that is independent of the network size. However, [Giridhar 05] uses a model of interference for simultaneous transmissions and their results do not directly compare with ours.

Following the notation in Leighton and Rao [Leighton 99], the *min-cut* of any network $\mathcal{N}$ with unit-capacity edges is

$$\text{min-cut}(\mathcal{N}) = \min_{C \in \Lambda(\mathcal{N})} \frac{|C|}{|I_C|}. \tag{2.3}$$

A more general version of the network min-cut plays a fundamental role in the field of multi-commodity flow [Leighton 99, Vazirani 04]. The min-cut provides an upper bound on the maximum flow for any multi-commodity flow problem. The min-cut is also referred to as "sparsity" by some authors, such as Harvey, Kleinberg, and Lehman [Harvey 06] and Vazirani [Vazirani 04]. We next generalize the definition in (2.3) to the network computing problem.

Definition 2.1.14. If $\mathcal{N}$ is a network and f is a target function, then define

$$\text{min-cut}(\mathcal{N}, f) = \min_{C \in \Lambda(\mathcal{N})} \frac{|C|}{\log_{|\mathcal{A}|} R_{C,f}}. \tag{2.4}$$

Example 2.1.15.

- If f is the identity target function, then

$$\text{min-cut}(\mathcal{N}, f) = \min_{C \in \Lambda(\mathcal{N})} \frac{|C|}{|I_C|}.$$

  Thus for the identity function, the definition of min-cut in (2.3) and (2.4) coincide.

- Let $\mathcal{A} = \{0, 1, \ldots, q-1\}$. If f is the arithmetic sum target function, then

$$\text{min-cut}(\mathcal{N}, f) = \min_{C \in \Lambda(\mathcal{N})} \frac{|C|}{\log_q \left((q-1)\,|I_C| + 1\right)}. \tag{2.5}$$

- Let $\mathcal{A}$ be an ordered set. If f is the maximum target function, then

$$\text{min-cut}(\mathcal{N}, f) = \min_{C \in \Lambda(\mathcal{N})} |C|\,.$$

2.1.3 Contributions

The main results of this paper are as follows. In Section 2.2, we show (Theorem 2.2.1) that for any network $\mathcal{N}$ and any target function f, the quantity min-cut$(\mathcal{N}, f)$ is an upper bound on the computing capacity $\mathcal{C}_{\text{cod}}(\mathcal{N}, f)$. In Section 2.3, we note that the computing capacity for any network with respect to the identity target function is equal to the min-cut upper bound (Theorem 2.3.1). We show that the min-cut bound on computing capacity can also be achieved for all networks with linear target functions over finite fields (Theorem 2.3.2) and for all multi-edge tree networks with any target function (Theorem 2.3.3). For any network and any target function, a lower bound on the computing capacity is given in terms of the Steiner tree packing number (Theorem 2.3.5). Another lower bound is given for networks with symmetric target functions (Theorem 2.3.7). In Section 2.4, the tightness of the above-mentioned bounds is analyzed for divisible (Theorem 2.4.2), symmetric (Theorem 2.4.3), λ-exponential (Theorem 2.4.4), and λ-bounded (Theorem 2.4.5) target functions. For λ-exponential target functions, the computing capacity is at least λ times the min-cut. If every non-receiver node in a network is a source, then for λ-bounded target functions the computing capacity is at least a constant times the min-cut divided by λ. It is also shown, with an example target function, that there are networks for which the computing capacity is less than an arbitrarily small fraction of the min-cut bound (Theorem 2.4.7). In Section 2.5, we discuss an example network and target function in detail to illustrate the above bounds. In Section 2.6, conclusions are given and various lemmas are proven in the Appendix.

2.2 Min-cut upper bound on computing capacity

The following shows that the maximum rate of computing a target function f in a network $\mathcal{N}$ is at most min-cut$(\mathcal{N}, f)$.

Theorem 2.2.1. *If $\mathcal{N}$ is a network with target function f, then*

$$\mathcal{C}_{cod}(\mathcal{N}, f) \leq \text{min-cut}(\mathcal{N}, f).$$

Proof. Let the network alphabet be $\mathcal{A}$ and consider any (k, n) solution for computing f in $\mathcal{N}$. Let C be a cut and for each $i \in \{1, 2, \ldots, k\}$, let $a^{(i)}, b^{(i)} \in \mathcal{A}^{|I_C|}$. Suppose $j \in \{1, 2, \ldots, k\}$ is such that $a^{(j)} \not\equiv b^{(j)}$, where $\equiv$ is the equivalence relation from Definition 2.1.5. Then there exist $x, y \in \mathcal{A}^s$ satsifying: $f(x) \neq f(y)$, $x_{I_C} = a^{(j)}$, $y_{I_C} = b^{(j)}$, and $x_i = y_i$ for every $i \notin I_C$.

The receiver ρ can compute the target function f only if, for every such pair $\left\{a^{(1)}, \ldots, a^{(k)}\right\}$ and $\left\{b^{(1)}, \ldots, b^{(k)}\right\}$ corresponding to the message vectors generated by the sources in I_C, the edges in cut C carry distinct vectors. Since the total number of equivalence classes for the relation $\equiv$ equals the footprint size $R_{C,f}$, the edges in cut C should carry at least $(R_{C,f})^k$ distinct vectors. Thus, we have

$$\mathcal{A}^{n|C|} \geq (R_{C,f})^k$$

and hence for any cut C,

$$\frac{k}{n} \leq \frac{|C|}{\log_{|\mathcal{A}|} R_{C,f}}.$$

Since the cut C is arbitrary, the result follows from Definition 2.1.3 and (2.4). □

The min-cut upper bound has the following intuition. Given any cut $C \in \Lambda(\mathcal{N})$, at least $\log_{|\mathcal{A}|} R_{C,f}$ units of information need to be sent across the cut to successfully compute a target function f. In subsequent sections, we study the tightness of this bound for different classes of functions and networks.

2.3 Lower bounds on the computing capacity

The following result shows that the computing capacity of any network $\mathcal{N}$ with respect to the identity target function equals the coding capacity for ordinary network coding.

Theorem 2.3.1. *If $\mathcal{N}$ is a network with the identity target function f, then*

$$\mathcal{C}_{cod}(\mathcal{N}, f) = \text{min-cut}(\mathcal{N}, f) = \text{min-cut}(\mathcal{N}).$$

Proof. Rasala Lehman and Lehman [Lehman 03, p.6, Theorem 4.2] showed that for any single-receiver network, the conventional coding capacity (when the receiver demands the messages generated by all the sources) always equals the min-cut$(\mathcal{N})$. Since the target function f is the identity, the computing capacity is the coding capacity and min-cut$(\mathcal{N}, f)$ = min-cut$(\mathcal{N})$, so the result follows. □

Theorem 2.3.2. *If $\mathcal{N}$ is a network with a finite field alphabet and with a linear target function f, then*

$$\mathcal{C}_{cod}(\mathcal{N}, f) = \text{min-cut}(\mathcal{N}, f).$$

Proof. Follows from [Rai 10b, Theorem 2]. □

Theorems 2.3.1 and 2.3.2 demonstrate the achievability of the min-cut bound for arbitrary networks with particular target functions. In contrast, the following result demonstrates the achievability of the min-cut bound for arbitrary target functions and a particular class of networks. The following theorem concerns multi-edge tree networks, which were defined in Section 2.1.1.

Theorem 2.3.3. *If $\mathcal{N}$ is a multi-edge tree network with target function f, then*

$$\mathcal{C}_{cod}(\mathcal{N}, f) = \text{min-cut}(\mathcal{N}, f).$$

Proof. Let $\mathcal{A}$ be the network alphabet. From Theorem 2.2.1, it suffices to show that $\mathcal{C}_{\text{cod}}(\mathcal{N}, f) \geq \text{min-cut}(\mathcal{N}, f)$. Since $\mathcal{E}_o(v)$ is a cut for node $v \in \mathcal{V} - \rho$, and using (2.2), we have

$$\text{min-cut}(\mathcal{N}, f) \leq \min_{v \in \mathcal{V}-\rho} \frac{|\mathcal{E}_o(v)|}{\log_{|\mathcal{A}|} R_{\mathcal{E}_o(v),f}}. \tag{2.6}$$

Consider any positive integers k, n such that

$$\frac{k}{n} \leq \min_{v \in \mathcal{V}-\rho} \frac{|\mathcal{E}_o(v)|}{\log_{|\mathcal{A}|} R_{I_{\mathcal{E}_o(v)},f}}. \tag{2.7}$$

Then we have

$$|\mathcal{A}|^{|\mathcal{E}_o(v)|n} \geq R^k_{I_{\mathcal{E}_o(v)},f} \quad \text{for every node } v \in \mathcal{V} - \rho. \tag{2.8}$$

We outline a (k, n) solution for computing f in the multi-edge tree network $\mathcal{N}$. Each source $\sigma_i \in S$ generates a message vector $\alpha(\sigma_i) \in \mathcal{A}^k$. Denote the vector of i-th components of the source messages by

$$x^{(i)} = (\alpha(\sigma_1)_i, \cdots, \alpha(\sigma_s)_i).$$

Every node $v \in \mathcal{V} - \{\rho\}$ sends out a unique index (as guaranteed by (2.8)) over $A^{|\mathcal{E}_o(v)|n}$ corresponding to the set of equivalence classes

$$\Phi_{I_{\mathcal{E}_o(v)},f}(x^{(l)}_{I_{\mathcal{E}_o(v)}}) \quad \text{for} \quad l \in \{1, \cdots, k\}. \tag{2.9}$$

If v has no in-edges, then by assumption, it is a source node, say σ_j. The set of equivalence classes in (2.9) is a function of its own messages $\alpha(\sigma_j)_l$ for $l \in \{1, \ldots, k\}$. On the other hand if v has in-edges, then let $u_1, u_2, \cdots, u_j$ be the nodes with out-edges to v. For each $i \in \{1, 2, \cdots, j\}$, using the uniqueness of the index received from u_i, node v recovers the equivalence classes

$$\Phi_{I_{\mathcal{E}_o(u_i)},f}(x^{(l)}_{I_{\mathcal{E}_o(u_i)}}) \quad \text{for} \quad l \in \{1, \cdots, k\}. \tag{2.10}$$

Furthermore, the equivalence classes in (2.9) can be identified by v from the equivalance classes in (2.10) (and $\alpha(v)$ if v is a source node) using the fact that for a multi-edge tree network $\mathcal{N}$, we have a disjoint union

$$I_{\mathcal{E}_o(v)} = \bigcup_{i=1}^{j} I_{\mathcal{E}_o(u_i)}.$$

If each node v follows the above steps, then the receiver ρ can identify the equivalence classes $\Phi_{I_{\mathcal{E}_i(\rho)},f}\left(x^{(i)}\right)$ for $i \in \{1, \ldots, k\}$. The receiver can evaluate $f(x^{(l)})$ for each l from these equivalence classes. The above solution achieves a computing rate of k/n. From (2.7), it follows that

$$\mathcal{C}_{\text{cod}}(\mathcal{N}, f) \geq \min_{v \in \mathcal{V}-\rho} \frac{|\mathcal{E}_o(v)|}{\log_{|\mathcal{A}|} R_{I_{\mathcal{E}_o(v)},f}}. \tag{2.11}$$

□

We next establish a general lower bound on the computing capacity for arbitrary target functions (Theorem 2.3.5) and then another lower bound specifically for symmetric target functions (Theorem 2.3.7).

For any network $\mathcal{N} = (G, S, \rho)$ with $G = (\mathcal{V}, \mathcal{E})$, define a *Steiner tree*[6] of $\mathcal{N}$ to be a minimal (with respect to nodes and edges) subgraph of G containing S and ρ such that every source in S has a directed path to the receiver ρ. Note that every non-receiver node in a Steiner tree has exactly one out-edge. Let $\mathcal{T}(\mathcal{N})$ denote the collection of all Steiner trees in $\mathcal{N}$. For each edge $e \in \mathcal{E}(G)$, let $J_e = \{i : t_i \in \mathcal{T}(\mathcal{N}) \text{ and } e \in \mathcal{E}(t_i)\}$. The *fractional Steiner tree packing number* $\Pi(\mathcal{N})$ is defined as the linear program

$$\Pi(\mathcal{N}) = \max \sum_{t_i \in \mathcal{T}(\mathcal{N})} u_i \quad \text{subject to} \quad \begin{cases} u_i \geq 0 \quad \forall\, t_i \in \mathcal{T}(\mathcal{N}) \ , \\ \sum_{i \in J_e} u_i \leq 1 \quad \forall\, e \in \mathcal{E}(G). \end{cases} \tag{2.12}$$

Note that $\Pi(\mathcal{N}) \geq 1$ for any network $\mathcal{N}$, and the maximum value of the sum in (2.12) is attained at one or more vertices of the closed polytope corresponding to the linear constraints. Since all coefficients in the constraints are rational, the maximum value in (2.12) can be attained with rational u_i's. The following theorem provides a lower bound[7] on the computing capacity for any network $\mathcal{N}$ with respect to a target function f and uses the quantity $\Pi(\mathcal{N})$. In the context of computing functions, u_i in the above linear program indicates the fraction of the time the edges in tree t_i are used to compute the desired function. The fact that every edge in the network has unit capacity implies $\sum_{i \in J_e} u_i \leq 1$.

[6] Steiner trees are well known in the literature for undirected graphs. For directed graphs a "Steiner tree problem" has been studied and our definition is consistent with such work (e.g., see [Jain 03]).

[7] In order to compute the lower bound, the fractional Steiner tree packing number $\Pi(\mathcal{N})$ can be evaluated using linear programming. Also note that if we construct the *reverse multicast network* by letting each source in the original network $\mathcal{N}$ become a receiver, letting the receiver in the $\mathcal{N}$ become the only source, and reversing the direction of each edge, then it can be verified that the routing capacity for the reverse multicast network is equal to $\Pi(\mathcal{N})$.

Lemma 2.3.4. *For any Steiner tree G' of a network $\mathcal{N}$, let $\mathcal{N}' = (G', S, \rho)$. Let C' be a cut in $\mathcal{N}'$. Then there exists a cut C in $\mathcal{N}$ such that $I_C = I_{C'}$.*

(Note that $I_{C'}$ is the set indices of sources separated in $\mathcal{N}'$ by C'. The set $I_{C'}$ may differ from the indices of sources separated in $\mathcal{N}$ by C'.)

Proof. Define the cut

$$C = \bigcup_{i' \in I_{C'}} \mathcal{E}_o(\sigma_{i'}). \tag{2.13}$$

C is the collection of out-edges in $\mathcal{N}$ of a set of sources disconnected by the cut C' in $\mathcal{N}'$. If $i \in I_{C'}$, then, by (2.13), C disconnects σ_i from ρ in $\mathcal{N}$, and thus $I_{C'} \subseteq I_C$.

Let σ_i be a source. such that $i \in I_C$ and Let P be a path from σ_i to ρ in $\mathcal{N}$. From (2.13), it follows that there exists $i' \in I_{C'}$ such that P contains at least one edge in $\mathcal{E}_o(\sigma_{i'})$. If P also lies in $\mathcal{N}'$ and does not contain any edge in C', then $\sigma_{i'}$ has a path to ρ in $\mathcal{N}'$ that does not contain any edge in C', thus contradicting the fact that $\sigma_{i'} \in I_{C'}$. Therefore, either P does not lie in $\mathcal{N}'$ or P contains an edge in C'. Thus $\sigma_i \in I_{C'}$, i.e., $I_C \subseteq I_{C'}$. □

Theorem 2.3.5. *If $\mathcal{N}$ is a network with alphabet $\mathcal{A}$ and target function f, then*

$$\mathcal{C}_{cod}(\mathcal{N}, f) \geq \Pi(\mathcal{N}) \cdot \min_{C \in \Lambda(\mathcal{N})} \frac{1}{\log_{|\mathcal{A}|} R_{C,f}}.$$

Proof. Suppose $\mathcal{N} = (G, S, \rho)$. Consider a Steiner tree $G' = (\mathcal{V}', \mathcal{E}')$ of $\mathcal{N}$, and let $\mathcal{N}' = (G', S, \rho)$. From Lemma 2.3.4 (taking C' to be $\mathcal{E}_o(v)$ in $\mathcal{N}'$), we have

$$\forall\ v \in \mathcal{V}' - \rho,\ \exists\ C \in \Lambda(\mathcal{N}) \text{ such that } I'_{\mathcal{E}_o(v)} = I_C. \tag{2.14}$$

Now we lower bound the computing capacity for the network $\mathcal{N}'$ with respect to target function f.

$$\begin{aligned}
\mathcal{C}_{\text{cod}}(\mathcal{N}', f) &= \text{min-cut}(\mathcal{N}', f) && \text{[from Theorem 2.3.3]} && (2.15)\\
&= \min_{v\ \in\ \mathcal{V}'-\rho} \frac{1}{\log_{|\mathcal{A}|} R_{I'_{\mathcal{E}_o(v)},f}} && \text{[from Theorem 2.2.1, (2.6), (2.11)]}\\
&\geq \min_{C \in \Lambda(\mathcal{N})} \frac{1}{\log_{|\mathcal{A}|} R_{I_C,f}} && \text{[from (2.14)].} && (2.16)
\end{aligned}$$

The lower bound in (2.16) is the same for every Steiner tree of $\mathcal{N}$. We will use this uniform bound to lower bound the computing capacity for $\mathcal{N}$ with respect to f. Denote the Steiner trees of $\mathcal{N}$ by $t_1, \ldots, t_T$. Let $\epsilon > 0$ and let r denote the quantity on the right hand side of (2.16). On every Steiner tree t_i, a computing rate of at least $r - \epsilon$ is achievable by (2.16). Using standard arguments for time-sharing between the different Steiner trees of the network $\mathcal{N}$, it follows that a computing rate of at least $(r - \epsilon) \cdot \Pi(\mathcal{N})$ is achievable in $\mathcal{N}$, and by letting $\epsilon \to 0$, the result follows. □

The lower bound in Theorem 2.3.5 can be readily computed and is sometimes tight. The procedure used in the proof of Theorem 2.3.5 may potentially be improved by maximizing the sum

$$\sum_{t_i \in \mathcal{T}(\mathcal{N})} u_i \, r_i \quad \text{subject to} \quad \begin{cases} u_i \geq 0 \quad \forall \, t_i \in \mathcal{T}(\mathcal{N}) \ , \\ \sum_{i \in J_e} u_i \leq 1 \quad \forall \, e \in \mathcal{E}(G) \end{cases} \tag{2.17}$$

where r_i is any achievable rate[8] for computing f in the Steiner tree network $\mathcal{N}_i = (t_i, S, \rho)$.

We now obtain a different lower bound on the computing capacity in the special case when the target function is the arithmetic sum. This lower bound is then used to give an alternative lower bound (in Theorem 2.3.7) on the computing capacity for the class of symmetric target functions. The bound obtained in Theorem 2.3.7 is sometimes better than that of Theorem 2.3.5, and sometimes worse (Example 2.3.8 illustrates instances of both cases).

Theorem 2.3.6. *If $\mathcal{N}$ is a network with alphabet $\mathcal{A} = \{0, 1, \ldots, q-1\}$ and the arithmetic sum target function f, then*

$$\mathcal{C}_{cod}(\mathcal{N}, f) \geq \min_{C \in \Lambda(\mathcal{N})} \frac{|C|}{\log_q P_{q,s}}$$

where $P_{q,s}$ denotes the smallest prime number greater than $s(q-1)$.

Proof. Let $p = P_{q,s}$ and let $\mathcal{N}'$ denote the same network as $\mathcal{N}$ but whose alphabet is , the finite field of order p.

Let $\epsilon > 0$. From Theorem 2.3.2, there exists a (k, n) solution for computing the -sum of the source messages in $\mathcal{N}'$ with an achievable computing rate satisfying

$$\frac{k}{n} \geq \min_{C \in \Lambda(\mathcal{N})} |C| - \epsilon.$$

This (k, n) solution can be repeated to derive a (ck, cn) solution for any integer $c \geq 1$ (note that edges in the network $\mathcal{N}$ carry symbols from the alphabet $\mathcal{A} = \{0, 1, \ldots, q-1\}$, while those in the network $\mathcal{N}'$ carry symbols from a larger alphabet). Any (ck, cn) solution for computing the -sum in $\mathcal{N}'$ can be 'simulated' in the network $\mathcal{N}$ by a $\left(ck, \lceil cn \log_q p \rceil\right)$ code (e.g. see [Appuswamy 09]). Furthermore, since $p \geq s(q-1) + 1$ and the source alphabet is $\{0, 1, \ldots, q-1\}$, the -sum of the source messages in network $\mathcal{N}$ is equal to their arithmetic sum. Thus, by choosing c large enough, the arithmetic sum target function is computed in $\mathcal{N}$ with an achievable computing rate of at least

$$\frac{\min_{C \in \Lambda(\mathcal{N})} |C|}{\log_q p} - 2\epsilon.$$

Since ϵ is arbitrary, the result follows. □

[8] From Theorem 2.3.3, r_i can be arbitrarily close to min-cut(t_i, f).

Theorem 2.3.7. *If $\mathcal{N}$ is a network with alphabet $\mathcal{A} = \{0, 1, \ldots, q-1\}$ and a symmetric target function f, then*

$$\mathcal{C}_{cod}(\mathcal{N}, f) \geq \frac{\min\limits_{C \in \Lambda(\mathcal{N})} |C|}{(q-1) \cdot \log_q P(s)}$$

where $P(s)$ is the smallest prime number[9] *greater than s.*

Proof. From Definition 2.1.10, it suffices to evaluate the histogram target function $\hat{f}$ for computing f. For any set of source messages $(x_1, x_2, \ldots, x_s) \in \mathcal{A}^s$, we have

$$\hat{f}(x_1, \ldots, x_s) = (c_0, c_1, \ldots, c_{q-1})$$

where $c_i = |\{j : x_j = i\}|$ for each $i \in \mathcal{A}$. Consider the network $\mathcal{N}' = (G, S, \rho)$ with alphabet $\mathcal{A}' = \{0, 1\}$. Then for each $i \in \mathcal{A}$, c_i can be evaluated by computing the arithmetic sum target function in $\mathcal{N}'$ where every source node σ_j is assigned the message 1 if $x_j = i$, and 0 otherwise. Since we know that

$$\sum_{i=0}^{q-1} c_i = s$$

the histogram target function $\hat{f}$ can be evaluated by computing the arithmetic sum target function $q-1$ times in the network $\mathcal{N}'$ with alphabet $\mathcal{A}' = \{0, 1\}$. Let $\epsilon > 0$. From Theorem 2.3.6 in the Appendix, there exists a (k, n) solution for computing the arithmetic sum target function in $\mathcal{N}'$ with an achievable computing rate of at least

$$\frac{k}{n} \geq \frac{\min\limits_{C \in \Lambda(\mathcal{N})} |C|}{\log_2 P(s)} - \epsilon.$$

The above (k, n) solution can be repeated to derive a (ck, cn) solution for any integer $c \geq 1$. Note that edges in the network $\mathcal{N}$ carry symbols from the alphabet $\mathcal{A} = \{0, 1, \ldots, q-1\}$, while those in the network $\mathcal{N}'$ carry symbols from $\mathcal{A}' = \{0, 1\}$. Any (ck, cn) code for computing the arithmetic sum function in $\mathcal{N}'$ can be simulated in the network $\mathcal{N}$ by a $(ck, \lceil cn \log_q 2 \rceil)$ code[10]. Thus by choosing c large enough, the above-mentioned code can be simulated in the network $\mathcal{N}$ to derive a solution for computing the histogram target function $\hat{f}$ with an achievable computing rate[11] of at least

$$\frac{1}{(q-1)} \cdot \frac{1}{\log_q 2} \cdot \frac{\min\limits_{C \in \Lambda(\mathcal{N})} |C|}{\log_2 P(s)} - 2\epsilon.$$

Since ϵ is arbitrary, the result follows. □

[9] From Bertrand's Postulate [Hardy 79, p.343], we have $P(s) \leq 2s$.

[10] To see details of such a simulation, we refer the interested reader to [Appuswamy 09].

[11] Theorem 2.3.7 provides a uniform lower bound on the achievable computing rate for any symmetric function. Better lower bounds can be found by considering specific functions; for example Theorem 2.3.6 gives a better bound for the arithmetic sum target function.

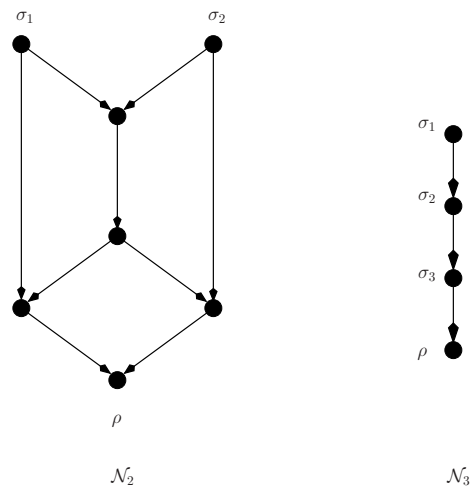


Figure 2.3: The Reverse Butterfly Network $\mathcal{N}_2$ has two binary sources $\{\sigma_1, \sigma_2\}$ and network $\mathcal{N}_3$ has three binary sources $\{\sigma_1, \sigma_2, \sigma_3\}$, each with $\mathcal{A} = \{0, 1\}$. Each network's receiver ρ computes the arithmetic sum of the source messages.

Example 2.3.8. Consider networks $\mathcal{N}_2$ and $\mathcal{N}_3$ in Figure 2.3, each with alphabet $\mathcal{A} = \{0, 1\}$ and the (symmetric) arithmetic sum target function f. Theorem 2.3.7 provides a larger lower bound on the computing capacity $\mathcal{C}_{\text{cod}}(\mathcal{N}_2, f)$ than Theorem 2.3.5, but a smaller lower bound on $\mathcal{C}_{\text{cod}}(\mathcal{N}_3, f)$.

- For network $\mathcal{N}_2$ (in Figure 2.3), we have $\max_{C \in \Lambda(\mathcal{N})} R_{C,f} = 3$ and $\min_{C \in \Lambda(\mathcal{N})} |C| = 2$, both of which occur, for example, when C consists of the two in-edges to the receiver ρ. Also, $(q-1)\log_q P(s,q) = \log_2 3$ and $\Pi(\mathcal{N}) = 3/2$, so

$$\begin{aligned} \mathcal{C}_{\text{cod}}(\mathcal{N}_2, f) &\geq (3/2)/\log_2 3 && \text{[from Theorem 2.3.5]} \\ \mathcal{C}_{\text{cod}}(\mathcal{N}_2, f) &\geq 2/\log_2 3 && \text{[from Theorem 2.3.7].} \end{aligned} \tag{2.18}$$

  In fact, we get the upper bound $\mathcal{C}_{\text{cod}}(\mathcal{N}_2, f) \leq 2/\log_2 3$ from Theorem 2.2.1, and thus from (2.18), $\mathcal{C}_{\text{cod}}(\mathcal{N}_2, f) = 2/\log_2 3$.

- For network $\mathcal{N}_3$, we have $\max_{C \in \Lambda(\mathcal{N})} R_{C,f} = 4$ and $\min_{C \in \Lambda(\mathcal{N})} |C| = 1$, both of which occur when $C = \{(\sigma_3, \rho)\}$. Also, $(q-1)\log_q P(s,q) = \log_2 5$ and $\Pi(\mathcal{N}) = 1$, so

$$\begin{aligned} \mathcal{C}_{\text{cod}}(\mathcal{N}_3, f) &\geq 1/\log_2 4 && \text{[from Theorem 2.3.5]} \\ \mathcal{C}_{\text{cod}}(\mathcal{N}_3, f) &\geq 1/\log_2 5 && \text{[from Theorem 2.3.7].} \end{aligned}$$

  From Theorem 2.3.3, we have $\mathcal{C}_{\text{cod}}(\mathcal{N}_3, f) = 1/\log_2 4$.

Remark 2.3.9. An open question, pointed out in [Cannons 06], is whether the coding capacity of a network can be irrational. Like the coding capacity, the computing capacity is the supremum of ratios k/n for which a (k, n) solution exists. Example 2.3.8 demonstrates that

the computing capacity of a network (e.g. $\mathcal{N}_2$) with unit capacity links can be irrational when the target function is the arithmetic sum function.

2.4 On the tightness of the min-cut upper bound

In the previous section, Theorems 2.3.1 - 2.3.3 demonstrated three special instances for which the min-cut$(\mathcal{N}, f)$ upper bound is tight. In this section, we use Theorem 2.3.5 and Theorem 2.3.7 to establish further results on the tightness of the min-cut$(\mathcal{N}, f)$ upper bound for different classes of target functions.

The following lemma provides a bound on the footprint size $R_{I,f}$ for any divisible target function f.

Lemma 2.4.1. *For any divisible target function $f : \mathcal{A}^s \longrightarrow \mathcal{B}$ and any index set $I \subseteq \{1, 2, \ldots, s\}$, the footprint size satisfies*

$$R_{I,f} \leq |f(\mathcal{A}^s)|.$$

Proof. From the definition of a divisible target function, for any $I \subseteq \{1, 2, \ldots, s\}$, there exist maps f^I, f^{I^c}, and g such that

$$f(x) = g\left(f^I(x_I), f^{I^c}(x_{I^c})\right) \qquad \forall\, x \in \mathcal{A}^s$$

where $I^c = \{1, 2, \ldots, s\} - I$. From the definition of the equivalence relation $\equiv$ (see Definition 2.1.5), it follows that $a, b \in \mathcal{A}^{|I|}$ belong to the same equivalence class whenever $f^I(a) = f^I(b)$. This fact implies that $R_{I,f} \leq \left|f^I\left(\mathcal{A}^{|I|}\right)\right|$. We need $\left|f^I\left(\mathcal{A}^{|I|}\right)\right| \leq |f(\mathcal{A}^s)|$ to complete the proof which follows from Definition 2.1.9(2). □

Theorem 2.4.2. *If $\mathcal{N}$ is a network with a divisible target function f, then*

$$\mathcal{C}_{cod}(\mathcal{N}, f) \geq \frac{\Pi(\mathcal{N})}{|\mathcal{E}_i(\rho)|} \cdot \text{min-cut}(\mathcal{N}, f)$$

where $\mathcal{E}_i(\rho)$ denotes the set of in-edges of the receiver ρ.

Proof. Let $\mathcal{A}$ be the network alphabet. From Theorem 2.3.5,

$$\begin{aligned}\mathcal{C}_{\text{cod}}(\mathcal{N}, f) &\geq \Pi(\mathcal{N}) \cdot \min_{C \in \Lambda(\mathcal{N})} \frac{1}{\log_{|\mathcal{A}|} R_{C,f}} \\ &\geq \Pi(\mathcal{N}) \cdot \frac{1}{\log_{|\mathcal{A}|} |f(\mathcal{A}^s)|} \qquad \text{[from Lemma 2.4.1].} \end{aligned} \tag{2.19}$$

On the other hand, for any network $\mathcal{N}$, the set of edges $\mathcal{E}_i(\rho)$ is a cut that separates the set of sources S from ρ. Thus,

$$\begin{aligned}\text{min-cut}(\mathcal{N}, f) &\leq \frac{|\mathcal{E}_i(\rho)|}{\log_{|\mathcal{A}|} R_{\mathcal{E}_i(\rho),f}} \qquad \text{[from (2.4)]} \\ &= \frac{|\mathcal{E}_i(\rho)|}{\log_{|\mathcal{A}|} |f(\mathcal{A}^s)|} \qquad \text{[from } I_{\mathcal{E}_i(\rho)} = S \text{ and Definition 2.1.6].} \end{aligned} \tag{2.20}$$

Combining (2.19) and (2.20) completes the proof. □

Theorem 2.4.3. *If $\mathcal{N}$ is a network with alphabet $\mathcal{A} = \{0, 1, \ldots, q-1\}$ and symmetric target function f, then*

$$\mathcal{C}_{cod}(\mathcal{N}, f) \geq \frac{\log_q \hat{R}_f}{(q-1) \cdot \log_q P(s)} \cdot \text{min-cut}(\mathcal{N}, f)$$

where $P(s)$ is the smallest prime number greater than s and[12]

$$\hat{R}_f = \min_{I \subseteq \{1,\ldots,s\}} R_{I,f}.$$

Proof. The result follows immediately from Theorem 2.3.7 and since for any network $\mathcal{N}$ and any target function f,

$$\text{min-cut}(\mathcal{N}, f) \leq \frac{1}{\log_q \hat{R}_f} \min_{C \in \Lambda(\mathcal{N})} |C| \qquad \text{[from (2.4) and the definition of } \hat{R}_f\text{]}.$$

□

The following results provide bounds on the gap between the computing capacity and the min-cut for λ-exponential and λ-bounded functions (see Definition 2.1.11).

Theorem 2.4.4. *If $\lambda \in (0, 1]$ and $\mathcal{N}$ is a network with a λ-exponential target function f, then*

$$\mathcal{C}_{cod}(\mathcal{N}, f) \geq \lambda \cdot \text{min-cut}(\mathcal{N}, f).$$

Proof. We have

$$\begin{aligned}
\text{min-cut}(\mathcal{N}, f) &= \min_{C \in \Lambda(\mathcal{N})} \frac{|C|}{\log_{|\mathcal{A}|} R_{C,f}} \\
&\leq \min_{C \in \Lambda(\mathcal{N})} \frac{|C|}{\lambda\, |I_C|} && \text{[from } f \text{ being } \lambda\text{-exponential]} \\
&= \frac{1}{\lambda} \cdot \text{min-cut}(\mathcal{N}) && \text{[from (2.3)]}.
\end{aligned}$$

Therefore,

$$\begin{aligned}
\frac{\text{min-cut}(\mathcal{N}, f)}{\mathcal{C}_{\text{cod}}(\mathcal{N}, f)} &\leq \frac{1}{\lambda} \cdot \frac{\text{min-cut}(\mathcal{N})}{\mathcal{C}_{\text{cod}}(\mathcal{N}, f)} \\
&\leq \frac{1}{\lambda}
\end{aligned}$$

where the last inequality follows because a computing rate of min-cut$(\mathcal{N})$ is achievable for the identity target function from Theorem 2.3.1, and the computing capacity for any target function f is lower bounded by the computing capacity for the identity target function (since any target function can be computed from the identity function), i.e., $\mathcal{C}_{\text{cod}}(\mathcal{N}, f) \geq \text{min-cut}(\mathcal{N})$. □

Theorem 2.4.5. *Let $\lambda > 0$. If $\mathcal{N}$ is a network with alphabet $\mathcal{A}$ and a λ-bounded target function f, and all non-receiver nodes in the network $\mathcal{N}$ are sources, then*

$$\mathcal{C}_{cod}(\mathcal{N}, f) \geq \frac{\log_{|\mathcal{A}|} \hat{R}_f}{\lambda} \cdot \text{min-cut}(\mathcal{N}, f)$$

[12] From our assumption, $\hat{R}_f \geq 2$ for any target function f.

where

$$\hat{R}_f = \min_{I \subseteq \{1,\ldots,s\}} R_{I,f}.$$

Proof. For any network $\mathcal{N}$ such that all non-receiver nodes are sources, it follows from Edmond's Theorem [West 01, p.405, Theorem 8.4.20] that

$$\Pi(\mathcal{N}) = \min_{C \in \Lambda(\mathcal{N})} |C|.$$

Then,

$$\begin{aligned} \mathcal{C}_{\text{cod}}(\mathcal{N}, f) &\geq \min_{C \in \Lambda(\mathcal{N})} |C| \cdot \min_{C \in \Lambda(\mathcal{N})} \frac{1}{\log_{|\mathcal{A}|} R_{C,f}} && \text{[from Theorem 2.3.5]} \\ &\geq \min_{C \in \Lambda(\mathcal{N})} \frac{|C|}{\lambda} && \text{[from } f \text{ being } \lambda\text{-bounded].} \end{aligned} \tag{2.21}$$

On the other hand,

$$\begin{aligned} \text{min-cut}(\mathcal{N}, f) &= \min_{C \in \Lambda(\mathcal{N})} \frac{|C|}{\log_{|\mathcal{A}|} R_{C,f}} \\ &\leq \min_{C \in \Lambda(\mathcal{N})} \frac{|C|}{\log_{|\mathcal{A}|} \hat{R}_f} && \text{[from the definition of } \hat{R}_f\text{].} \end{aligned} \tag{2.22}$$

Combining (2.21) and (2.22) gives

$$\begin{aligned} \frac{\text{min-cut}(\mathcal{N}, f)}{\mathcal{C}_{\text{cod}}(\mathcal{N}, f)} &\leq \min_{C \in \Lambda(\mathcal{N})} \frac{|C|}{\log_{|\mathcal{A}|} \hat{R}_f} \cdot \frac{1}{\min_{C \in \Lambda(\mathcal{N})} \frac{|C|}{\lambda}} \\ &= \frac{\lambda}{\log_{|\mathcal{A}|} \hat{R}_f}. \end{aligned}$$

□

Since the maximum and minimum functions are 1-bounded, and $\hat{R}_f = |\mathcal{A}|$ for each, we get the following corollary.

Corollary 2.4.6. *Let $\mathcal{A}$ be any ordered alphabet and let $\mathcal{N}$ be any network such that all non-receiver nodes in the network are sources. If the target function f is either the maximum or the minimum function, then*

$$\mathcal{C}_{cod}(\mathcal{N}, f) = \text{min-cut}(\mathcal{N}, f).$$

Theorems 2.4.2 - 2.4.5 provide bounds on the tightness of the min-cut$(\mathcal{N}, f)$ upper bound for different classes of target functions. In particular, we show that for λ-exponential (respectively, λ-bounded) target functions, the computing capacity $\mathcal{C}_{\text{cod}}(\mathcal{N}, f)$ is at least a constant fraction of the min-cut$(\mathcal{N}, f)$ for any constant λ and any network $\mathcal{N}$ (respectively, any network $\mathcal{N}$ where all non-receiver nodes are sources). The following theorem shows by

means of an example target function f and a network $\mathcal{N}$, that the min-cut$(\mathcal{N}, f)$ upper bound cannot always approximate the computing capacity $\mathcal{C}_{\text{cod}}(\mathcal{N}, f)$ up to a constant fraction. Similar results are known in network coding as well as in multicommodity flow. It was shown in [Leighton 99] that when s source nodes communicate independently with the same number of receiver nodes, there exist networks whose maximum multicommodity flow is $O(1/\log s)$ times a well known cut-based upper bound. It was shown in [Harvey 04] that with network coding there exist networks whose maximum throughput is $O(1/\log s)$ times the best known cut bound (i.e. ŞmeagernessŤ). Whereas these results do not hold for single-receiver networks (by Theorem 2.3.1), the following similar bound holds for network computing in single-receiver networks. The proof of Theorem 2.4.7 uses Lemma 2.7.1 which is presented in the Appendix.

Theorem 2.4.7. *For any $\epsilon > 0$, there exist networks $\mathcal{N}$ such that for the arithmetic sum target function f,*

$$\mathcal{C}_{cod}(\mathcal{N}, f) = O\left(\frac{1}{(\log s)^{1-\epsilon}}\right) \cdot \text{min-cut}(\mathcal{N}, f).$$

Proof. Note that for the network $\mathcal{N}_{M,L}$ and the arithmetic sum target function f,

$$\text{min-cut}(\mathcal{N}_{M,L}, f) = \min_{C \in \Lambda(\mathcal{N}_{M,L})} \frac{|C|}{\log_2(|I_C| + 1)} \qquad \text{[from (2.5)]}.$$

Let m be the number of sources disconnected from the receiver ρ by a cut C in the network $\mathcal{N}_{M,L}$. For each such source σ, the cut C must contain the edge (σ, ρ) as well as either the L parallel edges (σ, σ_0) or the L parallel edges (σ_0, ρ). Thus,

$$\text{min-cut}(\mathcal{N}_{M,L}, f) = \min_{1 \le m \le M} \left\{\frac{L + m}{\log_2(m + 1)}\right\}. \tag{2.23}$$

Let m^* attain the minimum in (2.23) and define $c^* = \text{min-cut}(\mathcal{N}_{M,L}, f)$. Then,

$$\begin{aligned} c^*/\ln 2 &\ge \min_{1 \le m \le M} \left\{\frac{m+1}{\ln(m+1)}\right\} \ge \min_{x \ge 2} \left\{\frac{x}{\ln x}\right\} > \min_{x \ge 2} \left\{\frac{x}{x-1}\right\} > 1 \\ L &= c^* \log_2(m^* + 1) - m^* && \text{[from (2.23)]} \\ &\le c^* \log_2\left(\frac{c^*}{\ln 2}\right) - \left(\frac{c^*}{\ln 2} - 1\right) && (2.24) \end{aligned}$$

where (2.24) follows since the function $c^* \log_2(x+1) - x$ attains its maximum value over $(0, \infty)$ at $x = (c^*/\ln 2) - 1$. Let us choose $L = \lceil (\log M)^{1-(\epsilon/2)} \rceil$. We have

$$\begin{aligned} L &= O(\text{min-cut}(\mathcal{N}_{M,L}, f) \log_2(\text{min-cut}(\mathcal{N}_{M,L}, f))) && \text{[from (2.24)]} && (2.25) \\ \text{min-cut}(\mathcal{N}_{M,L}, f) &= \Omega((\log M)^{1-\epsilon}) && \text{[from (2.25)]} && (2.26) \\ \mathcal{C}_{\text{cod}}(\mathcal{N}_{M,L}, f) &= O(1) && \text{[from Lemma 2.7.1]} \\ &= O\left(\frac{1}{(\log M)^{1-\epsilon}}\right) \cdot \text{min-cut}(\mathcal{N}_{M,L}, f) && \text{[from (2.26)]}. \end{aligned}$$

□

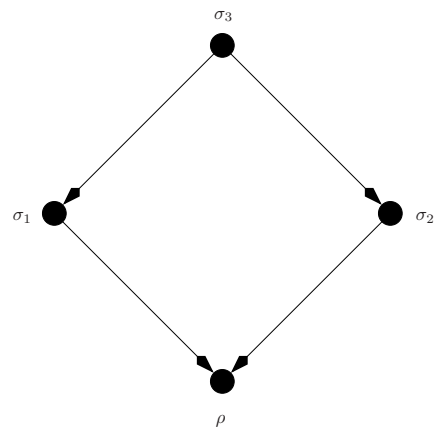


Figure 2.4: Network $\hat{\mathcal{N}}$ has three binary sources, σ_1, σ_2, and σ_3 with $\mathcal{A} = \{0, 1\}$ and the receiver ρ computes the arithmetic sum of the source messages.

2.5 An example network

In this section, we evaluate the computing capacity for an example network and a target function (which is divisible and symmetric) and show that the min-cut bound is not tight. In addition, the example demonstrates that the lower bounds discussed in Section 2.3 are not always tight and illustrates the combinatorial nature of the computing problem.

Theorem 2.5.1. *The computing capacity of network $\hat{\mathcal{N}}$ with respect to the arithmetic sum target function f is*

$$\mathcal{C}_{cod}\Big(\hat{\mathcal{N}}, f\Big) = \frac{2}{1 + \log_2 3}.$$

Proof. For any (k, n) solution for computing f, let $w^{(1)}, w^{(2)}, w^{(3)} \in \{0, 1\}^k$ denote the message vectors generated by sources σ_1, σ_2, σ_3, respectively, and let $z_1, z_2 \in \{0, 1\}^n$ be the vectors carried by edges (σ_1, ρ) and (σ_2, ρ), respectively.

Consider any positive integers k, n such that k is even and

$$\frac{k}{n} \leq \frac{2}{1 + \log_2 3}. \tag{2.27}$$

Then we have

$$2^n \geq 3^{k/2} 2^{k/2}. \tag{2.28}$$

We will describe a (k, n) network code for computing f in the network $\hat{\mathcal{N}}$. Define vectors $y^{(1)}, y^{(2)} \in \{0, 1\}^k$ by:

$$y_i^{(1)} = \begin{cases} w_i^{(1)} + w_i^{(3)} & \text{if } 1 \leq i \leq k/2 \\ w_i^{(1)} & \text{if } k/2 \leq i \leq k \end{cases}$$

$$y_i^{(2)} = \begin{cases} w_i^{(2)} & \text{if } 1 \leq i \leq k/2 \\ w_i^{(2)} + w_i^{(3)} & \text{if } k/2 \leq i \leq k. \end{cases}$$

The first $k/2$ components of $y^{(1)}$ can take on the values $0, 1, 2$, and the last $k/2$ components can take on the values $0, 1$, so there are a total of $3^{k/2}2^{k/2}$ possible values for $y^{(1)}$, and similarly for $y^{(2)}$. From (2.28), there exists a mapping that assigns unique values to z_1 for each different possible value of $y^{(1)}$, and similarly for z_2 and $y^{(2)}$. This induces a solution for $\hat{\mathcal{N}}$ as summarized below.

The source σ_3 sends its full message vector $w^{(3)}$ $(k < n)$ to each of the two nodes it is connected to. Source σ_1 (respectively, σ_2) computes the vector $y^{(1)}$ (respectively, $y^{(2)}$), then computes the vector z_1 (respectively, z_2), and finally sends z_1 (respectively, z_2) on its out-edge. The receiver ρ determines $y^{(1)}$ and $y^{(2)}$ from z_1 and z_2, respectively, and then computes $y^{(1)} + y^{(2)}$, whose i-th component is $w_i^{(1)} + w_i^{(2)} + w_i^{(3)}$, i.e., the arithmetic sum target function f. The above solution achieves a computing rate of k/n. From (2.27), it follows that

$$\mathcal{C}_{\text{cod}}\Big(\hat{\mathcal{N}}, f\Big) \geq \frac{2}{1 + \log_2 3}. \tag{2.29}$$

We now prove a matching upper bound on the computing capacity $\mathcal{C}_{\text{cod}}\Big(\hat{\mathcal{N}}, f\Big)$. Consider any (k, n) solution for computing the arithmetic sum target function f in network $\hat{\mathcal{N}}$. For any $p \in \{0, 1, 2, 3\}^k$, let

$$A_p = \{(z_1, z_2) : w^{(1)} + w^{(2)} + w^{(3)} = p\}.$$

That is, each element of A_p is a possible pair of input edge-vectors to the receiver when the target function value equals p.

Let j denote the number of components of p that are either 0 or 3. Without loss of generality, suppose the first j components of p belong to $\{0, 3\}$ and define $\tilde{w}^{(3)} \in \{0, 1\}^k$ by

$$\tilde{w}_i^{(3)} = \begin{cases} 0 & \text{if } p_i \in \{0, 1\} \\ 1 & \text{if } p_i \in \{2, 3\}. \end{cases}$$

Let

$$T = \{(w^{(1)}, w^{(2)}) \in \{0, 1\}^k \times \{0, 1\}^k \ : \ w^{(1)} + w^{(2)} + \tilde{w}^{(3)} = p\}$$

and notice that

$$\Big\{(z_1, z_2) : (w^{(1)}, w^{(2)}) \in T, w^{(3)} = \tilde{w}^{(3)}\Big\} \subseteq A_p. \tag{2.30}$$

If $w^{(1)} + w^{(2)} + \tilde{w}^{(3)} = p$, then:

(i) $p_i - \tilde{w}_i^{(3)} = 0$ implies $w_i^{(1)} = w_i^{(2)} = 0$;
(ii) $p_i - \tilde{w}_i^{(3)} = 2$ implies $w_i^{(1)} = w_i^{(2)} = 1$;
(iii) $p_i - \tilde{w}_i^{(3)} = 1$ implies $(w_i^{(1)}, w_i^{(2)}) = (0, 1)$ or $(1, 0)$.

Thus, the elements of T consist of k-bit vector pairs $(w^{(1)}, w^{(2)})$ whose first j components are fixed and equal (i.e., both are 0 when $p_i = 0$ and both are 1 when $p_i = 3$), and whose remaining

$k-j$ components can each be chosen from two possibilities (i.e., either $(0,1)$ or $(1,0)$, when $p_i \in \{1,2\}$). This observation implies that

$$|T| = 2^{k-j}. \tag{2.31}$$

Notice that if only $w^{(1)}$ changes, then the sum $w^{(1)}+w^{(2)}+w^{(3)}$ changes, and so z_1 must change (since z_2 is not a function of $w^{(1)}$) in order for the receiver to compute the target function. Thus, if $w^{(1)}$ changes and $w^{(3)}$ does not change, then z_1 must still change, regardless of whether $w^{(2)}$ changes or not. More generally, if the pair $(w^{(1)}, w^{(2)})$ changes, then the pair (z_1, z_2) must change. Thus,

$$\left|\left\{(z_1,z_2) : (w^{(1)}, w^{(2)}) \in T, w^{(3)} = \tilde{w}^{(3)}\right\}\right| \geq |T| \tag{2.32}$$

and therefore

$$\begin{aligned}
|A_p| &\geq \left|\left\{(z_1,z_2) : (w^{(1)}, w^{(2)}) \in T, w^{(3)} = \tilde{w}^{(3)}\right\}\right| && \text{[from (2.30)]} \\
&\geq |T| && \text{[from (2.32)]} \\
&= 2^{k-j}. && \text{[from (2.31)]} \qquad (2.33)
\end{aligned}$$

We have the following inequalities:

$$\begin{aligned}
4^n &\geq \left|\{(z_1,z_2) : w^{(1)}, w^{(2)}, w^{(3)} \in \{0,1\}^k\}\right| \\
&= \sum_{p\in\{0,1,2,3\}^k} |A_p| && (2.34) \\
&= \sum_{j=0}^{k} \sum_{\substack{p\in\{0,1,2,3\}^k \\ |\{i:p_i\in\{0,3\}\}|=j}} |A_p| \\
&\geq \sum_{j=0}^{k} \sum_{\substack{p\in\{0,1,2,3\}^k \\ |\{i:p_i\in\{0,3\}\}|=j}} 2^{k-j} && \text{[from (2.33)]} \\
&= \sum_{j=0}^{k} \binom{k}{j} 2^k 2^{k-j} \\
&= 6^k && (2.35)
\end{aligned}$$

where (2.34) follows since the A_p's must be disjoint in order for the receiver to compute the target function. Taking logarithms of both sides of (2.35), gives

$$\frac{k}{n} \leq \frac{2}{1+\log_2 3}$$

which holds for all k and n, and therefore

$$\mathcal{C}_{\text{cod}}\left(\hat{\mathcal{N}}, f\right) \leq \frac{2}{1+\log_2 3}. \tag{2.36}$$

Combining (2.29) and (2.36) concludes the proof. □

Corollary 2.5.2. *For the network $\hat{\mathcal{N}}$ with the arithmetic sum target function f,*

$$\mathcal{C}_{cod}\Big(\hat{\mathcal{N}}, f\Big) < \text{min-cut}\Big(\hat{\mathcal{N}}, f\Big).$$

Proof. Consider the network $\hat{\mathcal{N}}$ depicted in Figure 2.4 with the arithmetic sum target function f. It can be shown that the footprint size $R_{C,f} = |I_C| + 1$ for any cut C, and thus

$$\text{min-cut}\Big(\hat{\mathcal{N}}, f\Big) = 1 \qquad\qquad \text{[from (2.5)]}.$$

The result then follows immediately from Theorem 2.5.1. □

Remark 2.5.3. In light of Theorem 2.5.1, we compare the various lower bounds on the computing capacity of the network $\hat{\mathcal{N}}$ derived in Section 2.3 with the exact computing capacity. It can be shown that $\Pi\Big(\hat{\mathcal{N}}\Big) = 1$. If f is the arithmetic sum target function, then

$$\begin{aligned}
&\mathcal{C}_{\text{cod}}\Big(\hat{\mathcal{N}}, f\Big) \geq 1/2 && \text{[from Theorem 2.3.5]}\\
&\mathcal{C}_{\text{cod}}\Big(\hat{\mathcal{N}}, f\Big) \geq 1/\log_2 5 && \text{[from Theorem 2.3.7]}\\
&\mathcal{C}_{\text{cod}}\Big(\hat{\mathcal{N}}, f\Big) \geq 1/2 && \text{[from Theorem 2.4.2]}.
\end{aligned}$$

Thus, this example demonstrates that the lower bounds obtained in Section 2.3 are not always tight and illustrates the combinatorial nature of the problem.

2.6 Conclusions

We examined the problem of network computing. The network coding problem is a special case when the function to be computed is the identity. We have focused on the case when a single receiver node computes a function of the source messages and have shown that while for the identity function the min-cut bound is known to be tight for all networks, a much richer set of cases arises when computing arbitrary functions, as the min-cut bound can range from being tight to arbitrarily loose. One key contribution of the paper is to show the theoretical breadth of the considered topic, which we hope will lead to further research. This work identifies target functions (most notably, the arithmetic sum function) for which the min-cut bound is not always tight (even up to a constant factor) and future work includes deriving more sophisticated bounds for these scenarios. Extensions to computing with multiple receiver nodes, each computing a (possibly different) function of the source messages, are of interest.

2.7 Appendix

Define the function

$$Q \ : \ \prod_{i=1}^{M} \{0,1\}^k \longrightarrow \{0,1,\ldots,M\}^k$$

as follows. For every $a = (a^{(1)}, a^{(2)}, \ldots, a^{(M)})$ such that each $a^{(i)} \in \{0,1\}^k$,

$$Q(a)_j = \sum_{i=1}^{M} a_j^{(i)} \quad \text{for every } j \in \{1,2,\ldots,k\}. \tag{2.37}$$

We extend Q for $X \subseteq \prod_{i=1}^{M} \{0,1\}^k$ by defining $Q(X) = \{Q(a) \ : \ a \in X\}$.

We now present Lemma 2.7.1. The proof uses Lemma 2.7.2, which is presented thereafter. We define the following function which is used in the next lemma. Let

$$\gamma(x) = \mathcal{H}^{-1}\left(\frac{1}{2}\left(1 - \frac{1}{x}\right)\right) \bigcap \left[0, \frac{1}{2}\right] \qquad \text{for } x \geq 1 \tag{2.38}$$

where $\mathcal{H}^{-1}$ denotes the inverse of the binary entropy function $\mathcal{H}(x) = -x\log_2 x - (1-x)\log_2(1-x)$. Note that $\gamma(x)$ is an increasing function of x.

Lemma 2.7.1. *If* $\lim_{M\to\infty} \frac{L}{\log_2 M} = 0$, *then* $\lim_{M\to\infty} \mathcal{C}_{cod}(\mathcal{N}_{M,L}, f) = 1$.

Proof. For any M and L, a solution with computing rate 1 is obtained by having each source σ_i send its message directly to the receiver on the edge (σ_i, ρ). Hence $\mathcal{C}_{\text{cod}}(\mathcal{N}_{M,L}, f) \geq 1$. Now suppose that $\mathcal{N}_{M,L}$ has a (k,n) solution with computing rate $k/n > 1$ and for each $i \in \{1,2,\ldots,M\}$, let

$$g_i \ : \ \{0,1\}^k \longrightarrow \{0,1\}^n$$

be the corresponding encoding function on the edge (σ_i, ρ). Then for any $A_1, A_2, \ldots, A_M \subseteq \{0,1\}^k$, we have

$$\left(\prod_{i=1}^{M} |g_i(A_i)|\right) \cdot 2^{nL} \geq \left|Q\left(\prod_{i=1}^{M} A_i\right)\right|. \tag{2.39}$$

Each A_i represents a set of possible message vectors of source σ_i. The left-hand side of (2.39) is the maximum number of different possible instantiations of the information carried by the in-edges to the receiver ρ (i.e., $|g_i(A_i)|$ possible vectors on each edge (σ_i, ρ) and 2^{nL} possible vectors on the L parallel edges (σ_0, ρ)). The right-hand side of (2.39) is the number of distinct sum vectors that the receiver needs to discriminate, using the information carried by its in-edges.

For each $i \in \{1,2,\ldots,M\}$, let $z_i \in \{0,1\}^n$ be such that $\left|g_i^{-1}(z_i)\right| \geq 2^{k-n}$ and choose $A_i = g_i^{-1}(z_i)$ for each i. Also, let $U^{(M)} = \prod_{i=1}^{M} A_i$. Then we have

$$\left|Q\left(U^{(M)}\right)\right| \leq 2^{nL} \qquad \text{[from } |g_i(A_i)| = 1 \text{ and (2.39)]}. \tag{2.40}$$

Thus (2.40) is a necessary condition for the existence of a (k, n) solution for computing f in the network $\mathcal{N}_{M,L}$. Lemma 2.7.2 shows that[13]

$$\left|Q\left(U^{(M)}\right)\right| \geq (M+1)^{\gamma(k/n)k} \tag{2.41}$$

where the function γ is defined in (2.38). Combining (2.40) and (2.41), any (k, n) solution for computing f in the network $\mathcal{N}_{M,L}$ with rate $r = k/n > 1$ must satisfy

$$r\ \gamma(r)\ \log_2(M+1) \leq \frac{1}{n}\log_2\left|Q\left(U^{(M)}\right)\right| \leq L. \tag{2.42}$$

From (2.42), we have

$$r\ \gamma(r) \leq \frac{L}{\log_2(M+1)}. \tag{2.43}$$

The quantity $r\gamma(r)$ is monotonic increasing from 0 to ∞ on the interval $[1, \infty)$ and the right hand side of (2.43) goes to zero as $M \to \infty$. Thus, the rate r can be forced to be arbitrarily close to 1 by making M sufficiently large, i.e. $\mathcal{C}_{\text{cod}}(\mathcal{N}_{M,L}, f) \leq 1$. In summary,

$$\lim_{M \longrightarrow \infty} \mathcal{C}_{\text{cod}}(\mathcal{N}_{M,L}, f) = 1.$$

□

Lemma 2.7.2. *Let k, n, M be positive integers such that $k > n$. For each $i \in \{1, 2, \ldots, M\}$, let $A_i \subseteq \{0,1\}^k$ be such that $|A_i| \geq 2^{k-n}$ and let $U^{(M)} = \prod_{i=1}^{M} A_i$. Then,*

$$\left|Q\left(U^{(M)}\right)\right| \geq (M+1)^{\gamma(k/n)k}.$$

Proof. The result follows from Lemmas 2.7.4 and 2.7.7. □

The remainder of this Appendix is devoted to the proofs of lemmas used in the proof of Lemma 2.7.2. Before we proceed, we need to define some more notation. For every $j \in \{1, 2, \ldots, k\}$, define the map

$$h^{(j)}\ :\ \{0, 1, \ldots, M\}^k \ \longrightarrow\ \{0, 1, \ldots, M\}^k$$

by

$$\left(h^{(j)}(p)\right)_i = \begin{cases} \max\{0, p_i - 1\} & \text{if } i = j \\ p_i & \text{otherwise.} \end{cases} \tag{2.44}$$

That is, the map $h^{(j)}$ subtracts one from the j-th component of the input vector (as long as the result is non-negative) and leaves all the other components the same. For every $j \in \{1, 2, \ldots, k\}$, define the map

$$\hat{\phi}^{(j)} : 2^{\{0,1\}^k}\ \times\ \{0,1\}^k \longrightarrow \{0,1\}^k$$

[13] One can compare this lower bound to the upper bound $\left|Q\left(U^{(M)}\right)\right| \leq (M+1)^k$ which follows from (2.37).

by

$$\hat{\phi}^{(j)}(A,a) = \begin{cases} h^{(j)}(a) & \text{if } h^{(j)}(a) \notin A \\ a & \text{otherwise} \end{cases} \qquad \forall\ A \subseteq \{0,1\}^k \text{ and } a \in \{0,1\}^k. \tag{2.45}$$

Define

$$\phi^{(j)} : 2^{\{0,1\}^k} \longrightarrow 2^{\{0,1\}^k}$$

by

$$\phi^{(j)}(A) = \left\{ \hat{\phi}^{(j)}(A,a) \ : \ a \ \in \ A \right\}. \tag{2.46}$$

Note that

$$\left| \phi^{(j)}(A) \right| = |A|. \tag{2.47}$$

A set A is said to be *invariant* under the map $\phi^{(j)}$ if the set is unchanged when $\phi^{(j)}$ is applied to it, in which case from (2.45) and (2.46) we would have that for each $a \in A$,

$$h^{(j)}(a) \in A. \tag{2.48}$$

Lemma 2.7.3. *For any $A \subseteq \{0,1\}^k$ and all integers m and t such that $1 \leq m \leq t \leq k$, the set $\phi^{(t)}(\phi^{(t-1)}(\cdots \phi^{(1)}(A)))$ is invariant under the map $\phi^{(m)}$.*

Proof. For any $A' \subseteq \{0,1\}^k$, we have

$$\phi^{(i)}(\phi^{(i)}(A')) = \phi^{(i)}(A') \quad \forall\ i \in \{1,2,\ldots,k\}. \tag{2.49}$$

The proof of the lemma is by induction on t. For the base case $t = 1$, the proof is clear since $\phi^{(1)}(\phi^{(1)}(A)) = \phi^{(1)}(A)$ from (2.49). Now suppose the lemma is true for all $t < \tau$ (where $\tau \geq 2$). Now suppose $t = \tau$. Let $B = \phi^{(\tau-1)}(\phi^{(\tau-2)}(\cdots \phi^{(1)}(A)))$. Since $\phi^{(\tau)}(\phi^{(\tau)}(B)) = \phi^{(\tau)}(B)$ from (2.49), the lemma is true when $m = t = \tau$. In the following arguments, we take $m < \tau$. From the induction hypothesis, B is invariant under the map $\phi^{(m)}$, i.e.,

$$\phi^{(m)}(B) = B. \tag{2.50}$$

Consider any vector $c \in \phi^{(\tau)}(B)$. From (2.48), we need to show that $h^{(m)}(c) \in \phi^{(\tau)}(B)$. We have the following cases.

$$c_\tau = 1 \quad : \quad c, h^{(\tau)}(c) \in B \qquad \text{[from } c_\tau = 1 \text{ and } c \in \phi^{(\tau)}(B)] \tag{2.51}$$

$$h^{(m)}(c) \in B \qquad \text{[from (2.50) and (2.51)]} \tag{2.52}$$

$$h^{(\tau)}\left(h^{(m)}(c)\right) = h^{(m)}\left(h^{(\tau)}(c)\right) \in B \qquad \text{[from (2.50) and (2.51)]} \tag{2.53}$$

$$h^{(m)}(c) \in \phi^{(\tau)}(B) \qquad \text{[from (2.52) and (2.53)]}$$

$$
\begin{aligned}
c_\tau = 0 \quad : \quad & \exists\, b \in B \text{ such that } h^{(\tau)}(b) = c && [\text{from } c_\tau = 0 \text{ and } c \in \phi^{(\tau)}(B)] && (2.54)\\
& h^{(m)}(b) \in B && [\text{from (2.50) and (2.54)}] && (2.55)\\
& h^{(m)}\left(h^{(\tau)}(b)\right) = h^{(\tau)}\left(h^{(m)}(b)\right) \in \phi^{(\tau)}(B) && [\text{from (2.55)}] && (2.56)\\
& h^{(m)}(c) \in \phi^{(\tau)}(B) && [\text{from (2.54) and (2.56)}]. &&
\end{aligned}
$$

Thus, the lemma is true for $t = \tau$ and the induction argument is complete. □

Let $A_1, A_2, \ldots, A_M \subseteq \{0,1\}^k$ be such that $|A_i| \geq 2^{k-n}$ for each i. Let $U^{(M)} = \prod_{i=1}^{M} A_i$ and extend the definition of $\phi^{(j)}$ in (2.46) to products by

$$\phi^{(j)}(U^{(M)}) = \prod_{i=1}^{M} \phi^{(j)}(A_i).$$

$U^{(M)}$ is said to be *invariant under* $\phi^{(j)}$ if

$$\phi^{(j)}(U^{(M)}) = U^{(M)}.$$

It can be verifed that $U^{(M)}$ is invariant under $\phi^{(j)}$ iff each A_i is invariant under $\phi^{(j)}$. For each $i \in \{1, 2, \ldots, M\}$, let

$$B_i = \phi^{(k)}(\phi^{(k-1)}(\cdots \phi^{(1)}(A_i)))$$

and from (2.47) note that

$$|B_i| = |A_i| \geq 2^{k-n}. \tag{2.57}$$

Let

$$V^{(M)} = \phi^{(k)}(\phi^{(k-1)}(\cdots \phi^{(1)}(U^{(M)}))) = \prod_{i=1}^{M} B_i$$

and recall the definition of the function Q (2.37).

Lemma 2.7.4.

$$\left|Q\left(U^{(M)}\right)\right| \geq \left|Q\left(V^{(M)}\right)\right|.$$

Proof. We begin by showing that

$$\left|Q\left(U^{(M)}\right)\right| \geq \left|Q\left(\phi^{(1)}(U^{(M)})\right)\right|. \tag{2.58}$$

For every $p \in \{0, 1, \ldots, M\}^{k-1}$, let

$$
\begin{aligned}
\varphi(p) &= \left\{r \in Q\left(U^{(M)}\right) : (r_2, \cdots, r_k) = p\right\}\\
\varphi_1(p) &= \left\{s \in Q\left(\phi^{(1)}(U^{(M)})\right) : (s_2, \cdots, s_k) = p\right\}
\end{aligned}
$$

and note that

$$
\begin{aligned}
Q\left(U^{(M)}\right) &= \bigcup_{p \in \{0,1,\ldots,M\}^{k-1}} \varphi(p) && (2.59)\\
Q\left(\phi^{(1)}(U^{(M)})\right) &= \bigcup_{p \in \{0,1,\ldots,M\}^{k-1}} \varphi_1(p) && (2.60)
\end{aligned}
$$

where the two unions are in fact disjoint unions. We show that for every $p \in \{0, 1, \ldots, M\}^{k-1}$,

$$|\varphi(p)| \geq |\varphi_1(p)| \tag{2.61}$$

which by (2.59) and (2.60) implies (2.58).

If $|\varphi_1(p)| = 0$, then (2.61) is trivial. Now consider any $p \in \{0, 1, \ldots, M\}^{k-1}$ such that $|\varphi_1(p)| \geq 1$ and let

$$K_p = \max\{i \ : (i, p_1, \cdots, p_{k-1}) \in \varphi_1(p)\}.$$

Then we have

$$|\varphi_1(p)| \leq K_p + 1. \tag{2.62}$$

Since $(K_p, p_1, \cdots, p_{k-1}) \in \varphi_1(p)$, there exists $(a^{(1)}, a^{(2)}, \ldots, a^{(M)}) \in U^{(M)}$ such that

$$\sum_{i=1}^{M} \hat{\phi}^{(1)}\left(A_i, a^{(i)}\right) = (K_p, p_1, \cdots, p_{k-1}). \tag{2.63}$$

Then from the definition of the map $\hat{\phi}^{(1)}$ in (2.45), there are K_p of the $a^{(i)}$'s from amongst $\{a^{(1)}, a^{(2)}, \ldots, a^{(M)}\}$ such that $a_1^{(i)} = 1$ and $\hat{\phi}^{(1)}\left(A_i, a^{(i)}\right) = a^{(i)}$. Let $I = \{i_1, i_2, \ldots, i_{K_p}\} \subseteq \{1, 2, \ldots, M\}$ be the index set for these vectors and let $\hat{a}^{(i)} = h^{(1)}(a^{(i)})$ for each $i \in I$. Then for each $i \in I$, we have

$$\begin{aligned} a^{(i)} &= \left(1, a_2^{(i)}, \ldots, a_k^{(i)}\right) \ \in \ A_i \\ \hat{a}^{(i)} &= \left(0, a_2^{(i)}, \ldots, a_k^{(i)}\right) \ \in \ A_i \qquad\qquad [\text{from } \hat{\phi}^{(1)}\left(A_i, a^{(i)}\right) = a^{(i)} \text{ and } (2.45)]. \end{aligned}$$

Let

$$R = \left\{ \sum_{i=1}^{M} b^{(i)} \ : \ \begin{array}{ll} b^{(i)} \in \{a^{(i)}, \hat{a}^{(i)}\} & \text{for } i \in I, \\ b^{(i)} = a^{(i)} & \text{for } i \notin I \end{array} \right\} \subseteq \varphi(p). \tag{2.64}$$

From (2.63) and (2.64), for every $r \in R$ we have

$$\begin{aligned} &r_1 \in \{0, 1, \ldots, |I|\}, \\ &r_i = p_i \ \ \forall\, i \in \{2, 3, \ldots, k\} \end{aligned}$$

and thus

$$|R| = |I| + 1 = K_p + 1. \tag{2.65}$$

Hence, we have

$$\begin{aligned} |\varphi(p)| &\geq |R| && [\text{from } (2.64)] \\ &= K_p + 1 && [\text{from } (2.65)] \\ &\geq |\varphi_1(p)| && [\text{from } (2.62)] \end{aligned}$$

and then from (2.59) and (2.60), it follows that

$$\left|Q\left(U^{(M)}\right)\right| \geq \left|Q\left(\phi^{(1)}(U^{(M)})\right)\right|.$$

For any $A \subseteq \{0,1\}^k$ and any $j \in \{1,2,\ldots,k\}$, we know that $\left|\phi^{(j)}(A)\right| \subseteq \{0,1\}^k$. Thus, the same arguments as above can be repeated to show that

$$\begin{aligned}\left|Q\left(\phi^{(1)}(U^{(M)})\right)\right| &\geq \left|Q\left(\phi^{(2)}(\phi^{(1)}(U^{(M)}))\right)\right| \\ &\geq \left|Q\left(\phi^{(3)}(\phi^{(2)}(\phi^{(1)}(U^{(M)})))\right)\right| \\ &\quad\vdots \\ &\geq \left|Q\left(\phi^{(k)}(\phi^{(k-1)}(\cdots\phi^{(1)}(U^{(M)})))\right)\right| \\ &= \left|Q\left(V^{(M)}\right)\right|.\end{aligned}$$

□

For any $s, r \in \mathbb{Z}^k$, we say that $s \leq r$ if $s_l \leq r_l$ for every $l \in \{1,2,\ldots,k\}$.

Lemma 2.7.5. *Let $p \in Q\big(V^{(M)}\big)$. If $q \in \{0,1,\ldots,M\}^k$ and $q \leq p$, then $q \in Q\big(V^{(M)}\big)$.*

Proof. Since $q \leq p$, it can be obtained by iteratively subtracting 1 from the components of p, i.e., there exist $t \geq 0$ and $i_1, i_2, \ldots, i_t \in \{1,2,\ldots,k\}$ such that

$$q = h^{(i_1)}\left(h^{(i_2)}\left(\cdots\left(h^{(i_t)}(p)\right)\right)\right).$$

Consider any $i \in \{1,2,\ldots,k\}$. We show that $h^{(i)}(p) \in Q\big(V^{(M)}\big)$, which implies by induction that $q \in Q\big(V^{(M)}\big)$. If $p_i = 0$, then $h^{(i)}(p) = p$ and we are done. Suppose that $p_i > 0$. Since $p \in Q\big(V^{(M)}\big)$, there exists $b^{(j)} \in B_j$ for every $j \in \{1,2,\ldots,M\}$ such that

$$p = \sum_{j=1}^{M} b^{(j)}$$

and $b_i^{(m)} = 1$ for some $m \in \{1,2,\ldots,M\}$. From Lemma 2.7.3, $V^{(M)}$ is invariant under $\phi^{(i)}$ and thus from (2.48), $h^{(i)}(b^{(m)}) \in B_m$ and

$$h^{(i)}(p) = \sum_{j=1}^{m-1} b^{(j)} + h^{(i)}(b^{(m)}) + \sum_{j=m+1}^{M} b^{(j)}$$

is an element of $Q\big(V^{(M)}\big)$. □

The lemma below is presented in [Ayaso 07] without proof, as the proof is straightforward.

Lemma 2.7.6. *For all positive integers k, n, M, and $\delta \in (0,1)$,*

$$\min_{\substack{0 \leq m_i \leq M, \\ \sum_{i=1}^{k} m_i \geq \delta M k}} \prod_{i=1}^{k} (1+m_i) \geq (M+1)^{\delta k}. \tag{2.66}$$

For any $a \in \{0,1\}^k$, let $|a|_H$ denote the Hamming weight of a, i.e., the number of non-zero components of a. The next lemma uses the function γ defined in (2.38).

Lemma 2.7.7.

$$\left|Q\left(V^{(M)}\right)\right| \geq (M+1)^{\gamma(k/n)k}.$$

Proof. Let $\delta = \gamma(k/n)$. The number of distinct elements in $\{0,1\}^k$ with Hamming weight at most $\lfloor \delta k \rfloor$ equals

$$\begin{aligned}\sum_{j=0}^{\lfloor \delta k \rfloor} \binom{k}{j} &\leq 2^{k\mathcal{H}(\delta)} && \text{[from [Hoeffding 63, p.15, Theorem 1]]}\\ &= 2^{(k-n)/2} && \text{[from (2.38)].}\end{aligned}$$

For each $i \in \{1,2,\ldots,M\}$, $|B_i| \geq 2^{k-n}$ from (2.57) and hence there exists $b^{(i)} \in B_i$ such that $\left|b^{(i)}\right|_H \geq \delta k$. Let

$$p = \sum_{i=1}^{M} b^{(i)} \ \in Q\left(V^{(M)}\right).$$

It follows that $p_j \in \{0,1,2,\ldots,M\}$ for every $j \in \{1,2,\ldots,k\}$, and

$$\sum_{j=1}^{k} p_j = \sum_{i=1}^{M} \left|b^{(i)}\right|_H \geq \delta M k. \tag{2.67}$$

The number of vectors q in $\{0,1,\ldots,M\}^k$ such that $q \preceq p$ equals $\prod_{j=1}^{k}(1+p_j)$, and from Lemma 2.7.5, each such vector is also in $Q\left(V^{(M)}\right)$. Therefore,

$$\begin{aligned}\left|Q\left(V^{(M)}\right)\right| &\geq \prod_{j=1}^{k}(1+p_j)\\ &\geq (M+1)^{\delta k} && \text{[from (2.67) and Lemma 2.7.6].}\end{aligned}$$

Since $\delta = \gamma(k/n)$, the result follows. □

This chapter, in full, is a reprint of the material as it appears in: R. Appuswamy, M. Franceschetti, N. Karamchandani, and K. Zeger, "Network coding for computing: cut-set bounds," *IEEE Transactions on Information Theory*, vol. 57 (2), pp. 1015-1030, Feb. 2011. The dissertation author was the primary investigator of this paper.

Chapter 3

Linear Codes, Target Function Classes, and Network Computing Capacity

3.1 Introduction

Network coding concerns networks where each receiver demands a subset of messages generated by the source nodes and the objective is to satisfy the receiver demands at the maximum possible throughput rate. Accordingly, research efforts have studied coding gains over routing [Ahlswede 00, Harvey 06, Harvey 04], whether linear codes are sufficient to achieve the capacity [Li 03, Dougherty 05, Dougherty 08, Koetter 03], and cut-set upper bounds on the capacity and the tightness of such bounds [Harvey 06, Harvey 04, Lehman 03].

Network computing, on the other hand, considers a more general problem in which each receiver node demands a target function of the source messages [Giridhar 05, Appuswamy 11c, Kowshik 09, Rai 09, Ramamoorthy 08, Nazer 07]. Most problems in network coding are applicable to network computing as well. Network computing problems arise in various networks including sensor networks and vehicular networks.

In [Appuswamy 11c], a network computing model was proposed where the network is modeled by a directed, acyclic graph with independent, noiseless links. The sources generate independent messages and a single receiver node computes a target function f of these messages. The objective is to characterize the maximum rate of computation, that is, the maximum number of times f can be computed per network usage. Each node in the network sends out symbols on its out-edges which are arbitrary, but fixed, functions of the symbols received on its in-edges and any messages generated at the node. In linear network computing, this encoding

is restricted to be linear operations. Existing techniques for computing in networks use routing, where the codeword sent out by a node consists of symbols either received by that node, or generated by the node if it is a source (e.g. [Paek 09]).

In network coding, it is known that linear codes are sufficient to achieve the coding capacity for multicast networks [Ahlswede 00], but they are not sufficient in general to achieve the coding capacity for non-multicast networks [Dougherty 05]. In network computing, it is known that when multiple receiver nodes demand a scalar linear target function of the source messages, linear network codes may not be sufficient in general for solvability [Rai 10a]. However, it has been shown that for single-receiver networks, linear coding is sufficient for solvability when computing a scalar linear target function [Rai 09]. Analogous to the coding capacity for network coding, the notion of computing capacity was defined for network computing in [Giridhar 05] and is the supremum of achievable rates of computing the network's target function.

One fundamental objective in the present paper is to understand the performance of linear network codes for computing different types of target functions. Specifically, we compare the linear computing capacity with that of the (nonlinear) computing capacity and the routing computing capacity for various different classes of target functions in single-receiver networks. Such classes include reducible, injective, semi-injective, and linear target functions over finite fields. Informally, a target function is semi-injective if it uniquely maps at least one of its inputs, and a target function is reducible if it can be computed using a linear transformation followed by a function whose domain has a reduced dimension. Computing capacity bounds and achievability are given with respect to the target function classes studied for network codes that use routing, linear coding, or nonlinear coding.

Our specific contributions will be summarized next.

3.1.1 Contributions

Section 3.2 gives many of the formal definitions used in the paper (e.g. target function classes and computing capacity types). We show that routing messages through the intermediate nodes in a network forces the receiver to obtain all the messages even though only a function of the messages is required (Theorem 3.2.10), and we bound the computing capacity gain of using nonlinear versus routing codes (Theorem 3.2.12).

In Section 3.3, we demonstrate that the performance of optimal linear codes may depend on how 'linearity' is defined (Theorem 3.3.2). Specifically, we show that the linear computing capacity of a network varies depending on which ring linearity is defined over on the source alphabet.

In Sections 3.4 and 3.5, we study the computing capacity gain of using linear coding over routing, and nonlinear coding over linear coding. In particular, we study various classes of target functions, including injective, semi-injective, reducible, and linear. The relationships between these classes is illustrated in Figure 3.1.

Section 3.4 studies linear coding for network computing. We show that if a target function is not reducible, then the linear computing capacity and routing computing capacity are equal whenever the source alphabet is a finite field (Theorem 3.4.8); the same result also holds for semi-injective target functions over rings. We also show that whenever a target function is injective, routing obtains the full computing capacity of a network (Theorem 3.4.9), although whenever a target function is neither reducible nor injective, there exists a network such that the computing capacity is larger than the linear computing capacity (Theorem 3.4.11). Thus for non-injective target functions that are not reducible, any computing capacity gain of using coding over routing must be obtained through nonlinear coding. This result is tight in the sense that if a target function is reducible, then there always exists a network where the linear computing capacity is larger than the routing capacity (Theorem 3.4.12). We also show that there exists a reducible target function and a network whose computing capacity is strictly greater than its linear computing capacity, which in turn is strictly greater than its routing computing capacity. (Theorem 3.4.14).

Section 3.5 focuses on computing linear target functions over finite fields. We characterize the linear computing capacity for linear target functions over finite fields in arbitrary networks (Theorem 3.5.6). We show that linear codes are sufficient for linear target functions and we upper bound the computing capacity gain of coding (linear or nonlinear) over routing (Theorem 3.5.7). This upper bound is shown to be achievable for every linear target function and an associated network, in which case the computing capacity is equal to the routing computing capacity times the number of network sources (Theorem 3.5.8).

Finally, Section 3.6 studies an illustrative example for the computing problem, namely the reverse butterfly network – obtained by reversing the direction of all the edges in the multicast butterfly network (the butterfly network studied in [Ahlswede 00] illustrated the capacity gain of network coding over routing). For this network and the arithmetic sum target function, we evaluate the routing and linear computing capacity (Theorem 3.6.1) and the computing capacity (Theorem 3.6.3). We show that the latter is strictly larger than the first two, which are equal to each other. No network with such properties is presently known for network coding. Among other things, the reverse butterfly network also illustrates that the computing capacity can be a function of the coding alphabet (i.e. the domain of the target function f). In contrast, for network coding, the coding capacity and routing capacity are known to be independent of the coding alphabet used [Cannons 06].

Our main results are summarized in Table 3.1.

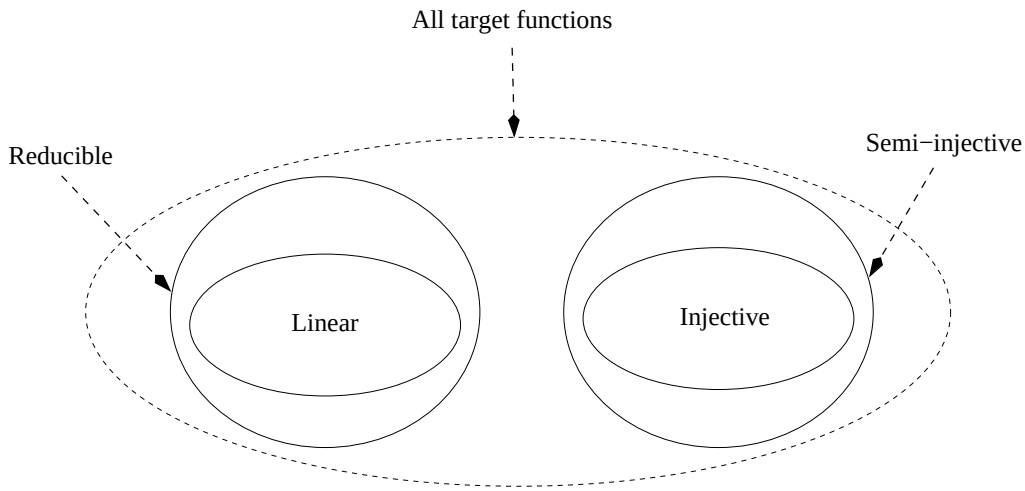


Figure 3.1: Decomposition of the space of all target functions into various classes.

Table 3.1: Summary of our main results for certain classes of target functions. The quantities $\mathcal{C}_{\text{cod}}(\mathcal{N}, f)$, $\mathcal{C}_{\text{lin}}(\mathcal{N}, f)$, and $\mathcal{C}_{\text{rout}}(\mathcal{N}, f)$ denote the computing capacity, linear computing capacity, and routing computing capacity, respectively, for a network $\mathcal{N}$ with s sources and target function f. The columns labeled f and $\mathcal{A}$ indicate contraints on the target function f and the source alphabet $\mathcal{A}$, respectively.

Result	f	$\mathcal{A}$	Location
$\forall f\ \forall \mathcal{N}\ \mathcal{C}_{\text{lin}}(\mathcal{N}, f) = \mathcal{C}_{\text{rout}}(\mathcal{N}, f)$	non-reducible	field	Theorem 3.4.8
	semi-injective	ring	
$\forall f\ \forall \mathcal{N}\ \mathcal{C}_{\text{cod}}(\mathcal{N}, f) = \mathcal{C}_{\text{rout}}(\mathcal{N}, f)$	injective		Theorem 3.4.9
$\forall f\ \exists \mathcal{N}\ \mathcal{C}_{\text{cod}}(\mathcal{N}, f) > \mathcal{C}_{\text{lin}}(\mathcal{N}, f)$	non-injective & non-reducible	field	Theorem 3.4.11
$\forall f\ \exists \mathcal{N}\ \mathcal{C}_{\text{lin}}(\mathcal{N}, f) > \mathcal{C}_{\text{rout}}(\mathcal{N}, f)$	reducible	ring	Theorem 3.4.12
$\exists f\ \exists \mathcal{N}\ \mathcal{C}_{\text{cod}}(\mathcal{N}, f) > \mathcal{C}_{\text{lin}}(\mathcal{N}, f) > \mathcal{C}_{\text{rout}}(\mathcal{N}, f)$	reducible		Theorem 3.4.14
$\forall f\ \forall \mathcal{N}\ \mathcal{C}_{\text{cod}}(\mathcal{N}, f) = \mathcal{C}_{\text{lin}}(\mathcal{N}, f) \leq s\ \mathcal{C}_{\text{rout}}(\mathcal{N}, f)$	linear	field	Theorem 3.5.7
$\forall f\ \exists \mathcal{N}\ \mathcal{C}_{\text{lin}}(\mathcal{N}, f) = s\ \mathcal{C}_{\text{rout}}(\mathcal{N}, f)$	linear	field	Theorem 3.5.8
$\exists f\ \exists \mathcal{N}\ \mathcal{C}_{\text{cod}}(\mathcal{N}, f)$ is irrational	arithmetic sum		Theorem 3.6.3

3.2 Network model and definitions

In this paper, a *network* $\mathcal{N} = (G, S, \rho)$ consists of a finite, directed acyclic multigraph $G = (\mathcal{V}, \mathcal{E})$, a set $S = \{\sigma_1, \ldots, \sigma_s\} \subseteq \mathcal{V}$ of s distinct *source nodes* and a single *receiver* $\rho \in \mathcal{V}$. We assume that $\rho \notin S$, and that the graph[1] G contains a directed path from every node in $\mathcal{V}$ to the receiver ρ. For each node $u \in \mathcal{V}$, let $\mathcal{E}_i(u)$ and $\mathcal{E}_o(u)$ denote the in-edges and out-edges of u respectively. We assume (without loss of generality) that if a network node has no in-edges, then it is a source node. If $e = (u, v) \in \mathcal{E}$, we will use the notation *head(e)* $= u$ and *tail(e)* $= v$.

An *alphabet* is a finite set of size at least two. Throughout this paper, $\mathcal{A}$ will denote a *source alphabet* and $\mathcal{B}$ will denote a *receiver alphabet*. For any positive integer m, any vector $x \in \mathcal{A}^m$, and any $i \in \{1, 2, \ldots, m\}$, let x_i denote the i-th component of x. For any index set $I = \{i_1, i_2, \ldots, i_q\} \subseteq \{1, 2, \ldots, m\}$ with $i_1 < i_2 < \ldots < i_q$, let x_I denote the vector $(x_{i_1}, x_{i_2}, \ldots, x_{i_q}) \in \mathcal{A}^{|I|}$. Sometimes we view $\mathcal{A}$ as an algebraic structure such as a ring, i.e., with multiplication and addition. Throughout this paper, vectors will always be taken to be row vectors. Let denote a finite field of order q. A superscript t will denote the transpose for vectors and matrices.

3.2.1 Target functions

For a given network $\mathcal{N} = (G, S, \rho)$, we use s throughout the paper to denote the number $|S|$ of receivers in $\mathcal{N}$. For given network $\mathcal{N}$, a *target function* is a mapping

$$f : \mathcal{A}^s \longrightarrow \mathcal{B}.$$

The goal in network computing is to compute f at the receiver ρ, as a function of the source messages. We will assume that all target functions depend on all the network sources (i.e. a target function cannot be a constant function of any one of its arguments). Some example target functions that will be referenced are listed in Table 3.2.

Table 3.2: Definitions of some target functions.

Target function f	Alphabet $\mathcal{A}$	$f(x_1, \ldots, x_s)$
identity	arbitrary	$(x_1, \ldots, x_s)$
arithmetic sum	$\{0, 1, \ldots, q-1\}$	$x_1 + x_2 + \cdots + x_s$
mod r sum	$\{0, 1, \ldots, q-1\}$	$x_1 \oplus x_2 \oplus \ldots \oplus x_s$
linear	any ring	$a_1 x_1 + a_2 x_2 + \ldots + a_s x_s$
maximum	any ordered set	$\max\{x_1, \ldots, x_s\}$

[1] Throughout the remainder of the paper, we use "graph" to mean a multigraph, and in the context of network computing we use "network" to mean a single-receiver network.

Definition 3.2.1. Let alphabet $\mathcal{A}$ be a ring. A target function $f : \mathcal{A}^s \longrightarrow \mathcal{B}$ is said to be *reducible* if there exists an integer λ satisfying $\lambda < s$, an $s \times \lambda$ matrix T with elements in $\mathcal{A}$, and a map $g : \mathcal{A}^\lambda \longrightarrow \mathcal{B}$ such that for all $x \in \mathcal{A}^s$,

$$g(xT) = f(x). \tag{3.1}$$

Reducible target functions are not injective, since, for example, if x and y are distinct elements of the null-space of T, then

$$f(x) = g(xT) = g(0) = g(yT) = f(y).$$

Example 3.2.2. Suppose the alphabet is $\mathcal{A} =$ and the target function is

$$f :^3 \longrightarrow \{0, 1\},$$

where

$$f(x) = (x_1 + x_2)x_3.$$

Then, by choosing $\lambda = 2$,

$$T = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix},$$

and $g(y_1, y_2) = y_1 y_2$, we get

$$\begin{aligned} g(xT) &= g(x_1 + x_2, x_3) \\ &= (x_1 + x_2)x_3 \\ &= f(x). \end{aligned}$$

Thus the target function f is reducible.

Example 3.2.3. The notion of reducibility requires that for a target function $f : \mathcal{A}^s \longrightarrow \mathcal{B}$, the set $\mathcal{A}$ must be a ring. If we impose any ring structure to the domains of the identity, arithmetic sum, maximum, and minimum target functions, then these can be shown (via our Example 3.4.2 and Lemma 3.4.3) to be non-reducible.

3.2.2 Network computing and capacity

Let k and n be positive integers. Given a network $\mathcal{N}$ with source set S and alphabet $\mathcal{A}$, a *message generator* is any mapping

$$\alpha \ : \ S \longrightarrow \mathcal{A}^k.$$

For each source $\sigma_i \in S$, $\alpha(\sigma_i)$ is called a *message vector* and its components

$$\alpha(\sigma_i)_1, \ldots, \alpha(\sigma_i)_k$$

are called *messages*[2].

Definition 3.2.4. A (k, n) *network code in a network* $\mathcal{N}$ consists of the following:

(i) *Encoding functions* $h^{(e)}$, for every out-edge $e \in \mathcal{E}_o(v)$ of every node $v \in \mathcal{V} - \rho$, of the form:

$$h^{(e)} : \left(\prod_{\hat{e} \in \mathcal{E}_i(v)} \mathcal{A}^n \right) \times \mathcal{A}^k \longrightarrow \mathcal{A}^n \quad \text{if } v \text{ is a source node}$$
$$h^{(e)} : \prod_{\hat{e} \in \mathcal{E}_i(v)} \mathcal{A}^n \longrightarrow \mathcal{A}^n \qquad \text{otherwise.}$$

(ii) A *decoding function* ψ of the form:

$$\psi : \prod_{\hat{e} \in \mathcal{E}_i(v)} \mathcal{A}^n \longrightarrow \mathcal{B}^k.$$

Furthermore, given a (k, n) network code, every edge $e \in \mathcal{E}$ carries a vector z_e of at most n alphabet symbols[3], which is obtained by evaluating the encoding function $h^{(e)}$ on the set of vectors carried by the in-edges to the node and the node's message vector if the node is a source. The objective of the receiver is to compute the target function f of the source messages, for any arbitrary message generator α. More precisely, the receiver constructs a vector of k alphabet symbols, such that for each $i \in \{1, 2, \ldots, k\}$, the i-th component of the receiver's computed vector equals the value of the desired target function f, applied to the i-th components of the source message vectors, for any choice of message generator α.

Definition 3.2.5. Suppose in a network $\mathcal{N}$, the in-edges of the receiver are $e_1, e_2, \ldots, e_{|\mathcal{E}_i(\rho)|}$. A (k, n) network code is said to *compute f in $\mathcal{N}$* if for each $j \in \{1, 2, \ldots, k\}$, and for each message generator α, the decoding function satisfies

$$\psi\left(z_{e_1}, \cdots, z_{e_{|\mathcal{E}_i(\rho)|}}\right)_j = f\Big(\left(\alpha(\sigma_1)_j, \cdots, \alpha(\sigma_s)_j\right)\Big). \tag{3.2}$$

If there exists a (k, n) code that computes f in $\mathcal{N}$, then the rational number k/n is said to be an *achievable computing rate*.

In the network coding literature, one definition of the *coding capacity* of a network is the supremum of all achievable coding rates [Cannons 06]. We use an analogous definition for the computing capacity.

Definition 3.2.6. The *computing capacity* of a network $\mathcal{N}$ with respect to a target function f is

$$\mathcal{C}_{\text{cod}}(\mathcal{N}, f) \;=\; \sup\left\{\frac{k}{n} \;:\; \exists\, (k, n) \text{ network code that computes } f \text{ in } \mathcal{N}\right\}.$$

[2] For simplicity we assume each source has associated with it exactly one message vector, but all of the results in this paper can readily be extended to the more general case.

[3] By default, we assume that edges carry exactly n symbols.

The notion of linear codes in networks is most often studied with respect to finite fields. Here we will sometimes use more general ring structures.

Definition 3.2.7. Let alphabet $\mathcal{A}$ be a ring. A (k, n) network code in a network $\mathcal{N}$ is said to be a *linear network code (over $\mathcal{A}$)* if the encoding functions are linear over $\mathcal{A}$.

Definition 3.2.8. The *linear computing capacity* of a network $\mathcal{N}$ with respect to target function f is

$$\mathcal{C}_{\text{lin}}(\mathcal{N}, f) \;=\; \sup\left\{\frac{k}{n} \;:\; \exists\ (k,n) \text{ linear network code that computes } f \text{ in } \mathcal{N}\right\}.$$

The *routing computing capacity* $\mathcal{C}_{rout}(\mathcal{N}, f)$ is defined similarly by restricting the encoding functions to routing. We call the quantity $\mathcal{C}_{\text{cod}}(\mathcal{N}, f) - \mathcal{C}_{\text{lin}}(\mathcal{N}, f)$ the *computing capacity gain* of using nonlinear coding over linear coding. Similar "gains", such as, $\mathcal{C}_{\text{cod}}(\mathcal{N}, f) - \mathcal{C}_{\text{rout}}(\mathcal{N}, f)$ and $\mathcal{C}_{\text{lin}}(\mathcal{N}, f) - \mathcal{C}_{\text{rout}}(\mathcal{N}, f)$ are defined.

Note that Definition 3.2.7 allows linear codes to have nonlinear decoding functions. In fact, since the receiver alphabet $\mathcal{B}$ need not have any algebraic structure to it, linear decoding functions would not make sense in general. We do, however, examine a special case where $\mathcal{B} = \mathcal{A}$ and the target function is linear, in which case we show that linear codes with linear decoders can be just as good as linear codes with nonlinear decoders (Theorem 3.5.7).

Definition 3.2.9. A set of edges $C \subseteq \mathcal{E}$ in network $\mathcal{N}$ is said to *separate* sources $\sigma_{m_1}, \ldots, \sigma_{m_d}$ from the receiver ρ, if for each $i \in \{1, 2, \ldots, d\}$, every directed path from σ_{m_i} to ρ contains at least one edge in C. Define

$$I_C = \{i : C \text{ separates } \sigma_i \text{ from the receiver}\}.$$

The set C is said to be a *cut* in $\mathcal{N}$ if it separates at least one source from the receiver (i.e. $|I_C| \geq 1$). We denote by $\Lambda(\mathcal{N})$ the collection of all cuts in $\mathcal{N}$.

Since I_C is the number of sources disconnected by C and there are s sources, we have

$$|I_C| \leq s. \tag{3.3}$$

For network coding with a single receiver node and multiple sources (where the receiver demands all the source messages), routing is known to be optimal [Lehman 03]. Let $\mathcal{C}_{\text{rout}}(\mathcal{N})$ denote the routing capacity of the network $\mathcal{N}$, or equivalently the routing computing capacity for computing the identity target function. It was observed in [Lehman 03, Theorem 4.2] that for any single-receiver network $\mathcal{N}$,

$$\mathcal{C}_{\text{rout}}(\mathcal{N}) = \min_{C \in \Lambda(\mathcal{N})} \frac{|C|}{|I_C|}. \tag{3.4}$$

The following theorem shows that if the intermediate nodes in a network are restricted to perform routing, then in order to compute a target function the receiver is forced to obtain all the source messages. This fact motivates the use of coding for computing functions in networks.

Theorem 3.2.10. *If $\mathcal{N}$ is a network with target function f, then*

$$\mathcal{C}_{rout}(\mathcal{N}, f) = \mathcal{C}_{rout}(\mathcal{N}) \,.$$

Proof. Since any routing code that computes the identity target function can be used to compute any target function f, we have

$$\mathcal{C}_{\text{rout}}(\mathcal{N}, f) \geq \mathcal{C}_{\text{rout}}(\mathcal{N}) \,.$$

Conversely, it is easy to see that every component of every source message must be received by ρ in order to compute f, so

$$\mathcal{C}_{\text{rout}}(\mathcal{N}, f) \leq \mathcal{C}_{\text{rout}}(\mathcal{N}) \,.$$

□

Theorem 3.2.12 below gives a general upper bound on how much larger the computing capacity can be relative to the routing computing capacity. It will be shown later, in Theorem 3.5.7, that for linear target functions over finite fields, the bound in Theorem 3.2.12 can be tightened by removing the logarithm term.

Lemma 3.2.11. *If $\mathcal{N}$ is network with a target function $f : \mathcal{A}^s \longrightarrow \mathcal{B}$, then*

$$\mathcal{C}_{cod}(\mathcal{N}, f) \leq (\log_2 |\mathcal{A}|) \min_{C \in \Lambda(\mathcal{N})} |C| \,.$$

Proof. Using [Appuswamy 11c, Theorem II.1], one finds the term min-cut$(\mathcal{N}, f)$ defined in [Appuswamy 11c, Equation (3)] in terms of a quantity $R_{I_C,f}$, which in turn is defined in [Appuswamy 11c, Definition 1.5]. Since target functions are restricted to not being constant functions of any of their arguments, we have $R_{I_C,f} \geq 2$, from which the result follows. □

Theorem 3.2.12. *If $\mathcal{N}$ is network with a target function $f : \mathcal{A}^s \longrightarrow \mathcal{B}$, then*

$$\mathcal{C}_{cod}(\mathcal{N}, f) \leq s \, (\log_2 |\mathcal{A}|) \, \mathcal{C}_{rout}(\mathcal{N}, f)$$

Proof.

$$\begin{aligned} \mathcal{C}_{\text{cod}}(\mathcal{N}, f) &\leq (\log_2 |\mathcal{A}|) \min_{C \in \Lambda(\mathcal{N})} |C| && \text{[from Lemma 3.2.11]} \\ &\leq s \, (\log_2 |\mathcal{A}|) \, \mathcal{C}_{\text{rout}}(\mathcal{N}, f) \,. && \text{[from (3.3), (3.4), and Theorem 3.2.10]} \end{aligned}$$

□

3.3 Linear coding over different ring alphabets

Whereas the size of a finite field characterizes the field, there are, in general, different rings of the same size, so one must address whether the linear computing capacity of a network might depend on which ring is chosen for the alphabet. In this section, we illustrate this possibility with a specific computing problem.

Let $\mathcal{A} = \{a_0, a_1, a_2, a_3\}$ and let $f : \mathcal{A}^2 \longrightarrow \{0, 1, 2\}$ be as defined in Table 3.3. We

Table 3.3: Definition of the 4-ary map f.

f	a_0	a_1	a_2	a_3
a_0	0	1	1	2
a_1	1	0	2	1
a_2	1	2	0	1
a_3	2	1	1	0

consider different rings R of size 4 for $\mathcal{A}$ and evaluate the linear computing capacity of the network $\mathcal{N}_1$ shown in Figure 3.2 with respect to the target function f. Specifically, we let R be either the ring $\mathbb{Z}_4$ of integers modulo 4 or the product ring $\mathbb{Z}_2 \times \mathbb{Z}_2$ of 2-dimensional binary vectors. Denote the linear computing capacity here by

$$\mathcal{C}_{\text{lin}}(\mathcal{N}_1)^R = \sup\left\{\frac{k}{n} \ : \ \exists\ (k,n)\ R\text{-linear code that computes } f \text{ in } \mathcal{N}\right\}.$$

The received vector z at ρ can be viewed as a function of the source vectors generated at σ_1

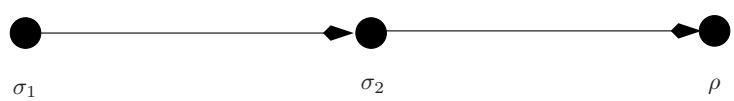


Figure 3.2: Network $\mathcal{N}_1$ has two sources σ_1 and σ_2 and a receiver ρ.

and σ_2. For any (k, n) R-linear code, there exist $k \times n$ matrices M_1 and M_2 such that z can be written as

$$z(\alpha(\sigma_1), \alpha(\sigma_2)) = \alpha(\sigma_1)\, M_1 + \alpha(\sigma_2)\, M_2. \tag{3.5}$$

Let $m_{i,1}, \cdots, m_{i,k}$ denote the row vectors of M_i, for $i \in \{1, 2\}$.

Lemma 3.3.1. *Let $\mathcal{A}$ be the ring $\mathbb{Z}_4$ and let $f : \mathcal{A}^2 \longrightarrow \{0, 1, 2\}$ be the target function shown in Table 3.3, where $a_i = i$, for each i. If a (k, n) linear code over $\mathcal{A}$ computes f in $\mathcal{N}_1$ and ρ receives a zero vector, then $\alpha(\sigma_1) = \alpha(\sigma_2) \in \{0, 2\}^k$.*

Proof. If $\alpha(\sigma_1) = \alpha(\sigma_2) = 0$, then ρ receives a 0 by (3.5) and must decode a 0 since $f((0,0)) = 0$ (from Table 3.3). Thus, ρ always decodes a 0 upon receiving a 0. But $f((x_1, x_2)) = 0$ if and only if $x_1 = x_2$ (from Table 3.3), so whenever ρ receives a 0, the source messages satisfy $\alpha(\sigma_1) = \alpha(\sigma_2)$.

Now suppose, contrary to the lemma's assertion, that there exist messages $\alpha(\sigma_1)$ and $\alpha(\sigma_2)$ such that $z(\alpha(\sigma_1), \alpha(\sigma_2)) = 0$ and $\alpha(\sigma_1)_j \notin \{0, 2\}$ for some $j \in \{1, 2, \cdots, k\}$. Since $\alpha(\sigma_1)_j$ is invertible in $\mathbb{Z}_4$ (it is either 1 or 3), we have from (3.5) that

$$m_{1,j} = \sum_{\substack{i=1 \\ i \neq j}}^{k} -\alpha(\sigma_1)_j^{-1}\, \alpha(\sigma_1)_i\, m_{1,i} + \sum_{i=1}^{k} -\alpha(\sigma_1)_j^{-1}\, \alpha(\sigma_2)_i\, m_{2,i} \tag{3.6}$$

$$= y^{(1)} M_1 + y^{(2)} M_2 \tag{3.7}$$

where $y^{(1)}$ and $y^{(2)}$ are k-dimensional vectors defined by

$$y_i^{(1)} = \begin{cases} -\alpha(\sigma_1)_j^{-1}\, \alpha(\sigma_1)_i & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases}$$

$$y_i^{(2)} = -\alpha(\sigma_1)_j^{-1}\, \alpha(\sigma_2)_i\,. \tag{3.8}$$

Also, define the k-dimensional vector x by

$$x_i = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j. \end{cases} \tag{3.9}$$

We have from (3.5) that $z(x, 0) = m_{1,j}$ and from (3.5) and (3.7) that $z(y^{(1)}, y^{(2)}) = m_{1,j}$. Thus, in order for the code to compute f, we must have $f(x_j, 0) = f(y_j^{(1)}, y_j^{(2)})$. But $f(x_j, 0) = f(1, 0) = 1$ and

$$\begin{aligned} f(y_j^{(1)}, y_j^{(2)}) &= f(0, -\alpha(\sigma_1)_j^{-1}\, \alpha(\sigma_2)_j) \\ &= f(0, -\alpha(\sigma_1)_j^{-1}\, \alpha(\sigma_1)_j) && [\text{from } \alpha(\sigma_1) = \alpha(\sigma_2)] \\ &= f(0, -1) \\ &= f(0, 3) && [\text{from } 3 = -1 \text{ in } \mathbb{Z}_4] \\ &= 2 && [\text{from Table 3.3}], \end{aligned}$$

a contradiction. Thus, $\alpha(\sigma_1) \in \{0, 2\}^k$. □

Theorem 3.3.2. *The network $\mathcal{N}_1$ in Figure 3.2 with alphabet $\mathcal{A} = \{a_0, a_1, a_2, a_3\}$ and target function $f : \mathcal{A}^2 \longrightarrow \{0, 1, 2\}$ shown in Table 3.3, satisfies*

$$\mathcal{C}_{lin}(\mathcal{N}_1, f)^{\mathbb{Z}_4} \leq \frac{2}{3}$$

$$\mathcal{C}_{lin}(\mathcal{N}_1, f)^{\mathbb{Z}_2 \times \mathbb{Z}_2} = 1.$$

(For $\mathcal{A} = \mathbb{Z}_4$, we identify $a_i = i$, for each i, and for $\mathcal{A} = \mathbb{Z}_2 \times \mathbb{Z}_2$, we identify each a_i with the 2-bit binary representation of i.)

Proof. Consider a (k,n) $\mathbb{Z}_2 \times \mathbb{Z}_2$-linear code that computes f. From (3.5), we have $z(x,0) = 0$ whenever $xM_1 = 0$. Since $f((0,0)) \neq f((x_i,0))$ (whenever $x_i \neq 0$), it must therefore be the case that $xM_1 = 0$ only when $x = 0$, or in other words, the rows of M_1 must be independent, so $n \geq k$. Thus,

$$\mathcal{C}_{\text{lin}}(\mathcal{N}, f)^{\mathbb{Z}_2 \times \mathbb{Z}_2} \leq 1. \tag{3.10}$$

Now suppose that $\mathcal{A}$ is the ring $\mathbb{Z}_2 \times \mathbb{Z}_2$ where, $a_0 = (0,0)$, $a_1 = (0,1)$, $a_2 = (1,0)$, and $a_3 = (1,1)$ and let $\oplus$ denote the addition over $\mathcal{A}$. For any $x \in \mathcal{A}^2$, the value $f(x)$, as defined in Table 3.3, is seen to be the Hamming distance between x_1 and x_2. If $k = n = 1$ and $M_1 = M_2 = [a_3]$ (i.e., the 1×1 identity matrix), then ρ receives $x_1 \oplus x_2$ from which f can be computed by summing its components. Thus, a computing rate of $k/n = 1$ is achievable. From (3.10), it then follows that

$$\mathcal{C}_{\text{lin}}(\mathcal{N}, f)^{\mathbb{Z}_2 \times \mathbb{Z}_2} = 1.$$

We now prove that $\mathcal{C}_{\text{lin}}(\mathcal{N}, f)^{\mathbb{Z}_4} \leq 2/3$. Let $\mathcal{A}$ denote the ring $\mathbb{Z}_4$ where $a_i = i$ for $0 \leq i \leq 3$. For a given (k,n) linear code over $\mathcal{A}$ that computes f, the n-dimensional vector received by ρ can be written as in (3.5). Let $\mathcal{K}$ denote the collection of all message vector pairs $(\alpha(\sigma_1), \alpha(\sigma_2))$ such that $z(\alpha(\sigma_1), \alpha(\sigma_2)) = 0$. Define the $2k \times n$ matrix

$$M = \begin{bmatrix} M_1 \\ M_2 \end{bmatrix}$$

and notice that $\mathcal{K} = \{y \in \mathcal{A}^{2k} : yM = 0\}$. Then,

$$\begin{aligned}
4^n &= |\mathcal{A}|^n \\
&\geq \left|\{yM : y \in \mathcal{A}^{2k}\}\right| && [\text{from } y \in \mathcal{A}^{2k} \Longrightarrow yM \in \mathcal{A}^n] \\
&\geq \frac{|\mathcal{A}|^{2k}}{|\mathcal{K}|} && [\text{from } y^{(1)}, y^{(2)} \in \mathcal{A}^{2k} \text{ and } y^{(1)}M = y^{(2)}M \Longrightarrow y^{(1)} - y^{(2)} \in \mathcal{K}] \\
&\geq \frac{|\mathcal{A}|^{2k}}{2^k} && [\text{from Lemma 3.3.1}] \\
&= 4^{3k/2}. && [\text{from } |\mathcal{A}| = 4]
\end{aligned}$$

Thus, $k/n \leq 2/3$, so $\mathcal{C}_{\text{lin}}(\mathcal{N}_1, f)^{\mathbb{Z}_4} \leq \frac{2}{3}$. $\square$

3.4 Linear network codes for computing target functions

Theorem 3.2.10 showed that if intermediate network nodes use routing, then a network's receiver learns all the source messages irrespective of the target function it demands. In Section 3.4.1, we prove a similar result when the intermediate nodes use linear network coding. It is shown that whenever a target function is not reducible the linear computing capacity coincides with the routing capacity and the receiver must learn all the source messages. We also show that there exists a network such that the computing capacity is larger than the routing capacity whenever the target function is non-injective. Hence, if the target function is not reducible, such capacity gain must be obtained from nonlinear coding. Section 3.4.2 shows that linear codes may provide a computing capacity gain over routing for reducible target functions and that linear codes may not suffice to obtain the full computing capacity gain over routing.

3.4.1 Non-reducible target functions

Verifying whether or not a given target function is reducible may not be easy. We now define a class of target functions that are easily shown to not be reducible.

Definition 3.4.1. A target function $f : \mathcal{A}^s \longrightarrow \mathcal{B}$ is said to be *semi-injective* if there exists $x \in \mathcal{A}^s$ such that $f^{-1}(\{f(x)\}) = \{x\}$.

Note that injective functions are semi-injective.

Example 3.4.2. If f is the arithmetic sum target function, then f is semi-injective (since $f(x) = 0$ implies $x = 0$) but not injective (since $f(0,1) = f(1,0) = 1$). Other examples of semi-injective target functions include the identity, maximum, and minimum functions.

Lemma 3.4.3. *If alphabet $\mathcal{A}$ is a ring, then semi-injective target functions are not reducible.*

Proof. Suppose that a target function f is reducible. Then there exists an integer λ satisfying $\lambda < s$, matrix $T \in \mathcal{A}^{s\times\lambda}$, and map $g : \mathcal{A}^\lambda \longrightarrow \mathcal{B}$ such that

$$g(xT) = f(x) \quad \text{for each } x \in \mathcal{A}^s. \tag{3.11}$$

Since $\lambda < s$, there exists a non-zero $d \in \mathcal{A}^s$ such that $dT = 0$. Then for each $x \in \mathcal{A}^s$,

$$f(d + x) = g((d + x)T) = g(xT) = f(x) \tag{3.12}$$

so f is not semi-injective. □

Definition 3.4.4. Let $\mathcal{A}$ be a finite field and let $\mathcal{M}$ be a subspace of the vector space $\mathcal{A}^s$ over the scalar field $\mathcal{A}$. Let

$$\mathcal{M}^\perp = \left\{y \in \mathcal{A}^s : xy^t = 0 \text{ for all } x \in \mathcal{M}\right\}$$

and let $\dim(\mathcal{M})$ denote the dimension of $\mathcal{M}$ over $\mathcal{A}$.

Lemma 3.4.5. [4] *If $\mathcal{A}$ is a finite field and $\mathcal{M}$ is a subspace of vector space $\mathcal{A}^s$, then $(\mathcal{M}^\perp)^\perp = \mathcal{M}$.*

Lemma 3.4.6 will be used in Theorem 3.4.8. The lemma states an alternative characterization of reducible target functions when the source alphabet is a finite field and of semi-injective target functions when the source alphabet is a group.

Lemma 3.4.6. *Let $\mathcal{N}$ be a network with target function $f : \mathcal{A}^s \longrightarrow \mathcal{B}$ and alphabet $\mathcal{A}$.*

(i) Let $\mathcal{A}$ be a finite field. f is reducible if and only if there exists a non-zero $d \in \mathcal{A}^s$ such that for each $a \in \mathcal{A}$ and each $x \in \mathcal{A}^s$,

$$f(ad + x) = f(x).$$

(ii) Let $\mathcal{A}$ be a group. f is semi-injective if and only if there exists $x \in \mathcal{A}^s$ such that for every non-zero $d \in \mathcal{A}^s$,

$$f(d + x) \neq f(x).$$

(The arithmetic in $ad + x$ and $d + x$ is performed component-wise over the corresponding $\mathcal{A}$.)

Proof. (i) If f is reducible, then there exists an integer λ satisfying $\lambda < s$, matrix $T \in \mathcal{A}^{s\times\lambda}$, and map $g : \mathcal{A}^\lambda \longrightarrow \mathcal{B}$ such that

$$g(xT) = f(x) \quad \text{for each } x \in \mathcal{A}^s. \tag{3.13}$$

Since $\lambda < s$, there exists a non-zero $d \in \mathcal{A}^s$ such that $dT = 0$. Then for each $a \in \mathcal{A}$ and each $x \in \mathcal{A}^s$,

$$f(ad + x) = g((ad + x)T) = g(xT) = f(x). \tag{3.14}$$

Conversely, suppose that there exists a non-zero d such that (3.14) holds for every $a \in \mathcal{A}$ and every $x \in \mathcal{A}^s$ and let $\mathcal{M}$ be the one-dimensional subspace of $\mathcal{A}^s$ spanned by d. Then

$$f(t + x) = f(x) \quad \text{for every } t \in \mathcal{M}, x \in \mathcal{A}^s. \tag{3.15}$$

Note that $\dim(\mathcal{M}^\perp) = s - 1$. Let $\lambda = s - 1$, let $T \in \mathcal{A}^{s\times\lambda}$ be a matrix such that its columns form a basis for $\mathcal{M}^\perp$, and let $\mathcal{R}_T$ denote the row space of T. Define the map

$$g : \mathcal{R}_T \longrightarrow f(\mathcal{A}^s)$$

as follows. For any $y \in \mathcal{R}_T$ such that $y = xT$ for $x \in \mathcal{A}^s$, let

$$g(y) = g(xT) = f(x). \tag{3.16}$$

[4] This lemma is a standard result in coding theory regarding dual codes over finite fields, even though the operation xy^t is not an inner product (e.g. [Hill 90, Theorem 7.5] or [Nebe 06, Corollary 3.2.3]). An analogous result for orthogonal complements over inner product spaces is well known in linear algebra (e.g. [Hoffman 71, Theorem 5 on pg. 286]).

Note that if $y = x^{(1)}T = x^{(2)}T$ for $x^{(1)} \neq x^{(2)}$, then

$$\begin{aligned}
(x^{(1)} - x^{(2)})T &= 0 \\
x^{(1)} - x^{(2)} &\in (\mathcal{M}^{\perp})^{\perp} && \text{[from construction of } T\text{]} \\
x^{(1)} - x^{(2)} &\in \mathcal{M} && \text{[from Lemma 3.4.5]} \\
f(x^{(1)}) &= f((x^{(1)} - x^{(2)}) + x^{(2)}) \\
&= f(x^{(2)}). && \text{[from (3.15)]}
\end{aligned}$$

Thus g is well defined. Then from (3.16) and Definition 3.2.1, f is reducible.

(ii) Since f is semi-injective, there exists a $x \in \mathcal{A}^s$ such that $\{x\} = f^{-1}(\{f(x)\})$, which in turn is true if and only if for each non-zero $d \in \mathcal{A}^s$, we have $f(d + x) \neq f(x)$.

□

The following example shows that if the alphabet $\mathcal{A}$ is not a finite field, then the assertion in Lemma 3.4.6(i) may not be true.

Example 3.4.7. Let $\mathcal{A} = \mathbb{Z}_4$, let $f : \mathcal{A} \longrightarrow \mathcal{A}$ be the target function defined by $f(x) = 2x$, and let $d = 2$. Then, for all $a \in \mathcal{A}$,

$$\begin{aligned}
f(2a + x) &= 2(2a + x) \\
&= 2x && \text{[from } 4 = 0 \text{ in } \mathbb{Z}_4\text{]} \\
&= f(x)
\end{aligned}$$

but, f is not reducible, since $s = 1$.

Theorem 3.4.8 establishes for a network with a finite field alphabet, whenever the target function is not reducible, linear computing capacity is equal to the routing computing capacity, and therefore if a linear network code is used, the receiver ends up learning all the source messages even though it only demands a function of these messages.

For network coding (i.e. when f is the identity function), many multi-receiver networks have a larger linear capacity than their routing capacity. However, all single-receiver networks are known to achieve their coding capacity with routing [Lehman 03]. For network computing, the next theorem shows that with non-reducible target functions there is no advantage to using linear coding over routing.[5]

Theorem 3.4.8. *Let $\mathcal{N}$ be a network with target function $f : \mathcal{A}^s \longrightarrow \mathcal{B}$ and alphabet $\mathcal{A}$. If $\mathcal{A}$ is a finite field and f is not reducible, or $\mathcal{A}$ is a ring with identity and f is semi-injective, then*

$$\mathcal{C}_{lin}(\mathcal{N}, f) = \mathcal{C}_{rout}(\mathcal{N}, f)\,.$$

[5] As a reminder, "network" here refers to single-receiver networks in the context of computing.

Proof. Since any routing code is in particular a linear code,

$$\mathcal{C}_{\text{lin}}(\mathcal{N}, f) \geq \mathcal{C}_{\text{rout}}(\mathcal{N}, f)\,.$$

Now consider a (k, n) linear code that computes the target function f in $\mathcal{N}$ and let C be a cut. We will show that for any two collections of source messages, if the messages agree at sources not separated from ρ by C and the vectors agree on edges in C, then there exist two other source message collections with different target function values, such that the receiver ρ cannot distinguish this difference. In other words, the receiver cannot properly compute the target function in the network.

For each $e \in C$, there exist $k \times n$ matrices $M(e)_1, \ldots, M(e)_s$ such that the vector carried on e is

$$\sum_{i=1}^{s} \alpha(\sigma_i)\, M(e)_i.$$

For any matrix M, denote its j-th column by $M^{(j)}$. Let w and y be different $k \times s$ matrices over $\mathcal{A}$, whose j-th columns agree for all $j \notin I_C$.

Let us suppose that the vectors carried on the edges of C, when the the column vectors of w are the source messages, are the same as when the the column vectors of y are the source messages. Then, for all $e \in C$,

$$\sum_{i=1}^{s} w^{(i)} M(e)_i = \sum_{i=1}^{s} y^{(i)} M(e)_i. \tag{3.17}$$

We will show that this leads to a contradiction, namely that ρ cannot compute f. Let m be an integer such that if d denotes the m-th row of $w - y$, then $d \neq 0$. For the case where $\mathcal{A}$ is a field and f is not reducible, by Lemma 3.4.6(i), there exist $a \in \mathcal{A}$ and $x \in \mathcal{A}^s$ such that $ad \neq 0$ and

$$f(ad + x) \neq f(x). \tag{3.18}$$

In the case where $\mathcal{A}$ is a ring with identity and f is semi-injective, we obtain (3.18) from Lemma 3.4.6(ii) in the special case of $a = 1$.

Let u be any $k \times s$ matrix over $\mathcal{A}$ whose m-th row is x and let $v = u + a(w - y)$. From (3.18), the target function f differs on the m-th rows of u and v. Thus, the vectors on the in-edges of the receiver ρ must differ between two cases: (1) when the sources messages are the columns of u, and (2) when the sources messages are the columns of v. The vector carried by any in-edge of the receiver is a function of each of the message vectors $\alpha(\sigma_j)$, for $j \notin I_C$, and the vectors carried by the edges in the cut C. Furthermore, the j-th columns of u and v agree if $j \notin I_C$. Thus, at least one of the vectors on an edge in C must change when the set of source message vectors changes from u to v. However this is contradicted by the fact that for

all $e \in C$, the vector carried on e when the columns of u are the source messages is

$$\begin{aligned}\sum_{i=1}^{s} u^{(i)} M(e)_i &= \sum_{i=1}^{s} u^{(i)} M(e)_i + a \sum_{i=1}^{s} (\mathbf{w}^{(i)} - y^{(i)})) M(e)_i \qquad \text{[from (3.17)]} \\ &= \sum_{i=1}^{s} v^{(i)} M(e)_i \qquad (3.19)\end{aligned}$$

which is also the vector carried on e when the columns of v are the source messages.

Hence, for any two different matrices w and y whose j-th columns agree for all $j \notin I_C$, at least one vector carried by an edge in the cut C has to differ in value in the case where the source messages are the columns of w from the case where the source messages are the columns of y. This fact implies that

$$(|\mathcal{A}|^n)^{|C|} \geq (|\mathcal{A}|^k)^{|I_C|}$$

and thus

$$\frac{k}{n} \leq \frac{|C|}{|I_C|}.$$

Since the cut C is arbitrary, we conclude (using (3.4)) that

$$\frac{k}{n} \leq \min_{C \in \Lambda(\mathcal{N})} \frac{|C|}{|I_C|} = \mathcal{C}_{\text{rout}}(\mathcal{N}, f)\,.$$

Taking the supremum over all (k, n) linear network codes that compute f in $\mathcal{N}$, we get

$$\mathcal{C}_{\text{lin}}(\mathcal{N}, f) \leq \mathcal{C}_{\text{rout}}(\mathcal{N}, f)\,.$$

□

Theorem 3.4.8 showed that if a network's target function is not reducible (e.g. semi-injective target functions) then there can be no computing capacity gain of using linear coding over routing. The following theorem shows that if the target function is injective, then there cannot even be any nonlinear computing gain over routing.

Note that if the identity target function is used in Theorem 3.4.9, then the result states that there is no coding gain over routing for ordinary network coding. This is consistent since our stated assumption in Section 3.2 is that only single-receiver networks are considered here (for some networks with two or more receivers, it is well known that linear coding may provide network coding gain over network routing).

Theorem 3.4.9. *If $\mathcal{N}$ is a network with an injective target function f, then*

$$\mathcal{C}_{cod}(\mathcal{N}, f) = \mathcal{C}_{rout}(\mathcal{N}, f)\,.$$

Proof. It follows from [Lehman 03, Theorem 4.2] that for any single-receiver network $\mathcal{N}$ and the identity target function f, we have $\mathcal{C}_{\text{cod}}(\mathcal{N}, f) = \mathcal{C}_{\text{rout}}(\mathcal{N}, f)$. This can be straightforwardly extended to injective target functions for network computing. □

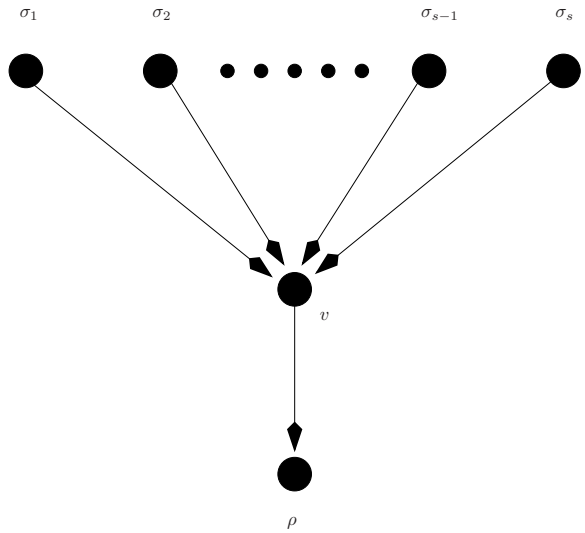


Figure 3.3: Network $\mathcal{N}_{2,s}$ has sources $\sigma_1, \sigma_2, \ldots, \sigma_s$, each connected to the relay v by an edge and v is connected to the receiver by an edge.

Theorem 3.4.8 showed that there cannot be linear computing gain for networks whose target functions are not reducible, and Theorem 3.4.9 showed that the same is true for target functions that are injective. However, Theorem 3.4.11 will show via an example network that nonlinear codes may provide a capacity gain over linear codes if the target function is not injective. This reveals a limitation of linear codes compared to nonlinear ones for non-injective target functions that are not reducible. For simplicity, in Theorem 3.4.11 we only consider the case when there are two or more sources. We need the following lemma first.

Lemma 3.4.10. *The computing capacity of the network $\mathcal{N}_{2,s}$ shown in Figure 3.3, with respect to a target function $f : \mathcal{A}^s \longrightarrow \mathcal{B}$, satisfies*

$$\mathcal{C}_{cod}(\mathcal{N}_{2,s}, f) \geq \min\left\{1,\ \frac{1}{\log_{|\mathcal{A}|}|f(\mathcal{A}^s)|}\right\}.$$

Proof. Suppose

$$\log_{|\mathcal{A}|}|f(\mathcal{A}^s)| < 1. \tag{3.20}$$

Let $k = n = 1$ and assume that each source node sends its message to node v. Let

$$g\ :\ f(\mathcal{A}^s) \longrightarrow \mathcal{A}$$

be any injective map (which exists by (3.20)). Then the node v can compute g and send it to the receiver. The receiver can compute the value of f from the value of g and thus a rate of 1 is achievable, so $\mathcal{C}_{\text{cod}}(\mathcal{N}_{2,s}, f) \geq 1$.

Now suppose

$$\log_{|\mathcal{A}|} |f(\mathcal{A}^s)| \geq 1. \tag{3.21}$$

Choose integers k and n such that

$$\frac{1}{\log_{|\mathcal{A}|} |f(\mathcal{A}^s)|} - \epsilon \leq \frac{k}{n} \leq \frac{1}{\log_{|\mathcal{A}|} |f(\mathcal{A}^s)|}. \tag{3.22}$$

Now choose an arbitrary injective map (which exists by (3.22))

$$g \ : \ (f(\mathcal{A}^s))^k \longrightarrow \mathcal{A}^n.$$

Since $n \geq k$ (by (3.21) and (3.22)), we can still assume that each source sends its k-length message vector to node v. Node v computes f for each of the k sets of source messages, encodes those values into an n-length vector over $\mathcal{A}$ using the injective map g and transmits it to the receiver. The existence of a decoding function which satisfies (3.2) is then obvious from the fact that g is injective. From (3.22), the above code achieves a computing rate of

$$\frac{k}{n} \geq \frac{1}{\log_{|\mathcal{A}|} |f(\mathcal{A}^s)|} - \epsilon.$$

Since ϵ was arbitrary, it follows that the computing capacity $\mathcal{C}_{\text{cod}}(\mathcal{N}_{2,s}, f)$ is at least $1/\log_{|\mathcal{A}|} |f(\mathcal{A}^s)|$. □

Theorem 3.4.11. *Let $\mathcal{A}$ be a finite field alphabet. Let $s \geq 2$ and let f be a target function that is neither injective nor reducible. Then there exists a network $\mathcal{N}$ such that*

$$\mathcal{C}_{cod}(\mathcal{N}, f) > \mathcal{C}_{lin}(\mathcal{N}, f).$$

Proof. If $\mathcal{N}$ is the network $\mathcal{N}_{2,s}$ shown in Figure 3.3 with alphabet $\mathcal{A}$, then

$$\begin{aligned}
\mathcal{C}_{\text{lin}}(\mathcal{N}, f) &= 1/s && \text{[from Theorem 3.4.8 and (3.4)]} \\
&< \min\left\{1, \ \frac{1}{\log_{|\mathcal{A}|} |f(\mathcal{A}^s)|}\right\} && \text{[from } s \geq 2 \text{ and } |f(\mathcal{A}^s)| < |\mathcal{A}|^s\text{]} \\
&\leq \mathcal{C}_{\text{cod}}(\mathcal{N}, f). && \text{[from Lemma 3.4.10]}
\end{aligned}$$

□

The same proof of Theorem 3.4.11 shows that it also holds if the alphabet $\mathcal{A}$ is a ring with identity and the target function f is semi-injective but not injective.

3.4.2 Reducible target functions

In Theorem 3.4.12, we prove a converse to Theorem 3.4.8 by showing that if a target function is reducible, then there exists a network in which the linear computing capacity is larger than the routing computing capacity. Theorem 3.4.14 shows that, even if the target function is reducible, linear codes may not achieve the full (nonlinear) computing capacity of a network.

Theorem 3.4.12. *Let $\mathcal{A}$ be a ring. If a target function $f : \mathcal{A}^s \longrightarrow \mathcal{B}$ is reducible, then there exists a network $\mathcal{N}$ such that*

$$\mathcal{C}_{lin}(\mathcal{N}, f) > \mathcal{C}_{rout}(\mathcal{N}, f)\,.$$

Proof. Since f is reducible, there exist $\lambda < s$, a matrix $T \in \mathcal{A}^{s\times\lambda}$, and a map $g : \mathcal{A}^\lambda \longrightarrow f(\mathcal{A}^s)$ such that

$$g(xT) = f(x) \;\text{ for every } x \in \mathcal{A}^s. \qquad \text{[from Definition 3.2.1]} \tag{3.23}$$

Let $\mathcal{N}$ denote the network $\mathcal{N}_{2,s}$ with alphabet $\mathcal{A}$ and target function f. Let $k = 1$, $n = \lambda$ and let the decoding function be $\psi = g$. Since $n \geq 1$, we assume that all the source nodes transmit their messages to node v. For each source vector

$$x = (\alpha(\sigma_1)\,, \alpha(\sigma_2)\,, \ldots, \alpha(\sigma_s))$$

node v computes xT and sends it to the receiver. Having received the n-dimensional vector xT, the receiver computes

$$\begin{aligned} \psi(xT) &= g(xT) && \text{[from } \psi = g\text{]} \\ &= f(x). && \text{[from (3.23)]} \end{aligned}$$

Thus there exists a linear code that computes f in $\mathcal{N}$ with an achievable computing rate of

$$\begin{aligned} \frac{k}{n} &= \frac{1}{\lambda} \\ &> 1/s && \text{[from } \lambda \leq s-1\text{]} \\ &= \mathcal{C}_{\text{rout}}(\mathcal{N}) && \text{[from (3.4)]} \end{aligned}$$

which is sufficient to establish the claim. □

For target functions that are not reducible, any improvement on achievable rate of computing using coding must be provided by nonlinear codes (by Theorem 3.4.8). However, within the class of reducible target functions, it turns out that there are target functions for which linear codes are optimal (i.e., capacity achieving) as shown in Theorem 3.5.7, while for certain other reducible target functions, nonlinear codes might provide a strictly larger achievable computing rate compared to linear codes.

Remark 3.4.13. It is possible for a network $\mathcal{N}$ to have a reducible target function f but satisfy $\mathcal{C}_{\text{lin}}(\mathcal{N}, f) = \mathcal{C}_{\text{rout}}(\mathcal{N}, f)$ since the network topology may not allow coding to exploit the structure of the target function to obtain a capacity gain. For example, the 3-node network in Figure 3.4 with $f(x_1, x_2) = x_1 + x_2$ and finite field alphabet $\mathcal{A}$ has

$$\mathcal{C}_{\text{lin}}(\mathcal{N}, f) = \mathcal{C}_{\text{rout}}(\mathcal{N}, f) = 1.$$

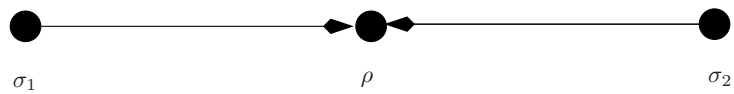


Figure 3.4: A network where there is no benefit to using linear coding over routing for computing f.

Theorem 3.4.11 shows that for every non-injective, non-reducible target function, some network has a nonlinear computing gain over linear coding, and Theorem 3.4.12 shows that for every reducible (hence non-injective) target function, some network has a linear computing gain over routing. The following theorem shows that for some reducible target function, some network has both of these linear and nonlinear computing gains.

Theorem 3.4.14. *There exists a network $\mathcal{N}$ and a reducible target function f such that:*

$$\mathcal{C}_{cod}(\mathcal{N}, f) > \mathcal{C}_{lin}(\mathcal{N}, f) > \mathcal{C}_{rout}(\mathcal{N}, f)\,.$$

Proof. Let $\mathcal{N}$ denote the network $\mathcal{N}_{2,3}$ shown in Figure 3.3 with $s = 3$, alphabet $\mathcal{A} =$, and let f be the target function in Example 3.2.2. The routing capacity is given by

$$\mathcal{C}_{\text{rout}}(\mathcal{N}, f) = 1/3. \qquad \text{[from (3.4)]} \tag{3.24}$$

Let $k = n = 1$. Assume that the sources send their respective messages to node v. The target function f can then be computed at v and sent to the receiver. Hence, $k/n = 1$ is an achievable computing rate and thus

$$\mathcal{C}_{\text{cod}}(\mathcal{N}, f) \geq 1. \tag{3.25}$$

Now consider any (k, n) linear code that computes f in $\mathcal{N}$. Such a linear code immediately implies a (k, n) linear code that computes the target function $g(x_1, x_2) = x_1 x_2$ in network $\mathcal{N}_{2,2}$ as follows. From the (k, n) linear code that computes f in $\mathcal{N}$, we get a $3k \times n$ matrix M such that the node v in network $\mathcal{N}$ computes

$$\begin{pmatrix} \alpha(\sigma_1) & \alpha(\sigma_2) & \alpha(\sigma_3) \end{pmatrix} M$$

and the decoding function computes f from the resulting vector. Now, in $\mathcal{N}_{2,2}$, we let the node v compute

$$\begin{pmatrix} \alpha(\sigma_1) & 0 & \alpha(\sigma_2) \end{pmatrix} M$$

and send it to the receiver. The receiver can compute the function g from the received n-dimensional vector using the relation $g(x_1, x_2) = f(x_1, 0, x_2)$. Using the fact that the function g is not reducible (in fact, it is semi-injective),

$$\begin{aligned} \frac{k}{n} &\leq \mathcal{C}_{\text{lin}}(\mathcal{N}_{2,2}, g) \\ &= \mathcal{C}_{\text{rout}}(\mathcal{N}_{2,2}, g) && \text{[from Theorem 3.4.8]} \\ &= 1/2. && \text{[from (3.4)]} \end{aligned}$$

Consequently,

$$\mathcal{C}_{\text{lin}}(\mathcal{N}, f) \leq 1/2. \tag{3.26}$$

Now we will construct a $(1, 2)$ linear code that computes f in $\mathcal{N}$. Let $k = 1$, $n = 2$ and

$$M = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Let the sources send their respective messages to v while v computes

$$\begin{pmatrix} \alpha(\sigma_1) & \alpha(\sigma_2) & \alpha(\sigma_3) \end{pmatrix} M$$

and transmits the result to the receiver from which f is computable. Since the above code achieves a computing rate of $1/2$, combined with (3.26), we get

$$\mathcal{C}_{\text{lin}}(\mathcal{N}, f) = 1/2. \tag{3.27}$$

The claim of the theorem now follows from (3.24), (3.25), and (3.27). □

3.5 Computing linear target functions

We have previously shown that for reducible target functions there may be a computing capacity gain for using linear codes over routing. In this section, we show that for a special subclass of reducible target functions, namely linear target functions[6] over finite fields, linear network codes achieve the full (nonlinear) computing capacity. We now describe a special class of linear codes over finite fields that suffice for computing linear target functions over finite fields at the maximum possible rate.

Throughout this section, let $\mathcal{N}$ be a network and let k, n, and c be positive integers such that $k/n = c$. Each k symbol message vector generated by a source $\sigma \in S$ can be viewed as a c-dimensional vector

$$\alpha(\sigma) = (\alpha(\sigma)_1, \alpha(\sigma)_2, \ldots, \alpha(\sigma)_c) \in$$

where $\alpha(\sigma)_i \in$ for each i. Likewise, the decoder ψ generates a vector of k symbols from , which can be viewed as a c-dimensional vector of symbols from . For each $e \in \mathcal{E}$, the edge vector z_e is viewed as an element of .

For every node $u \in \mathcal{V} - \rho$, and every out-edge $e \in \mathcal{E}_o(u)$, we choose an encoding function $h^{(e)}$ whose output is:

$$\begin{cases} \sum_{\hat{e} \in \mathcal{E}_i(u)} \gamma_{\hat{e}}^{(e)} z_{\hat{e}} + \sum_{j=1}^{c} \beta_j^{(e)} \alpha(u)_j & \text{if } u \in S \\ \sum_{\hat{e} \in \mathcal{E}_i(u)} \gamma_{\hat{e}}^{(e)} z_{\hat{e}} & \text{otherwise} \end{cases} \tag{3.28}$$

for some $\gamma_{\hat{e}}^{(e)}, \beta_j^{(e)} \in$ and we use a decoding function ψ whose j-th component output ψ_j is:

$$\sum_{e \in \mathcal{E}_i(\rho)} \delta_j^{(e)} z_e \quad \text{for all } j \in \{1, 2, \ldots, c\} \tag{3.29}$$

for certain $\delta_j^{(e)} \in$. Here we view each $h^{(e)}$ as a function of the in-edges to e and the source messages generated by u and we view ψ as a function of the inputs to the receiver. The chosen encoder and decoder are seen to be linear.

Let us denote the edges in $\mathcal{E}$ by $e_1, e_2, \ldots, e_{|\mathcal{E}|}$. For each source σ and each edge $e_j \in \mathcal{E}_o(\sigma)$, let $x_1^{(e_j)}, \ldots, x_c^{(e_j)}$ be variables, and for each $e_j \in \mathcal{E}_i(\rho)$, let $w_1^{(e_j)}, \ldots, w_c^{(e_j)}$ be variables. For every $e_i, e_j \in \mathcal{E}$ such that *head*(e_i) = *tail*(e_j), let $y_{e_i}^{(e_j)}$ be a variable. Let x, y, w be vectors containing all the variables $x_i^{(e_j)}$, $y_{e_i}^{(e_j)}$, and $w_i^{(e_j)}$, respectively. We will use the short hand notation $[y]$ to mean the ring of polynomials $[\cdots, y_{e_i}^{(e_j)}, \cdots]$ and similarly for $[x, y, w]$.

Next, we define matrices $A_\tau(x)$, $F(y)$, and $B(w)$.

[6]The definition of "linear target function" was given in Table 3.2.

(i) For each $\tau \in \{1, 2, \cdots, s\}$, let $A_\tau(x)$ be a $c \times |\mathcal{E}|$ matrix $A_\tau(x)$, given by

$$(A_\tau(x))_{i,j} = \begin{cases} x_i^{(e_j)} & \text{if } e_j \in \mathcal{E}_o(\sigma_\tau) \\ 0 & \text{otherwise} \end{cases} \tag{3.30}$$

(ii) Let $F(y)$ be a $|\mathcal{E}| \times |\mathcal{E}|$ matrix, given by

$$(F(y))_{i,j} = \begin{cases} y_{e_i}^{(e_j)} & \text{if } e_i, e_j \in \mathcal{E} \text{ and } head(e_i) = tail(e_j) \\ 0 & \text{otherwise} \end{cases} \tag{3.31}$$

(iii) Let $B(w)$ be a $c \times |\mathcal{E}|$ matrix, given by

$$(B(w))_{i,j} = \begin{cases} w_i^{(e_j)} & \text{if } e_j \in \mathcal{E}_i(\rho) \\ 0 & \text{otherwise.} \end{cases} \tag{3.32}$$

Consider an (nc, n) linear code of the form in (3.28)–(3.29).

Since the graph G associated with the network is acyclic, we can assume that the edges $e_1, e_2, \ldots$ are ordered such that the matrix F is strictly upper-triangular, and thus we can apply Lemma 3.5.1. Let I denote the identity matrix of suitable dimension.

Lemma 3.5.1. *(Koetter-Médard [Koetter 03, Lemma 2]) The matrix $I - F(y)$ is invertible over the ring $[y]$.*

Lemma 3.5.2. *(Koetter-Médard[Koetter 03, Theorem 3]) For $s = 1$ and for all $\tau \in \{1, \ldots, s\}$, the decoder in* (3.29) *satisfies*

$$\psi = \alpha(\sigma_1)\, A_\tau(\beta)(I - F(\gamma))^{-1} B(\delta)^t.$$

Lemma 3.5.3. *(Alon[Alon 99, Theorem 1.2]) Let be an arbitrary field, and let $g = g(x_1, \ldots, x_m)$ be a polynomial in $[x_1, \ldots, x_m]$. Suppose the degree $deg(g)$ of g is $\sum_{i=1}^m t_i$, where each t_i is a nonnegative integer, and suppose the coefficient of $\prod_{i=1}^m x_i^{t_i}$ in g is nonzero. Then, if $S_1, \ldots, S_m$ are subsets of with $|S_i| > t_i$, there are $s_1 \in S_1$, $s_2 \in S_2, \ldots, s_m \in S_m$ so that*

$$g(s_1, \ldots, s_m) \neq 0.$$

For each $\tau \in \{1, 2, \ldots, s\}$, define the $c \times c$ matrix

$$M_\tau(x, y, w) = A_\tau(x)(I - F(y))^{-1} B(w)^t \tag{3.33}$$

where the components of $M_\tau(x, y, w)$ are viewed as lying in $[x, y, w]$.

Lemma 3.5.4. *If for all $\tau \in \{1, 2, \ldots, s\}$,*

$$\det\left(M_\tau(x, y, w)\right) \neq 0$$

in the ring $[x, y, w]$, then there exists an integer $n > 0$ and vectors β, γ, δ over such that for all $\tau \in \{1, 2, \ldots, s\}$ the matrix $M_\tau(\beta, \gamma, \delta)$ is invertible in the ring of $c \times c$ matrices with components in .

Proof. The quantity

$$\det\left(\prod_{\tau=1}^{s} M_\tau(x,y,w)\right)$$

is a nonzero polynomial in $[x,y,w]$ and therefore also in $[x,y,w]$ for any $n \geq 1$. Therefore, we can choose n large enough such that the degree of this polynomial is less than q^n. For such an n, Lemma 3.5.3 implies there exist vectors β, γ, δ (whose components correspond to the components of the vector variables x, y, w) over such that

$$\det\left(\prod_{\tau=1}^{s} M_\tau(\beta,\gamma,\delta)\right) \neq 0. \tag{3.34}$$

and therefore, for all $\tau \in \{1, 2, \ldots, s\}$

$$\det\left(M_\tau(\beta,\gamma,\delta)\right) \neq 0.$$

Thus, each $M_\tau(\beta,\gamma,\delta)$ is invertible.

□

The following lemma improves upon the upper bound of Lemma 3.2.11 in the special case where the target function is linear over a finite field.

Lemma 3.5.5. *If $\mathcal{N}$ is network with a linear target function f over a finite field, then*

$$\mathcal{C}_{cod}(\mathcal{N}, f) \leq \min_{C \in \Lambda(\mathcal{N})} |C|.$$

Proof. The same argument is used as in the proof of Lemma 3.2.11, except instead of using $R_{I_C,f} \geq 2$, we use the fact that $R_{I_C,f} = |\mathcal{A}|$ for linear target functions. □

Theorem 3.5.6. *If $\mathcal{N}$ is a network with a linear target function f over finite field , then*

$$\mathcal{C}_{lin}(\mathcal{N}, f) = \min_{C \in \Lambda(\mathcal{N})} |C|.$$

Proof. We have

$$\begin{aligned}\mathcal{C}_{\text{lin}}(\mathcal{N}, f) &\leq \mathcal{C}_{\text{cod}}(\mathcal{N}, f) \\ &\leq \min_{C \in \Lambda(\mathcal{N})} |C|. && \text{[from Lemma 3.5.5]}\end{aligned}$$

For a lower bound, we will show that there exists an integer n and an (nc, n) linear code that computes f with a computing rate of $c = \min_{C \in \Lambda(\mathcal{N})} |C|$.

From Lemma 3.5.1, the matrix $I - F(y)$ in invertible over the ring $[x,y,w]$ and therefore also over $[x,y,w]$. Since any minimum cut between the source σ_τ and the receiver ρ has at least c edges, it follows from [Koetter 03, Theorem 2][7] that $\det\left(M_\tau(x,y,w)\right) \neq 0$ for every

[7] Using the implication (1) $\Longrightarrow$ (3) in [Koetter 03, Theorem 2].

$\tau \in \{1, 2, \ldots, s\}$. From Lemma 3.5.4, there exists an integer $n > 0$ and vectors β, γ, δ over such that $M_\tau(\beta, \gamma, \delta)$ is invertible for every $\tau \in \{1, 2, \ldots, s\}$. Since f is linear, we can write

$$f(u_1, \ldots, u_s) = a_1 u_1 + \cdots + a_s u_s.$$

For each $\tau \in \{1, 2, \ldots, s\}$, let

$$\hat{A}_\tau(\beta) = a_\tau \left(M_\tau(\beta, \gamma, \delta)\right)^{-1} A_\tau(\beta). \tag{3.35}$$

If a linear code corresponding to the matrices $\hat{A}_\tau(\beta), B(\delta)$, and $F(\gamma)$ is used in network $\mathcal{N}$, then the c-dimensional vector over computed by the receiver ρ is

$$\begin{aligned}
\psi &= \sum_{\tau=1}^{s} \alpha(\sigma_\tau)\, \hat{A}_\tau(\beta)(I - F(\gamma))^{-1} B(\delta)^t && \text{[from Lemma 3.5.2 and linearity]} \\
&= \sum_{\tau=1}^{s} \alpha(\sigma_\tau)\, a_\tau \left(M_\tau(\beta, \gamma, \delta)\right)^{-1} A_\tau(\beta)(I - F(\gamma))^{-1} B(\delta)^t && \text{[from (3.35)]} \\
&= \sum_{\tau=1}^{s} a_\tau\, \alpha(\sigma_\tau) && \text{[from (3.33)]} \\
&= \left(f(\alpha(\sigma_1)_1, \ldots, \alpha(\sigma_s)_1), \ldots, f(\alpha(\sigma_1)_c, \ldots, \alpha(\sigma_s)_c)\right)
\end{aligned}$$

which proves that the linear code achieves a computing rate of c. □

Theorem 3.5.7 below proves the optimality of linear codes for computing linear target functions in a single-receiver network. It also shows that the computing capacity of a network for a given target function cannot be larger than the number of network sources times the routing computing capacity for the same target function. This bound tightens the general bound given in Theorem 3.2.12 for the special case of linear target functions over finite fields. Theorem 3.5.8 shows that this upper bound can be tight.

Theorem 3.5.7. *If $\mathcal{N}$ is network with s sources and linear target function f over finite field , then*

$$\mathcal{C}_{lin}(\mathcal{N}, f) = \mathcal{C}_{cod}(\mathcal{N}, f) \leq s\, \mathcal{C}_{rout}(\mathcal{N}, f)\,.$$

Proof.

$$\begin{aligned}
s\, \mathcal{C}_{\text{rout}}(\mathcal{N}, f) &\geq \min_{C \in \Lambda(\mathcal{N})} |C| && \text{[from (3.4) and Theorem 3.2.10]} \\
&\geq \mathcal{C}_{\text{cod}}(\mathcal{N}, f) && \text{[from Lemma 3.5.5]} \\
&\geq \mathcal{C}_{\text{lin}}(\mathcal{N}, f) \\
&= \min_{C \in \Lambda(\mathcal{N})} |C|\,. && \text{[from Theorem 3.5.6]}
\end{aligned}$$

□

We note that the inequality in Theorem 3.5.7 can be shown to apply to certain target functions other than linear functions over finite fields, such as the minimum, maximum, and arithmetic sum target functions.

Theorem 3.5.8. *For every s, if a target function $f : \mathcal{A}^s \longrightarrow \mathcal{A}$ is linear over finite field , then there exists a network $\mathcal{N}$ with s sources, such that*

$$\mathcal{C}_{lin}(\mathcal{N}, f) = s\ \mathcal{C}_{rout}(\mathcal{N}, f)\,.$$

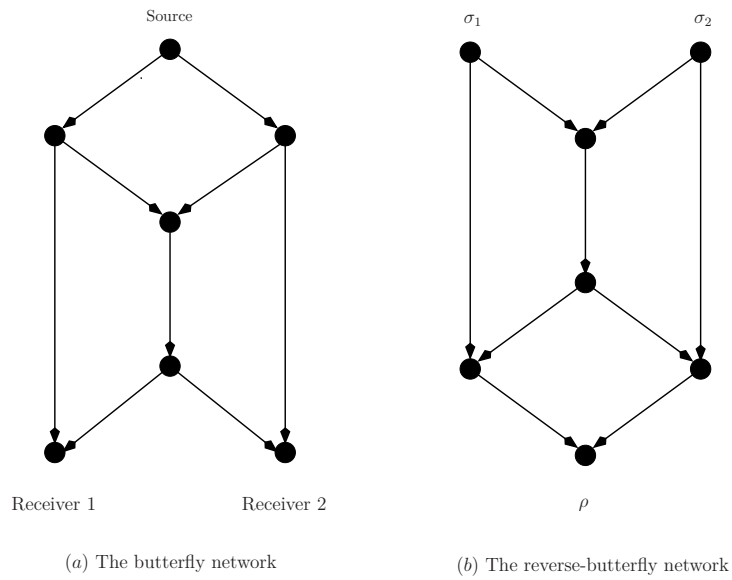
Proof. Let $\mathcal{N}$ denote the network $\mathcal{N}_{2,s}$ shown in Figure 3.3. Then

$$\begin{aligned}
\mathcal{C}_{\text{lin}}(\mathcal{N}, f) &= 1 && \text{[from Theorem 3.5.6]}\\
\mathcal{C}_{\text{rout}}(\mathcal{N}, f) &= \mathcal{C}_{\text{rout}}(\mathcal{N}) && \text{[from Theorem 3.2.10]}\\
&= 1/s. && \text{[from (3.4)]}
\end{aligned}$$

□

3.6 The reverse butterfly network

In this section we study an example network which illustrates various concepts discussed previously in this paper and also provides some interesting additional results for network computing.



(a) The butterfly network

(b) The reverse-butterfly network

Figure 3.5: The butterfly network and its reverse $\mathcal{N}_3$.

The network $\mathcal{N}_3$ shown in Figure 3.5(b) is called the *reverse butterfly network*. It has $S = \{\sigma_1, \sigma_2\}$, receiver node ρ, and is obtained by reversing the direction of all the edges of the multicast butterfly network shown in Figure 3.5(a).

Theorem 3.6.1. *The routing and linear computing capacities of the reverse butterfly network $\mathcal{N}_3$ with alphabet $\mathcal{A} = \{0, 1, \ldots, q-1\}$ and arithmetic sum target function $f : \mathcal{A}^2 \longrightarrow \{0, 1, \ldots, 2(q-1)\}$ are*

$$\mathcal{C}_{rout}(\mathcal{N}_3, f) = \mathcal{C}_{lin}(\mathcal{N}_3, f) = 1.$$

Proof. We have

$$\begin{aligned} \mathcal{C}_{\text{lin}}(\mathcal{N}_3, f) &= \mathcal{C}_{\text{rout}}(\mathcal{N}_3) && \text{[from Theorem 3.4.8]} \\ &= 1. && \text{[from (3.4)]} \end{aligned}$$

□

Remark 3.6.2. The arithmetic sum target function can be computed in the reverse butterfly network at a computing rate of 1 using only routing (by sending σ_1 down the left side and σ_2 down the right side of the graph). Combined with Theorem 3.6.1, it follows that the routing computing capacity is equal to 1 for all $q \geq 2$.

Theorem 3.6.3. *The computing capacity of the reverse butterfly network $\mathcal{N}_3$ with alphabet $\mathcal{A} = \{0, 1, \ldots, q-1\}$ and arithmetic sum target function $f : \mathcal{A}^2 \longrightarrow \{0, 1, \ldots, 2(q-1)\}$ is*

$$\mathcal{C}_{cod}(\mathcal{N}_3, f) = \frac{2}{\log_q (2q-1)}.$$

Remark 3.6.4. The computing capacity $\mathcal{C}_{\text{cod}}(\mathcal{N}_3, f)$ obtained in Theorem 3.6.3 is a function of the coding alphabet $\mathcal{A}$ (i.e. the domain of the target function f). In contrast, for ordinary network coding (i.e. when the target function is the identity map), the coding capacity and routing capacity are known to be independent of the coding alphabet used [Cannons 06]. For the reverse butterfly network, if, for example, $q = 2$, then $\mathcal{C}_{\text{cod}}(\mathcal{N}_3, f)$ is approximately equal to 1.26 and increases asymptotically to 2 as $q \to \infty$.

Remark 3.6.5. The ratio of the coding capacity to the routing capacity for the multicast butterfly network with two messages was computed in [Cannons 06] to be 4/3 (i.e. coding provides a gain of about 33%). The corresponding ratio for the reverse butterfly network increases as a function of q from approximately 1.26 (i.e. 26%) when $q = 2$ to 2 (i.e. 100%) when $q = \infty$. Furthermore, in contrast to the multicast butterfly network, where the coding capacity is equal to the linear coding capacity, in the reverse butterfly network the computing capacity is strictly greater than the linear computing capacity.

Remark 3.6.6. Recall that capacity is defined as the supremum of a set of rational numbers k/n such that a (k, n) code that computes a target function exists. It was pointed out in [Cannons 06] that it remains an open question whether the coding capacity of a network can be irrational. Our Theorem 3.6.3 demonstrates that the computing capacity of a network (e.g. the reverse butterfly network) with unit capacity links can be irrational when the target function to be computed is the arithmetic sum target function of the source messages.

The following lemma is used to prove Theorem 3.6.3.

Lemma 3.6.7. *The computing capacity of the reverse butterfly network $\mathcal{N}_3$ with $\mathcal{A} = \{0, 1, \ldots, q-1\}$ and the mod q sum target function f is*

$$\mathcal{C}_{cod}(\mathcal{N}_3, f) = 2.$$

Proof. The upper bound of 2 on $\mathcal{C}_{\text{cod}}(\mathcal{N}_3, f)$ follows from [Appuswamy 11c, Theorem II.1]. To establish the achievability part, let $k = 2$ and $n = 1$. Consider the code shown in Figure 3.6, where '$\oplus$' indicates the mod q sum. The receiver node ρ gets $\alpha(\sigma_1)_1 \oplus \alpha(\sigma_2)_1$ and $\alpha(\sigma_1)_1 \oplus \alpha(\sigma_2)_1 \oplus \alpha(\sigma_1)_2 \oplus \alpha(\sigma_2)_2$ on its in-edges, from which it can compute $\alpha(\sigma_1)_2 \oplus \alpha(\sigma_2)_2$. This code achieves a rate of 2. □

Proof of Theorem 3.6.3: We have

$$\mathcal{C}_{\text{cod}}(\mathcal{N}, f) \leq 2/\log_q(2q-1). \qquad \text{[from [Appuswamy 11c, Theorem II.1]]}$$

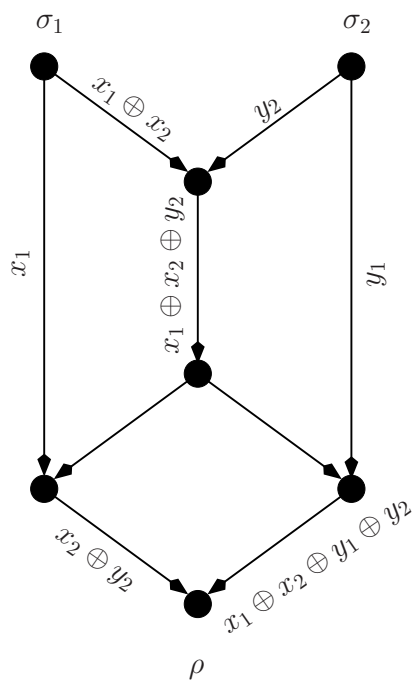


Figure 3.6: The reverse butterfly network with a code that computes the mod q sum target function.

To establish the lower bound, we use the fact the that arithmetic sum of two elements from $\mathcal{A} = \{0, 1, \ldots, q-1\}$ is equal to their mod $2q-1$ sum. Let the reverse butterfly network have alphabet $\hat{\mathcal{A}} = \{0, 1, \ldots, 2(q-1)\}$. From Lemma 3.6.7 (with alphabet $\hat{\mathcal{A}}$), the mod $2q-1$ sum target function can be computed in $\mathcal{N}$ at rate 2. Indeed for every $n \geq 1$, there exists a $(2n, n)$ network code that computes the mod $2q-1$ sum target function at rate 2. So for the remainder of this proof, let $k = 2n$. Furthermore, every such code using $\hat{\mathcal{A}}$ can be "simulated" using $\mathcal{A}$ by a corresponding $(2n, \lceil n \log_q (2q-1) \rceil)$ code for computing the mod $2q-1$ sum target function, as follows. Let n' be the smallest integer such that $q^{n'} \geq (2q-1)^n$, i.e., $n' = \lceil n \log_q (2q-1) \rceil$. Let $g : \hat{\mathcal{A}}^n \to \mathcal{A}^{n'}$ be an injection (which exists since $q^{n'} \geq (2q-1)^n$) and let the function g^{-1} denote the inverse of g on it's image $g(\hat{\mathcal{A}})$. Let $x^{(1)}, x^{(2)}$ denote the first and last, respectively, halves of the message vector $\alpha(\sigma_1) \in \mathcal{A}^{2n}$, where we view $x^{(1)}$ and $x^{(2)}$ as lying in $\hat{\mathcal{A}}^n$ (since $\mathcal{A} \subseteq \hat{\mathcal{A}}$). The corresponding vectors $y^{(1)}, y^{(2)}$ for the source σ_2 are similarly defined.

Figure 3.7 illustrates a $(2n, n')$ code for network $\mathcal{N}$ using alphabet $\mathcal{A}$ where '$\oplus$' denotes the mod $2q-1$ sum. Each of the nodes in $\mathcal{N}$ converts each of the received vectors over $\mathcal{A}$ into a vector over $\hat{\mathcal{A}}$ using the function g^{-1}, then performs coding in Figure 3.6 over $\hat{\mathcal{A}}$, and finally converts the result back to $\mathcal{A}$. Similarly, the receiver node T computes the component-wise

arithmetic sum of the source message vectors $\alpha(\sigma_1)$ and $\alpha(\sigma_2)$ using

$$\begin{aligned}
&\alpha(\sigma_1) + \alpha(\sigma_2) \\
&= \big(g^{-1}(g(x^{(1)} \oplus x^{(2)} \oplus y^{(1)} \oplus y^{(2)})) \ominus g^{-1}(g(x^{(2)} \oplus y^{(2)})), \\
&\qquad g^{-1}(g(x^{(2)} \oplus y^{(2)}))\big) \\
&= (x^{(1)} \oplus y^{(1)}, x^{(2)} \oplus y^{(2)}).
\end{aligned}$$

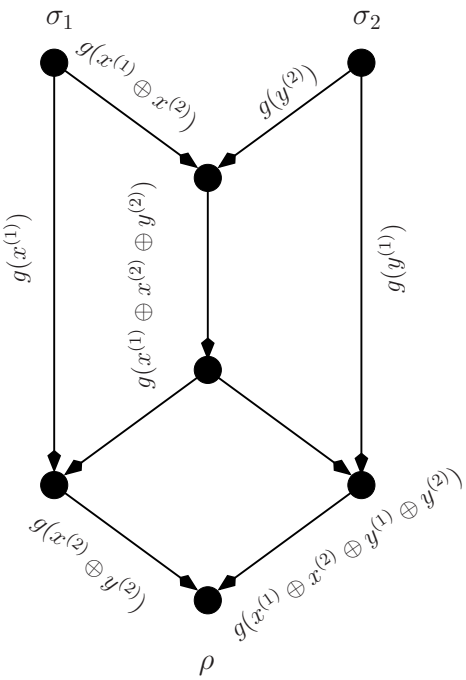


Figure 3.7: The reverse butterfly network with a code that computes the arithmetic sum target function. '$\oplus$' denotes mod $2q-1$ addition.

For any $n \geq 1$, the above code computes the arithmetic sum target function in $\mathcal{N}$ at a rate of

$$\frac{k}{n'} = \frac{2n}{\lceil n \log_q (2q-1) \rceil}.$$

Thus for any $\epsilon > 0$, by choosing n large enough we obtain a code that computes the arithmetic sum target function, and which achieves a computing rate of at least

$$\frac{2}{\log_q (2q-1)} - \epsilon.$$

□

This chapter, in full, is a reprint of the material as it appears in: R. Appuswamy, M. Franceschetti, N. Karamchandani, and K. Zeger, "Linear Codes, Target Function Classes, and

Network Computing Capacity," submitted to the *IEEE Transactions on Information Theory*, May 2011. The dissertation author was the primary investigator of this paper.

Chapter 4

Computing linear functions by linear coding over networks

4.1 Introduction

In many practical networks, including sensor networks and vehicular networks, receivers demand a function of the messages generated by the sources that are distributed across the network rather than the generated messages. This situation is studied in the framework of network computing [Appuswamy 11c, Ramamoorthy 08, Rai 09, Ma 08, Nazer 07]. The classical network coding model of Ahlswede, Cai, Li, and Yeung [Ahlswede 00] can be viewed as a the special case of network computing in which the function to be computed at the receivers corresponds to a subset of the source messages and communication occurs over a network with noiseless links.

In the same noiseless set up of [Ahlswede 00], we consider the scenario in which a set of source nodes generate messages over a finite field and a single receiver node computes a linear function of these messages. We ask whether this linear function can be computed by performing linear coding operations at the intermediate nodes.

In multiple-receiver networks, if each receiver node demands a subset of the source messages (which is an example of a linear function), then Dougherty, Freiling, and Zeger [Dougherty 05] showed that linear codes are not sufficient to recover the source messages. Similarly, if each receiver node demands the sum of the source messages, then Ray and Dei [Rai 09] showed that linear codes are also not sufficient to recover the source messages. In contrast, in single-receiver networks linear codes are sufficient for both the above problems and a simple cut-based condition can be used to test whether a linear solution exists.

Our contribution is as follows. We extend above results investigating if a similar cut-based condition guarantees the existence of a linear solution when the receiver node demands

an arbitrary linear function of the source messages. We identify two classes of functions, one for which the cut-based condition is sufficient for solvability and the other for which it is not. These classes are complements of each other when the source messages are over the binary field. Along the way, we develop an algebraic framework to study linear codes and provide an algebraic condition to test whether a linear solution exists, similar to the one given by Koetter and Médard [Koetter 03] for classical network coding.

The paper is organized as follows. We formally introduce the network computation model in Section 4.1.1. In Section 4.2 we develop the necessary algebraic tools to study linear codes and introduce the cut-based condition. In Section 4.3, we show the main results for the two classes of functions.

4.1.1 Network model and preliminaries

In this paper, a *network* $\mathcal{N}$ consists of a finite, directed acyclic multigraph $G = (\mathcal{V}, \mathcal{E})$, a set of *source nodes* $S = \{\sigma_1, \ldots, \sigma_s\} \subseteq \mathcal{V}$, and a *receiver* $\rho \in \mathcal{V}$. Such a network is denoted by $\mathcal{N} = (G, S, \rho)$. We use the word "graph" to mean a multigraph, and "network" to mean a single-receiver network. We assume that $\rho \notin S$, and that the graph G contains a directed path from every node in $\mathcal{V}$ to the receiver ρ. For each node $u \in \mathcal{V}$, let $\mathcal{E}_i(u)$ and $\mathcal{E}_o(u)$ denote the in-edges and out-edges of u respectively. We also assume (without loss of generality) that if a network node has no in-edges, then it is a source node. We use s to denote the number of sources $|S|$ in the network.

An *alphabet* $\mathcal{A}$ is a nonzero finite field. For any positive integer m, any vector $x \in \mathcal{A}^m$, and any i, let x_i denote the i-th component of x. For any index set $K = \{i_1, i_2, \ldots, i_q\} \subseteq \{1, 2, \ldots, m\}$ with $i_1 < i_2 < \ldots < i_q$, let x_K denote the vector $(x_{i_1}, x_{i_2}, \ldots, x_{i_q}) \in \mathcal{A}^{|K|}$.

The *network computing* problem consists of a network $\mathcal{N}$, a source alphabet $\mathcal{A}$, and a *target function*

$$f \ : \ \mathcal{A}^s \longrightarrow \mathcal{B}$$

where $\mathcal{B}$ is the *decoding alphabet*. A target function f is *linear* if there exists a matrix T over $\mathcal{A}$ such that

$$f(x) = Tx^t, \quad \forall\ x \in \mathcal{A}^s$$

where 't' denotes matrix transposition. For linear target functions the decoding alphabet is of the form $\mathcal{A}^l$, with $1 \leq l \leq s$. Without loss of generality, we assume that T is full rank (over $\mathcal{A}$) and has no zero columns. For example, if T is the $s \times s$ identity matrix, then the receiver demands the complete set of source messages, and this corresponds to the classical network coding problem. On the other hand, if T is the row vector of 1's, then the receiver demands a sum (over $\mathcal{A}$) of the source values. Let n be a positive integer. Given a network $\mathcal{N}$ with source set S and alphabet $\mathcal{A}$, a *message generator* is a mapping

$$\alpha \ : \ S \longrightarrow \mathcal{A}^n.$$

For each source $\sigma_i \in S$, $\alpha(\sigma_i)$ is called a *message vector* and it can be viewed as an element of $\mathbb{F}_{q^n}$ (rather than as a vector).

Definition 4.1.1. A *linear network code* in a network $\mathcal{N}$ consists of the following:

(i) Every edge $e \in \mathcal{E}$ *carries an element* of $\mathbb{F}_{q^n}$ and this element is denoted by z_e. For any node $v \in \mathcal{V} - \rho$ and any out-edge $e \in \mathcal{E}_o(v)$, the network code specifies an *encoding function* $h^{(e)}$ of the form:

$$h^{(e)} = \begin{cases} x_{1,e}\alpha(u) + \sum\limits_{\hat{e}\in\mathcal{E}_i(u)} x_{\hat{e},e} z_{\hat{e}} & \text{if } u \in S \\ \sum\limits_{\hat{e}\in\mathcal{E}_i(u)} x_{\hat{e},e} z_{\hat{e}} & \text{otherwise} \end{cases} \tag{4.1}$$

where $x_{\hat{e},e}, x_{1,e} \in \mathbb{F}_{q^n}$ for all $\hat{e} \in \mathcal{E}_i(u)$.

(ii) The *decoding function* ψ outputs a vector of length l whose j-th component is of the form:

$$\sum_{e\in\mathcal{E}_i(\rho)} x_{e,j} z_e \tag{4.2}$$

where $x_{e,j} \in \mathbb{F}_{q^n}$ for all $e \in \mathcal{E}_i(\rho)$. The arithmetic in (4.1) and (4.2) is performed over $\mathbb{F}_{q^n}$.

In this paper, by a *network code*, we always mean a linear network code. In the literature, the class of network codes we define here is referred to as *scalar linear codes.* These codes were introduced and studied in [Koetter 03]. A more general class of linear codes over $\mathbb{F}_{q^n}$ were defined and studied in [Dougherty 05, Dougherty 08].

Depending on the context, we may view z_e as a vector of length-n over $\mathbb{F}_q$ or as an element of $\mathbb{F}_{q^n}$. Without explicit mention, we use the fact that the addition of $a, b \in \mathbb{F}_{q^n}$ as elements of a finite field coincides with their sum as elements of a vector space over $\mathbb{F}_q$. Furthermore, we also view $\mathbb{F}_q$ as a subfield of $\mathbb{F}_{q^n}$ without explicitly stating the inclusion map. Let $z_{e_1}, z_{e_2}, \ldots, z_{e_{|\mathcal{E}_i(\rho)|}}$ denote the vectors carried by the in-edges of the receiver.

Definition 4.1.2. A linear network code over $\mathbb{F}_{q^n}$ is called *a linear solution for computing f in $\mathcal{N}$* (or simply a *linear solution* if f and $\mathcal{N}$ are clear from the context) if the decoding function ψ is such that for every message generator α,

$$\psi\left(z_{e_1}, \cdots, z_{e_{|\mathcal{E}_i(\rho)|}}\right)_j = f\Big(\alpha(\sigma_1)_j, \cdots, \alpha(\sigma_s)_j\Big) \quad \text{for all } j \in \{1, 2, \ldots, n\}. \tag{4.3}$$

Remark 4.1.3. Each source generates n symbols over $\mathbb{F}_q$ (viewing $\mathbb{F}_{q^n}$ as a vector space over $\mathbb{F}_q$) and the decoder computes the target function f for each set of source symbols.

A set of edges $C \subseteq \mathcal{E}$ is said to *separate* sources $\sigma_{m_1}, \ldots, \sigma_{m_d}$ from the receiver ρ, if for each $i \in \{1, 2, \ldots, d\}$, every path from σ_{m_i} to ρ contains at least one edge in C. A set $C \in \mathcal{E}$ is

said to be a *cut* if it separates at least one source from the receiver. Let $\Lambda(\mathcal{N})$ denote the set of all cuts in network $\mathcal{N}$.

For any matrix $T \in \mathbb{F}_q^{l\times s}$, let T_i denote its i-th column. For an index set $K \in \{1, 2, \ldots, s\}$, let T_K denote the $l \times |K|$ submatrix of T obtained by choosing the columns of T indexed by K. If C is a cut in a network $\mathcal{N}$, we define the set

$$K_C = \{i \in S : C \text{ disconnects } \sigma_i \text{ from } \rho\}.$$

Finally, for any network $\mathcal{N}$ and matrix T, we define

$$\text{min-cut}(\mathcal{N}, T) = \min_{C\in\Lambda(\mathcal{N})} \frac{|C|}{\text{rank}(T_{K_C})}. \tag{4.4}$$

4.2 Algebraic framework

4.2.1 An algebraic test for the existence of a linear solution

Linear solvability for the classical network coding problem was shown to be equivalent to the existence of a non-empty algebraic variety in [Koetter 03]. In the following, we present an analogous characterization for computing linear functions, providing an algebraic test to determine whether a linear solution for computing a linear function exists. The reverse problem of constructing a multiple-receiver network coding (respectively, network computing) problem given an arbitrary set of polynomials, which is solvable if and only if the corresponding set of polynomials is simultaneously solvable is considered in reference [Dougherty 08] (respectively, [Rai 09]).

We begin by giving some definitions and stating a technical lemma, followed by the main theorem below.

For any edge $e = (u, v) \in \mathcal{E}$, let *head(e)* $= v$ and *tail(e)* $= u$. Associated with a linear code over $\mathbb{F}_{q^n}$, we define the following three types of matrices:

- For each source $\sigma_\tau \in S$, define the $1 \times |\mathcal{E}|$ matrix A_τ as follows:

$$(A_\tau)_{1,j} = \begin{cases} x_{1,e_j} & \text{if } e_j \in \mathcal{E}_o(\sigma_t) \\ 0 & \text{otherwise.} \end{cases} \tag{4.5}$$

- Similarly define the $l \times |\mathcal{E}|$ matrix B as follows:

$$B_{i,j} = \begin{cases} x_{e_j,i} & \text{if } e_j \in \mathcal{E}_i(\rho) \\ 0 & \text{otherwise.} \end{cases} \tag{4.6}$$

- Define the $|\mathcal{E}| \times |\mathcal{E}|$ matrix F as follows:

$$F_{i,j} = \begin{cases} x_{e_i,e_j} & \text{if } \textit{head}(e_i) = \textit{tail}(e_j) \\ 0 & \text{otherwise.} \end{cases} \tag{4.7}$$

Since the graph G associated with the network is acyclic, we can assume that the edges $e_1, e_2, \ldots$ are ordered such that the matrix F is strictly upper-triangular. Let I denote the identity matrix of suitable dimension. Consider a network $\mathcal{N}$ with alphabet $\mathbb{F}_q$ and consider a linear code over $\mathbb{F}_{q^n}$ with associated matrices $A_1, A_2, \ldots, A_s, B$ and F. For every $\tau \in \{1, 2, \ldots, s\}$, define the $1 \times l$ matrix

$$M_\tau = A_\tau (I - F)^{-1} B^t. \tag{4.8}$$

Now let x_A be a vector containing all the non-zero entries of the matrices $A_\tau, \tau = 1, 2, \cdots, s$, and let x_B (respectively, x_F) be a vector containing all the non-zero entries of the matrix B (respectively, F).

By abusing notation, depending on the context we may view x_{e_i,e_j}, x_{i,e_j}, $x_{e_i,j}$ as elements of $\mathbb{F}_{q^n}$ or as indeterminates. Thus, each of the matrices defined above may either be a matrix over $\mathbb{F}_{q^n}$ or a matrix over the polynomial ring $R = \mathbb{F}_{q^n}[x_A, x_F, x_B]$. The context should make it clear which of these two notions is being referred to at any given point.

Lemma 4.2.1. *The following two statements hold:*

1. *The matrix $I - F$ has a polynomial inverse with coefficients in $\mathbb{F}_{q^n}[x_F]$, the ring of polynomials in the variables constituting x_F.*

2. *The decoding function can be written as*

$$\sum_{\tau=1}^{s} \alpha(\sigma_\tau)\, A_\tau (I - F)^{-1} B^t$$

Proof. The first assertion is a restatement of [Koetter 03, Lemma 2] and the second assertion follows from [Koetter 03, Theorem 3]. □

Definition 4.2.2. Let R be a polynomial ring. The ideal generated by a subset $X \subset R$ and denoted by $\langle X \rangle$ is the smallest ideal in R containing X.

Let $\mathcal{N}$ be a network with alphabet $\mathbb{F}_q$. Let $R = \mathbb{F}_q[x_A, x_F, x_B]$ and $T \in \mathbb{F}_q^{l \times s}$. Consider a linear network code for computing the linear function corresponding to T in $\mathcal{N}$ and the associated matrices $M_\tau, \tau = 1, 2, \ldots, s$ over R and define

$$Z_\tau = (T_\tau)^t - M_\tau \text{ for } \tau = 1, 2, \ldots, s.$$

Let J denote the ideal generated by the elements of $Z_\tau \in R^{1 \times l}, \tau = 1, 2, \ldots, s$ in the ring R. More formally, let

$$J = \langle \{ \{ (Z_\tau)_1, (Z_\tau)_2, \ldots, (Z_\tau)_l \} : \tau = 1, 2, \ldots, s \} \rangle .$$

The polynomials $(Z_i)_j$ are referred to as the *generating polynomials* of the ideal J. We denote the Gröbner basis of an ideal generated by subset $X \subset R$ of a polynomial ring R by $\mathcal{G}(X)$. The following theorem is a consequence of Hilbert Nullstellensatz (see [Hungerford 97, Lemma VIII.7.2] and the remark after [Hungerford 97, Proposition VIII.7.4]).

Theorem 4.2.3. *Consider a network $\mathcal{N}$ with alphabet $\mathbb{F}_q$ and the linear target function f corresponding to a matrix $T \in \mathcal{A}^{l\times s}$. There exists an $n > 0$ and a linear solution over $\mathbb{F}_{q^n}$ for computing f in $\mathcal{N}$ if and only if $\mathcal{G}(J) \neq \{1\}$.*

Proof. From Lemma 4.2.1, the vector computed at the receiver can be written as

$$\psi\left(z_{e_1}, \cdots, z_{e_{|\mathcal{E}_i(\rho)|}}\right) = \begin{pmatrix} M_1^t & M_2^t & \cdots & M_s^t \end{pmatrix} \begin{pmatrix} \alpha(\sigma_1) \\ \alpha(\sigma_2) \\ \vdots \\ \alpha(\sigma_s) \end{pmatrix}. \tag{4.9}$$

On the other hand, to compute the linear function corresponding to T, the decoding function must satisfy

$$\psi\left(z_{e_1}, \cdots, z_{e_{|\mathcal{E}_i(\rho)|}}\right) = T \begin{pmatrix} \alpha(\sigma_1) \\ \alpha(\sigma_2) \\ \vdots \\ \alpha(\sigma_s) \end{pmatrix}. \qquad \text{[from (4.3)]} \tag{4.10}$$

It follows that the encoding coefficients in a linear solution must be such that

$$(T_\tau)^t - M_\tau = 0 \text{ for } \tau = 1, 2, \ldots, s. \qquad \text{[from (4.9) and (4.10)]} \tag{4.11}$$

If we view the coding coefficients as variables, then it follows that a solution must simultaneously solve the generating polynomials of the corresponding ideal J. By [Hungerford 97, Lemma VIII.7.2], such a solution exists over the algebraic closure $\bar{\mathbb{F}}_q$ of $\mathbb{F}_q$ if and only if $J \neq \mathbb{F}_q[x_A, x_F, x_B]$. Furthermore, $J \neq \mathbb{F}_q[x_A, x_F, x_B]$ if and only if $\mathcal{G}(J) \neq \{1\}$. Moreover, a solution exists over the algebraic closure $\bar{\mathbb{F}}_q$ of $\mathbb{F}_q$ if and only if it exists over some extension field $\mathbb{F}_{q^n}$ of $\mathbb{F}_q$ and the proof is now complete. □

4.2.2 Minimum cut condition

It is clear that the set of linear functions that can be solved in a network depends on the network topology. It is easily seen that a linear solution for computing a linear target function corresponding to $T \in \mathbb{F}_q^{l\times s}$ exists only if the network $\mathcal{N}$ is such that for every $C \in \Lambda(\mathcal{N})$, the value of the cut $|C|$ is at least the rank of the submatrix T_{K_C} (recall that K_C is the index set of the sources separated by the cut C). This observation is stated in the following lemma which is an immediate consequence of the cut-based bound in [Appuswamy 11c, Theorem 2.1].

Lemma 4.2.4. *For a network $\mathcal{N}$, a necessary condition for the existence of a linear solution for computing the target function corresponding to $T \in \mathbb{F}_q^{l\times s}$ is*

$$\textit{min-cut}(\mathcal{N}, T) \geq 1.$$

We now consider two special cases. First, consider the case in which the receiver demands all the source messages. The corresponding T is given by the $s \times s$ identity matrix I and the condition min-cut$(\mathcal{N}, T) \geq 1$ reduces to

$$\frac{|C|}{|K_C|} \geq 1 \quad \forall\, C \in \Lambda(\mathcal{N})$$

i.e., the number of edges in the cut be at least equal to the number of sources separated by the cut. Second, consider the case in which the receiver demands the sum of the source messages. The corresponding matrix T is an $1 \times s$ row vector and the requirement that min-cut$(\mathcal{N}, T) \geq 1$ reduces to

$$|C| \geq 1 \quad \forall\, C \in \Lambda(\mathcal{N})$$

i.e., all the sources have a directed path to the receiver. For both of the above cases, the cut condition in Lemma 4.2.4 is also sufficient for the existence of a solution. This is shown in [Appuswamy 11c, Theorem 3.1 and Theorem 3.2] and is reported in the following Lemma:

Lemma 4.2.5. *Let $l \in \{1, s\}$. For a network $\mathcal{N}$ with the linear target function f corresponding to a matrix $T \in \mathcal{A}^{l \times s}$, a linear solution exists if and only if min-cut$(\mathcal{N}, T) \geq 1$.*

The focus in the rest of the paper is to extend above results to the case $l \notin \{1, s\}$ by using the algebraic test of Theorem 4.2.3.

4.3 Computing linear functions

In the following, we first define an equivalence relation among matrices and then use it to identify a set of functions that are linearly solvable in every network satisfying the condition min-cut$(\mathcal{N}, T) \geq 1$. We then construct a linear function outside this set, and a corresponding network with min-cut$(\mathcal{N}, T) \geq 1$, on which such a function cannot be computed with linear codes. Finally, we use this example as a building block to identify a set of linear functions for which there exist networks satisfying the min-cut condition and on which these functions are not solvable.

Notice that for a linear function with matrix $T \in \mathbb{F}_q^{l \times s}$, each column of T corresponds to a single source node. Hence, for every $s \times s$ permutation matrix Π, computing Tx is equivalent to computing $T\Pi x$ after appropriately renaming the source nodes. Furthermore, for every $l \times l$ full rank matrix Q over $\mathbb{F}_q$, computing Tx is equivalent to computing QTx. These observations motivate the following definition:

Definition 4.3.1. Let $T \in \mathbb{F}_2^{l \times s}$ and $T' \in \mathbb{F}_2^{l \times s}$. We say $T \sim T'$ if there exist an invertible matrix Q of size $l \times l$ and a permutation matrix Π of size $s \times s$ such that $T = QT'\Pi$, and $T \nsim T'$ if such Q and Π do not exist.

Since T is assumed to be a full rank matrix, Π can be chosen such that the first l columns of $T\Pi$ are linearly independent. Let $\hat{T}$ denote the first l columns of $T\Pi$. By choosing $Q = \hat{T}^{-1}$, we have $T \sim QT\Pi = (I\ P)$ where P is an $l \times s-l$ matrix. So for an arbitrary linear target function f and an associated matrix T, there exists an $l \times s-l$ matrix P such that $T \sim (I\ P)$. Without loss of generality, we assume that each column of T associated with a target function is non-zero.

Theorem 4.3.2. *Consider a network $\mathcal{N}$ with a linear target function corresponding to a matrix $T \in \mathbb{F}_q^{(s-1)\times s}$ (i.e., $l = s-1$). If*

$$T \sim (I\ u)$$

where u is a column vector of units, then a necessary and sufficient condition for the existence of a linear solution is min-cut$(\mathcal{N}, T) \geq 1$.

Proof. Let $T = (I\ u)$. The 'necessary' part is clear from Lemma 4.2.4. We now focus on the 'sufficiency' part. Notice that for each $\tau = 1, 2, \ldots, s$, the matrix M_τ (computed as in (4.8)) is a row vector of length $s-1$. Stack these s row vectors to form an $s \times (s-1)$ matrix M as follows,

$$M = \begin{pmatrix} M_1 \\ M_2 \\ \vdots \\ M_s \end{pmatrix}.$$

Let $M_{(i)}$ denote the $(s-1) \times (s-1)$ submatrix of M obtained by deleting its i-th row.
Claim 1: The matrix

$$\prod_{i=1}^{s} M_{(i)}$$

has a non-zero determinant over the ring $R = \mathbb{F}_q[x_A, x_F, x_B]$.
Claim 2: For each $i = 1, 2, \ldots, s-1$, we have $\left(A_s(I-F)^{-1}B^t M_{(s)}^{-1}\right)_i \neq 0$.
By Claim 1 and the sparse zeros lemma [Koetter 03], [Schwartz 80], it follows that that there exists some $n > 0$ such that the variables $x_{e',e}, x_{e,l}$ can be assigned values over $\mathbb{F}_{q^n}$ so that the $s \times (s-1)$ matrix

$$M = \begin{pmatrix} A_1(I-F)^{-1}B^t \\ A_2(I-F)^{-1}B^t \\ \vdots \\ A_s(I-F)^{-1}B^t \end{pmatrix}$$

is such that any of its $(s-1) \times (s-1)$ submatrices $M_{(i)}, i = 1, 2, \ldots, s$ obtained by deleting the i-th row in M, is full rank over $\mathbb{F}_{q^n}$. Define two $s-1 \times s-1$ diagonal matrices U and D such that for $i \in \{1, 2, \cdots, s-1\}$

$$\begin{aligned} U_{i,i} &= u_i \\ D_{i,i} &= \left(A_s(I-F)^{-1}B^t M_{(s)}^{-1}\right)_i. \end{aligned} \tag{4.12}$$

Now define the following matrices over $\mathbb{F}_{q^n}$:

$$\begin{aligned}
\bar{B} &= D^{-1}U(M^t_{(s)})^{-1}B \\
\bar{A}_i &= u_i^{-1}\left(A_s(I-F)^{-1}\bar{B}^t\right)_i A_i \quad i = 1, 2, \ldots, s-1 \\
\bar{A}_s &= A_s.
\end{aligned} \tag{4.13}$$

By by Claim 2 it follows that D^{-1} exists. If the matrices $\bar{A}_\tau, F$, and $\bar{B}$ define a linear network code, then by Lemma 4.2.1, the vector received by ρ can be written as,

$$\bar{M}^t \begin{pmatrix} \alpha(\sigma_1) \\ \alpha(\sigma_2) \\ \vdots \\ \alpha(\sigma_s) \end{pmatrix} \tag{4.14}$$

where,

$$\bar{M} = \begin{pmatrix} \bar{A}_1(I-F)^{-1}\bar{B}^t \\ \bar{A}_2(I-F)^{-1}\bar{B}^t \\ \vdots \\ \bar{A}_s(I-F)^{-1}\bar{B}^t \end{pmatrix}. \tag{4.15}$$

We have

$$\begin{aligned}
\begin{pmatrix} A_1(I-F)^{-1}\bar{B}^t \\ A_2(I-F)^{-1}\bar{B}^t \\ \vdots \\ A_s(I-F)^{-1}\bar{B}^t \end{pmatrix} &= \begin{pmatrix} A_1(I-F)^{-1}(D^{-1}U(M_{(s)}^t)^{-1}B)^t \\ A_2(I-F)^{-1}(D^{-1}U(M_{(s)}^t)^{-1}B)^t \\ \vdots \\ A_s(I-F)^{-1}(D^{-1}U(M_{(s)}^t)^{-1}B)^t \end{pmatrix} && [\text{from } \bar{B} = D^{-1}U(M_{(s)}^t)^{-1}B] \\
&= \begin{pmatrix} A_1(I-F)^{-1}B^tM_{(s)}^{-1} \\ A_2(I-F)^{-1}B^tM_{(s)}^{-1} \\ \vdots \\ A_s(I-F)^{-1}B^tM_{(s)}^{-1} \end{pmatrix} D^{-1}U && [\text{from } \left((M_{(s)}^t)^{-1}\right)^t = M_{(s)}^{-1}] \\
&= \begin{pmatrix} I \\ A_s(I-F)^{-1}B^tM_{(s)}^{-1} \end{pmatrix} D^{-1}U && [\text{from construction of } M_{(s)}]
\end{aligned} \tag{4.16}$$

$$\begin{aligned}
\begin{pmatrix} \bar{A}_1(I-F)^{-1}\bar{B}^t \\ \bar{A}_2(I-F)^{-1}\bar{B}^t \\ \vdots \\ \bar{A}_s(I-F)^{-1}\bar{B}^t \end{pmatrix} &= \begin{pmatrix} U^{-1}D \\ A_s(I-F)^{-1}B^tM_{(s)}^{-1} \end{pmatrix} D^{-1}U && [\text{from (4.13) and (4.16) }] \\
&= \begin{pmatrix} U^{-1} \\ \mathbf{1}^t \end{pmatrix} U && [\text{from (4.12)}] \\
&= \begin{pmatrix} I \\ \mathbf{1}^tU \end{pmatrix} \\
&= \begin{pmatrix} I \\ u^t \end{pmatrix} && (4.17) \\
\bar{M}^t &= \begin{pmatrix} I & u \end{pmatrix}. && [\text{from (4.15) and (4.17)}] \quad (4.18)
\end{aligned}$$

By substituting (4.18) in (4.14), we conclude that the receiver computes the desired linear function by employing the network code defined by the encoding matrices $\{\bar{A}_i, i = 1, 2, \ldots, s\}$, $\bar{B}$, and F.

The proof of the theorem is now complete for the case when $T = (I\ u)$. If $T \sim (I\ u)$, then there exists a full-rank matrix Q and a column vector u' of non-zero elements over $\mathbb{F}_q$ such that

$$T = Q\ (I\ u'). \qquad [\text{from From Lemma 4.4.1 in the Appendix}]$$

Since a full-rank linear operator preserves linear-independence among vectors, for every such full-rank matrix Q, we have

$$\text{rank}(T_{K_C}) = \text{rank}\big((Q^{-1}T)_{K_C}\big) \quad \forall\ C \in \Lambda(\mathcal{N}). \tag{4.19}$$

Equation (4.19) implies that min-cut$(\mathcal{N}, T) =$ min-cut$\big(\mathcal{N}, Q^{-1}T\big)$. Since $Q^{-1}T = (I\ u')$, from the first part of the proof, there exist an $n > 0$ and coding matrices $A_\tau, \tau = 1, 2, \cdots, s$, F, and B over $\mathbb{F}_{q^n}$ such that the receiver can compute the linear target function corresponding to $(I\ u')$ if and only if min-cut$(\mathcal{N}, T) \geq 1$. It immediately follows that by utilizing a code corresponding to the coding matrices $A_\tau, \tau = 1, 2, \cdots, s$, F, and QB, the receiver can compute the target function corresponding to $Q(I\ u') = T$.

All that remains to be done is to provide proofs of claims 1 and 2.

Proof of Claim 1: If a cut C is such that $|K_C| \leq s - 1$, then

$$\begin{aligned} |C| &\geq \text{ rank}(T_{K_C}) && \text{[from min-cut}(\mathcal{N}, T) \geq 1 \text{ and (4.4)]} \\ &= |K_C|\,. && \text{[from } T = (I\ u)\text{]} \end{aligned}$$

Thus by [Appuswamy 11c, Theorem 3.1], there exists a routing solution to compute the identity function of the sources $\{\sigma_i, i \in K_C\}$ at the receiver. Let $|K_C| = s-1$ and let $K_C = \{1, 2, \ldots, j-1, j+1, \ldots, s\}$ for some (arbitrary) j. By Lemma 4.2.1, after fixing $\alpha(\sigma_j) = 0$, the vector received by ρ can be written as

$$M_{(j)}^t \begin{pmatrix} \alpha(\sigma_1) \\ \alpha(\sigma_2) \\ \vdots \\ \alpha(\sigma_{j-1}) \\ \alpha(\sigma_{j+1}) \\ \vdots \\ \alpha(\sigma_s) \end{pmatrix}.$$

The existence of a routing solution for computing the identity function guarantees that there exist $x_{e',e}, x_{e,l} \in \{0, 1\}$ such that the matrix $M_{(j)}$ has a non-zero determinant over $\mathbb{F}_q$. It follows that the determinant of $M_{(j)}$ is non-zero over $\mathbb{F}_q[x_A, x_F, x_B]$. Since $j \in \{1, 2, \ldots, s\}$ was arbitrary in the above argument, it follows that the determinant of each $M_{(j)}, j = 1, 2, \ldots, s$ is non-zero over $\mathbb{F}_q[x_A, x_F, x_B]$ and the claim follows.

Proof of Claim 2: We have

$$\begin{aligned} M\ M_{(s)}^{-1} &= \begin{pmatrix} A_1(I-F)^{-1}B^t \\ A_2(I-F)^{-1}B^t \\ \vdots \\ A_s(I-F)^{-1}B^t \end{pmatrix} \quad M_{(s)}^{-1} \\ &\overset{(a)}{=} \begin{pmatrix} I \\ A_s(I-F)^{-1}B^t M_{(s)}^{-1} \end{pmatrix} \end{aligned} \tag{4.20}$$

where, (a) follows from the definition of $M_{(s)}^{-1}$. By contraction, assume that there exists an

$i \in \{1, 2, \ldots, s-1\}$ such that $\left(A_s(I-F)^{-1}\bar{B}^t\right)_i = 0$. It then follows that

$$A_s(I-F)^{-1}B^t M_{(s)}^{-1} = \sum_{j=1}^{s-2} \left(A_s(I-F)^{-1}B^t M_{(s)}^{-1}\right)_{i_j} (A_{i_j}(I-F)^{-1}B^t M_{(s)}^{-1}) \quad \text{[from (4.20)]} \tag{4.21}$$

for some choice of $i_j \in \{1, 2, \ldots, s-1\}, j = 1, 2, \ldots, s-2$ and

$$\left(A_s(I-F)^{-1}B^t - \sum_{j=1}^{s-2} \left(A_s(I-F)^{-1}B^t M_{(s)}^{-1}\right)_{i_j} (A_{i_j}(I-F)^{-1}B)^t\right) M_{(s)}^{-1} = 0 \quad \text{[from (4.21)]}$$

$$\left(A_s(I-F)^{-1}B^t - \sum_{j=1}^{s-2} \left(A_s(I-F)^{-1}B^t M_{(s)}^{-1}\right)_{i_j} (A_{i_j}(I-F)^{-1}B)^t\right) = 0. \tag{4.22}$$

Equation (4.22) implies a linear dependence among $s-1$ rows of the matrix M. This contradicts the fact that for each $i = 1, 2, \ldots, s$, $M_{(i)}$ is full rank. Thus $\left(A_s(I-F)^{-1}B^t M_{(s)}^{-1}\right)_i \neq 0$ for $i = 1, 2, \ldots, s-1$ and the claim follows.

□

Remark 4.3.3. We provide the following communication-theoretic interpretation of our method of proof above. We may view the computation problem as a MIMO (multiple input multiple output) channel where the multiple input is given by the vector of symbols generated by the sources, the output is the vector decoded by the receiver, and the channel is given by the network topology and the network code. Our objective is to choose a channel to guarantee the desired output, by way of code design subject to the constraints imposed by network topology. The channel gain from source σ_i to the receiver is given by the vector M_i of length $s-1$. The first part of the proof utilizes the sparse zeros lemma to establish that there exists a choice of channels such that the channel between every set of $s-1$ sources and the receiver is invertible. This is similar to the proof of the multicast theorem in [Koetter 03]. In the second part of the proof, we recognize that the interference from different sources must also be "aligned" at the output for the receiver to be able to compute the desired function. Accordingly, we have modified the code construction to provide such alignment.

We now show the existence of a linear function that cannot be computed on a network satisfying the min-cut condition. This network will then be used as a building block to show an analogous result for a larger class of functions. Let T_1 denote the matrix

$$\begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \tag{4.23}$$

and let f_1 denote the corresponding linear function. It is possible to show with some algebra that $T_1 \not\sim (I\ u)$, for any column vector u of units, so that the conclusion of Theorem 4.3.2 does not hold. Indeed, for the function f_1 the opposite conclusion is true, namely f_1 cannot be computed over $\mathcal{N}_1$ using linear codes. This is shown by the following Lemma.

Lemma 4.3.4. *Let $\mathcal{N}_1$ be the network shown in Figure 4.1 with alphabet $\mathbb{F}_q$. We have*

1. *min-cut$(\mathcal{N}_1, T_1) = 1$.*
2. *There does not exist a linear solution for computing f_1 in $\mathcal{N}_1$.*

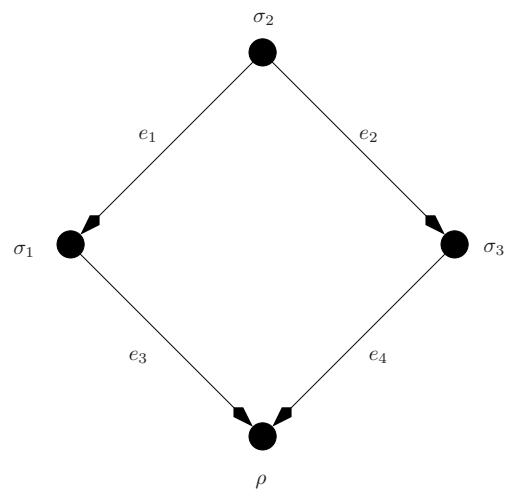


Figure 4.1: Network on which there is no linear solution for computing f_1.

Proof. That min-cut$(\mathcal{N}_1, T_1) = 1$ is easily verified by considering the cut $C = \{e_3, e_4\}$ which attains the minimum. We now proceed to show, using Theorem 4.2.3, that a linear solution does not exist.

We may assume, without loss of generality, that the node σ_2 sends its message directly to nodes σ_1 and σ_3 (i.e., $x_{1,e_1} = x_{1,e_2} = 1$). The matrices Z_1, Z_2, and Z_3 over R can then be written as

$$\begin{aligned}
(T_1)^t - M_1 &= \begin{pmatrix} (1 - x_{1,e_3}x_{e_3,1}) & (0 - x_{1,e_3}x_{e_3,2}) \end{pmatrix} \\
(T_2)^t - M_2 &= \begin{pmatrix} 0 - x_{e_1,e_3}x_{e_3,1} - x_{e_2,e_4}x_{e_4,1} \\ 1 - x_{e_1,e_3}x_{e_3,2} - x_{e_2,e_4}x_{e_4,2} \end{pmatrix}^t \\
(T_3)^t - M_3 &= \begin{pmatrix} (1 - x_{1,e_4}x_{e_4,1}) & (0 - x_{1,e_4}x_{e_4,2}) \end{pmatrix}.
\end{aligned}$$

Consequently, the ideal J is given by

$$\begin{aligned}
J = \big\langle &(1 - x_{1,e_3}x_{e_3,1}),\ (0 - x_{1,e_3}x_{e_3,2}), \\
&(0 - x_{e_1,e_3}x_{e_3,1} - x_{e_2,e_4}x_{e_4,1}), \\
&(1 - x_{e_1,e_3}x_{e_3,2} - x_{e_2,e_4}x_{e_4,2}), \\
&(1 - x_{1,e_4}x_{e_4,1}),\ (0 - x_{1,e_4}x_{e_4,2})\big\rangle.
\end{aligned}$$

We have

$$\begin{aligned}1 = &(1 - x_{e_1,e_3}x_{e_3,2} - x_{e_2,e_4}x_{e_4,2})\\ &+ x_{e_1,e_3}x_{e_3,2}(1 - x_{1,e_3}x_{e_3,1})\\ &- x_{e_1,e_3}x_{e_3,1}(0 - x_{1,e_3}x_{e_3,2})\\ &+ x_{e_2,e_4}x_{e_4,2}(1 - x_{1,e_4}x_{e_4,1})\\ &- x_{e_2,e_4}x_{e_4,1}(0 - x_{1,e_4}x_{e_4,2}) \ \in \ J.\end{aligned}$$

Thus, it follows that $\mathcal{G}(J) = \{1\}$. By Theorem 4.2.3, a linear solution does not exist for computing f_1 in $\mathcal{N}_1$. □

We now identify a much larger class of linear functions for which there exist networks satisfying the min-cut condition but for which linear solutions do not exist. Let P be an $l \times s - l$ matrix with at least one zero element and $T \sim (I\ P)$. For each T in this equivalence class we show that there exist a network $\mathcal{N}$ that does not have a solution for computing the linear target function corresponding to T but satisfies the cut condition in Lemma 4.2.4. The main idea of the proof is to establish that a solution for computing such a function in network $\mathcal{N}$ implies a solution for computing the function corresponding to T_1 in $\mathcal{N}_1$, and then to use Lemma 4.3.4.

Theorem 4.3.5. *Consider a linear target function f corresponding to a matrix $T \in \mathbb{F}_q^{l\times s}$. If $T \sim (I\ P)$ such that at least one element of P is zero, then there exists a network $\mathcal{N}$ such that*

1. *min-cut$(\mathcal{N}, T) = 1$.*
2. *There does not exist a linear solution for computing f in $\mathcal{N}$.*

Proof. Let $\hat{T} = (I\ P)$ and let $\hat{f}$ denote the corresponding linear target function. It is enough to show that there exists a network $\mathcal{N}_P$ such that min-cut$(\mathcal{N}_P, f) = 1$ but $\mathcal{N}_P$ does not have a linear solution for computing $\hat{f}$. This is because a network $\mathcal{N}$ that does not have a solution for computing T is easily obtained by renaming the sources in $\mathcal{N}_P$ as follows: Since $T \sim (I\ P)$, there exist Q and Π such that $T = Q(I\ P)\Pi$. Let κ denote the permutation function on the set $\{1, 2, \ldots, s\}$ defined by the permutation matrix Π^{-1}. Obtain the network $\mathcal{N}$ by relabeling source σ_i in $\mathcal{N}_P$ as $\sigma_{\kappa(i)}$. To see that there does not exist a solution for computing f in $\mathcal{N}$, assume to the contrary that a solution exists. By using the same network code in $\mathcal{N}_P$, the receiver computes

$$Q(I\ P)\Pi\ (x_{\kappa(1)}, x_{\kappa(2)}, \ldots, x_{\kappa(s)})^t = Q(I\ P)\ (x_1, x_2, \ldots, x_s)^t.$$

Thus the receiver in $\mathcal{N}_P$ can compute $\hat{T}x^t$, which is a contradiction.

Now we construct the network $\mathcal{N}_P$ as claimed. Since P has at least once zero element, there exists a $\tau \in \{l+1, l+2, \ldots, s\}$ such that $\hat{T}$ has a zero in τ-th column. Define

$$K = \left\{ i \in \{1, 2, \ldots, l\} : \hat{T}_{i,\tau} = 1 \right\}$$

Denote the elements of K by

$$\left\{j_1, j_2, \ldots, j_{|K|}\right\}.$$

Let p be an element of $\{1, 2, \ldots, l\} - K$ (such a p exists from the fact that the τ-th column contains at least one zero) and define

$$\bar{K} = \{1, 2, \ldots, s\} - K - \{\tau, p\}$$

and denote the elements of $\bar{K}$ by

$$\left\{j_{|K|+1}, j_{|K|+2}, \ldots, j_{s-|K|-2}\right\}.$$

Since $\hat{T}$ does not contain an all-zero column, $|K| > 0$. Now, let $\mathcal{N}_P$ denote the network shown in Figure 4.2 where, v denotes a relay node. It follows from the construction that

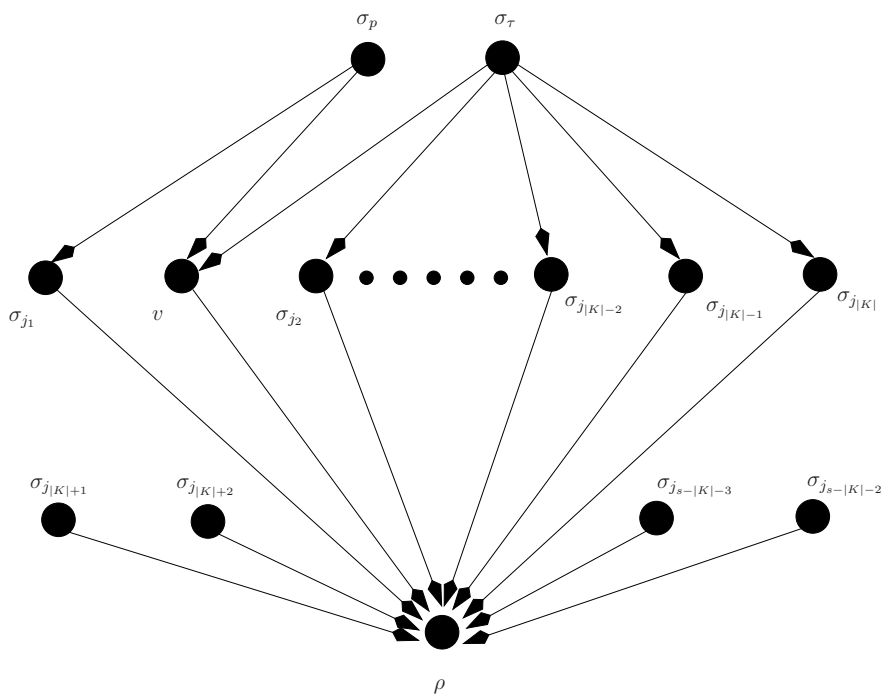


Figure 4.2: Network $\mathcal{N}_P$ with min-cut 1 that does not have an $\mathbb{F}_q$-linear solution for computing $(I\ P)$.

$$\begin{pmatrix} \hat{T}_{j_1,j_1} & \hat{T}_{j_1,p} & \hat{T}_{j_1,\tau} \\ \hat{T}_{p,j_1} & \hat{T}_{p,p} & \hat{T}_{p,\tau} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \tag{4.24}$$

which is equal to the transfer matrix T_1 defined in (4.23).

Notice that in the special case when $K = \{j_1\}$ and $\left|\bar{K}\right| = 0$, the network shown in Figure 4.2 reduces to the network shown in Figure 4.3 which is equivalent to the network $\mathcal{N}_1$

in Figure 4.1 with target function f_1. Since $\mathcal{N}_1$ does not have a solution for computing f_1 by Lemma 4.3.4, we conclude that $\mathcal{N}_1$ cannot have a solution either.

Similarly, we now show that in the general case, if the network $\mathcal{N}_P$ has a solution for computing $\hat{f}$, then such a solution induces a solution for computing f_1 in network $\mathcal{N}_1$, contradicting Lemma 4.3.4. Let there exist an $n > 0$ for which there is a linear solution for computing $\hat{f}$ over $\mathcal{N}_P$ using an alphabet over $\mathbb{F}_{q^n}$. In any such solution, for each $j \in K - \{j_1\}$, the encoding function on the edge (σ_j, ρ) must be of the form

$$\beta_{1,j}\alpha(\sigma_j) + \beta_{2,j}\alpha(\sigma_\tau) \tag{4.25}$$

for some $\beta_{1,j}, \beta_{2,j} \in \mathbb{F}_{q^n}$. Since (σ_j, ρ) is the only path from source σ_j to the receiver, it is obvious that $\beta_{1,j} \neq 0$.

We define the map α as follows. Let $\alpha(\sigma_{j_1}), \alpha(\sigma_p), \alpha(\sigma_\tau)$ be arbitrary elements of $\mathbb{F}_{q^n}$ and let

$$\alpha(\sigma_j) = \begin{cases} 0 & \text{for } j \in \bar{K} \\ -(\beta_{1,j})^{-1}\beta_{2,j}\alpha(\sigma_\tau) & \text{for } j \in K - \{j_1\}. \end{cases} \tag{4.26}$$

Note that α has been chosen such that for any choice of $\alpha(\sigma_{j_1}), \alpha(\sigma_p)$, and $\alpha(\sigma_\tau)$, every edge $e \in \mathcal{E}_i(\rho) - \{(\sigma_{i_1}, \rho), (v, \rho)\}$ carries the zero vector. Furthermore, for the above choice of α, the target function associated with $\hat{T}$ reduces to

$$\left(\alpha(\sigma_1) + \hat{T}_{1,\tau}\alpha(\sigma_\tau), \alpha(\sigma_2) + \hat{T}_{2,\tau}\alpha(\sigma_\tau), \ldots, \alpha(\sigma_l) + \hat{T}_{l,\tau}\alpha(\sigma_\tau)\right). \tag{4.27}$$

Substituting $\hat{T}_{j_1,\tau} = 1$ and $\hat{T}_{p,\tau} = 0$ in (4.27), it follows that the receiver can compute

$$(\alpha(\sigma_{j_1}) + \alpha(\sigma_\tau), \alpha(\sigma_p))$$

from the vectors received on edges (σ_{i_1}, ρ) and (v, ρ). Consequently, it follows that there exist a linear solution over $\mathbb{F}_{q^n}$ for computing the linear target function associated with the transfer matrix

$$\begin{pmatrix} \hat{T}_{j_1,j_1} & \hat{T}_{j_1,p} & \hat{T}_{j_1,\tau} \\ \hat{T}_{p,j_1} & \hat{T}_{p,p} & \hat{T}_{p,\tau} \end{pmatrix}$$

in the network shown in Figure 4.3. It is easy to see that the existence of such a code implies a scalar linear solution for computing f_1 in $\mathcal{N}_1$. This establishes the desired contradiction.

Finally, we show that min-cut$(\mathcal{N}, T) = 1$. Let $C \in \Lambda(\mathcal{N})$ be a cut such that $K_C \subset K \cup \{p, \tau\}$ (i.e, C separates sources from only the top and middle rows in the network $\mathcal{N}_P$). We have the following two cases:

1. If $\sigma_\tau \notin K_C$, then it is easy to see that $|C| \geq |K_C|$. Similarly, if $\sigma_\tau \in K_C$ and $\sigma_p \notin K_C$, then again $|C| \geq |K_C|$. Consequently, we have

$$\begin{aligned} \frac{|C|}{\text{rank}(T_{K_C})} &\geq \frac{|C|}{|K_C|} && [\text{from } \text{rank}(T_{K_C}) \leq |K_C|] \\ &\geq 1. && [\text{from } |C| \geq |K_C|] \end{aligned} \tag{4.28}$$

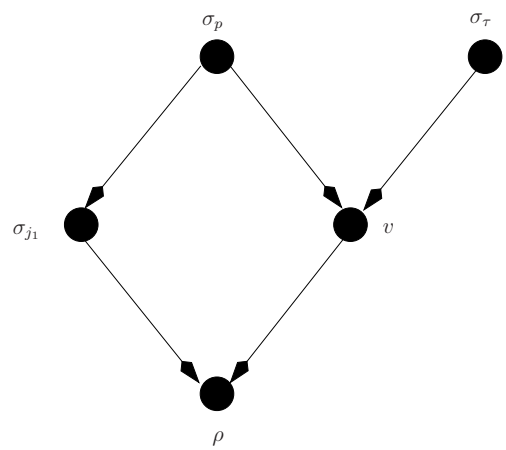


Figure 4.3: Subnetwork of $\mathcal{N}_P$ used to show the equivalence between solving network $\mathcal{N}_P$ and solving network $\mathcal{N}_1$.

2. If $\sigma_\tau \in K_C$ and $\sigma_p \in K_C$, then from Figure 4.3, $|C| = |K| + 1$ and $K_C = K \cup \{p, \tau\}$. Moreover, the index set K was constructed such that

$$\hat{T}_\tau = \sum_{i \in K} \hat{T}_{i,\tau} \hat{T}_i. \tag{4.29}$$

Consequently, we have

$$\begin{aligned} \operatorname{rank}(T_{K_C}) &= \operatorname{rank}\big(T_{K \cup \{p,\tau\}}\big) && [\text{from } K_C = K \cup \{p, \tau\}] \\ &\leq |K| + 1 && [\text{from (4.29)}] \\ &= |C|\,. \end{aligned} \tag{4.30}$$

From (4.28) and (4.30), we conclude that if $K_C \subset K \cup \{p, \tau\}$, then

$$\frac{|C|}{\operatorname{rank}(T_{K_C})} \geq 1. \tag{4.31}$$

For an arbitrary cut $C \in \Lambda(\mathcal{N})$, let $c_{\bar{K}}$ denote the number of sources in $\bar{K}$ that are separated from the receiver by C (i.e, $c_{\bar{K}} = \big|K_C \cap \bar{K}\big|$). We have

$$\begin{aligned} \frac{|C|}{\operatorname{rank}(T_{K_C})} &= \frac{|C| - c_{\bar{K}} + c_{\bar{K}}}{\operatorname{rank}(T_{K_C})} \\ &\geq \frac{|C| - c_{\bar{K}} + c_{\bar{K}}}{\operatorname{rank}\big(T_{K_C - \bar{K}}\big) + c_{\bar{K}}} \end{aligned} \tag{4.32}$$

Since each source in $\bar{K}$ is directly connected to the receiver, $|C| - c_{\bar{K}}$ is equal to the number of edges in C separating the sources in $K_C - \bar{K}$ from the receiver. Consequently, from (4.31), it follows that

$$\frac{|C| - c_{\bar{K}}}{\operatorname{rank}\big(T_{K_C - \bar{K}}\big)} \geq 1. \tag{4.33}$$

Substituting (4.33) in (4.32), we conclude that for all $C \in \Lambda(\mathcal{N})$

$$\text{min-cut}(\mathcal{N}, T) \geq 1.$$

Since the edge $(\sigma_{j_{|K|+1}}, \rho)$ disconnects the source $\sigma_{j_{|K|+1}}$ from the receiver, min-cut$(\mathcal{N}, T) \leq 1$ is immediate and the proof of the theorem is now complete. □

We now consider the case in which the source alphabet is over the binary field. In this case, we have that the two function classes identified by Theorems 4.3.2 and 4.3.5 are complements of each other, namely either $T \sim (I\ \mathbf{1})$ or $T \sim (I\ P)$ with P containing at least one zero element.

Theorem 4.3.6. *Let $l \notin \{1, s\}$ and let $T \in \mathbb{F}_2^{l \times s}$. If $T \nsim (I\ \mathbf{1})$, then there exists an $l \times (s-l)$ matrix P such that P has at least one zero element and $T \sim (I\ P)$.*

Proof. Since T is assumed to have a full row rank, $T \sim (I\ \bar{P})$ for some $l \times (s-l)$ matrix $(I\ \bar{P})$ over $\mathbb{F}_2$. If $\bar{P}$ has 0's, then we are done. Assume to the contrary that $\bar{P}$ is a matrix of non-zero elements. We only need to consider the case when $(s-l) > 1$ (since $T \nsim (I\ \mathbf{1})$). For $i = 1, 2, \ldots, l-1$, let $\phi^{(i)}$ denote the i-th column vector of the $l \times l$ identity matrix. Define $Q = (\phi^{(1)}\phi^{(2)} \cdots \phi^{(l-1)}\ \mathbf{1})$ and let Π be a permutation matrix that interchanges the l-th and $(l+1)$-th columns and leaves the remaining columns unchanged. It is now easy to verify that

$$\begin{aligned} Q\ (I\ \bar{P})\ \Pi &= (Q\ Q\bar{P})\ \Pi \\ &= (I\ P) \end{aligned} \tag{4.34}$$

where P is an $l \times s-l$ matrix with at least one 0 element: for $i \in \{1, 2, \cdots, l-1\}$

$$\begin{aligned} P_{i,2} &= (Q\bar{P})_{i,2} \\ &= (Q\mathbf{1})_i \\ &= 1 + 1 \\ &= 0. \end{aligned}$$

Thus, $(I\ \bar{P}) \sim (I\ P)$ and by transitivity we conclude that $T \sim (I\ P)$ which proves the claim. □

4.4 Appendix

Lemma 4.4.1. *Let $T \in \mathbb{F}_q^{l \times s}$. If $u \in \mathbb{F}_q^{s-1}$ is a column vector of non-zero elements and $T \sim (I\ u)$, then there exists a full rank matrix Q and a column vector u' of non-zero elements over $\mathbb{F}_q$ such that $T = Q\ (I\ u')$.*

Proof. Let Q denote the matrix obtained by collecting the first $(s-1)$ columns of T. We will first show that the matrix Q is full-rank. After factoring out Q, we then prove that the last column must have non-zero entries.

Since $T \sim (I\ u)$, there exists a full-rank matrix $\bar{Q}$ and a permutation matrix $\bar{\Pi}$ such that

$$\begin{aligned} T &= \bar{Q}\ (I\ u)\ \bar{\Pi} \\ &= (\bar{Q}\ \bar{Q}u)\ \bar{\Pi}. \end{aligned} \tag{4.35}$$

From (4.35), the columns of Q are constituted by the columns of $\bar{Q}$ in which case Q is full-rank, or columns of Q contains $(s-2)$ columns of $\bar{Q}$ and $\bar{Q}u$. We will now show that the vector $\bar{Q}u$ cannot be written as a linear combination of any set of $s-2$ column vectors of $\bar{Q}$. Assume to the contrary that there exist $a_j \in \mathbb{F}_q$ for $j \in \{1, 2, s-2\}$ such that

$$\bar{Q}u = \sum_{j=1}^{s-2} a_j \bar{Q}_j \tag{4.36}$$

where $\bar{Q}_j$ denotes the j-th column of $\bar{Q}$. Let a denote the vector such that $a_j = a_j, j = 1, 2, \ldots s-2$, and $a_{s-1} = 0$. We have

$$\begin{aligned} u - a &\neq 0 && [\text{from } u_{s-1} \neq 0 \text{ and } a_{s-1} = 0] \\ \bar{Q}(u-a) &= 0 && [\text{from (4.36)}]. \end{aligned} \tag{4.37}$$

(4.37) contradicts the fact that $\bar{Q}$ is full-rank. Hence a_i's satisfying (4.36) do not exist and consequently, Q is a full-rank matrix. We now have

$$T = Q(I\ u')$$

where $u' = Q^{-1}T_s$ and hence $T \sim (I\ u')$. Furthermore, $T \sim (I\ u)$ and $T \sim (I\ u')$ implies that $(I\ u) \sim (I\ u')$. Thus, there exists a full-rank matrix P and a permutation matrix Π such that

$$\begin{aligned} (I\ u) &= P\ (I\ u')\ \Pi \\ &= (P\ Pu')\ \Pi. \end{aligned} \tag{4.38}$$

Let $\phi^{(i)}$ denote the i-th column of I. It follows from (4.38) that either (a) $Pu' = u$ and P itself is an $(s-1)\times(s-1)$ permutation matrix, or (b) For some $j \in \{1, 2, \ldots, s-1\}$, j-th column of P is u, and the remaining columns must constitute the $s-2$ columns $\phi^{(1)}, \phi^{(2)}, \ldots, \phi^{(\tau-1)}, \phi^{(\tau+1)}, \phi^{(s-1)}$ of I for some τ. If (a) is true, then $u' = P^{-1}u$ and the elements of u' are non-zero since P^{-1} is another permutation matrix. If (b) is true, then $Pu' = \phi^{(\tau)}$ and it must be that $u'_j \neq 0$ (if $u'_j = 0$, then $(Pu')_\tau = 0$ which contradicts $Pu' = \phi^{(\tau)}$). Let $L = \{i : i \neq j, \text{ and } u'_i \neq 0\}$. We must have

$$\phi^{(\tau)} = u'_j u + \sum_{i \in D} u'_i\ \phi^{(j_i)}. \tag{4.39}$$

If we denote the number of non-zero entries in a vector u by $|u|$, then we have

$$\begin{aligned} 1 &= \left|\phi^{(\tau)}\right| \\ &\geq \left|u_j' u\right| - |D| && \text{[from (4.39)]} \\ &= (s-1) - |D| \\ &\geq 1 && \text{[from } |D| \leq s-2\text{]} \end{aligned} \tag{4.40}$$

From (4.40), it follows that $|D| = s - 2$ and consequently that every element of u' is non-zero. The proof of the lemma is now complete. □

This chapter, in full, is a reprint of the material as it appears in: R.Appuswamy, and M.Franceschetti, "Computing linear functions by linear coding over networks," submitted to the *IEEE Transactions on Information Theory*, Feb. 2011. The dissertation author was the primary investigator of this paper.

Bibliography

[Ahlswede 00] R. Ahlswede, N. Cai, S.-Y. R. Li & R. W. Yeung. *Network information flow*. IEEE Transactions on Information Theory, vol. 46, no. 4, pages 1204–1216, July 2000.

[Alon 99] N. Alon. *Combinatorial Nullstellensatz*. Combinatorics, Probability and Computing, pages 7–29, 1999.

[Appuswamy 09] R. Appuswamy, M. Franceschetti, N. Karamchandani & K. Zeger. *Network computing capacity for the reverse butterfly network*. In Proceedings of the IEEE International Symposium on Information Theory, pages 259–262, 2009.

[Appuswamy 11a] R. Appuswamy & M. Franceschetti. *Computing linear functions by linear coding over networks*. submitted to the *IEEE Transactions on Information Theory*, 2011.

[Appuswamy 11b] R. Appuswamy, M. Franceschetti, N. Karamchandani & K. Zeger. *Linear Codes, Target Function Classes, and Network Computing Capacity*. submitted to the *IEEE Transactions on Information Theory*, 2011.

[Appuswamy 11c] R. Appuswamy, M. Franceschetti, N. Karamchandani & K. Zeger. *Network coding for computing: cut-set bounds*. IEEE Transactions on Information Theory, vol. 57, no. 2, pages 1015–1030, 2011.

[Ayaso 07] O. Ayaso, D. Shah & M. Dahleh. *Lower Bounds on Information Rates for Distributed Computation via Noisy Channels*. In Proceedings of the forty-fifth Allerton Conference on Computation, Communication and Control, 2007.

[Ayaso 08] O. Ayaso, D. Shah & M. Dahleh. *Counting bits for distributed function computation*. In Proceedings of the IEEE International Symposium on Information Theory, pages 652–656, 2008.

[Benezit 07] F. Benezit, A. G. Dimakis, P. Thiran & M. Vetterli. *Gossip Along the Way: Order-Optimal Consensus through Randomized Path Averaging*. In Proceedings of the forty-fifth Allerton Conference on Computation, Communication and Control, 2007.

[Boyd 06] S. Boyd, A. Ghosh, B. Prabhakar & D. Shah. *Randomized Gossip Algorithms*. IEEE Transactions on Information Theory, vol. 52, no. 6, pages 2508–2530, June 2006.

[Cannons 06] J. Cannons, R. Dougherty, C. Freiling & K. Zeger. *Network Routing Capacity*. IEEE Transactions on Information Theory, vol. 52, no. 3, pages 777–788, March 2006.

[Cuff 09] P. Cuff, H. Su & A. El Gamal. *Cascade Multiterminal Source Coding*. In Proceedings of the IEEE International Symposium on Information Theory, pages 1199–1203, 2009.

[Dimakis 06] A. G. Dimakis, A. D. Sarwate & M. J. Wainwright. *Geographic gossip: efficient aggregation for sensor networks*. In Proceedings of the fifth international conference on Information Processing in Sensor Networks, pages 69–76, 2006.

[Doshi 06] V. Doshi, D. Shah, M. Medard & S. Jaggi. *Graph coloring and conditional graph entropy*. In Proceedings of the Fortieth Asilomar Conference on Signals, Systems and Computers, pages 2137–2141, 2006.

[Doshi 07a] V. Doshi, D. Shah & M. Medard. *Source Coding with Distortion through Graph Coloring*. In Proceedings of the IEEE International Symposium on Information Theory, pages 1501–1505, 2007.

[Doshi 07b] V. Doshi, D. Shah, M. Medard & S. Jaggi. *Distributed functional compression through graph coloring*. In Proceedings of the Data Compression Conference, pages 93–102, 2007.

[Dougherty 05] R. Dougherty, C. Freiling & K. Zeger. *Insufficiency of linear coding in network information flow*. IEEE Transactions on Information Theory, vol. 51, no. 8, pages 2745–2759, 2005.

[Dougherty 06] R. Dougherty, C. Freiling & K. Zeger. *Unachievability of Network Coding Capacity*. IEEE Transactions on Information Theory & IEEE/ACM Transactions on Networking (joint issue), vol. 52, no. 6, pages 2365–2372, June 2006.

[Dougherty 08] R. Dougherty, C. Freiling & K. Zeger. *Linear network codes and systems of polynomial equations*. IEEE Transactions on Information Theory, vol. 54, no. 5, pages 2303–2316, 2008.

[Dutta 08] C. Dutta, Y. Kanoria, D. Manjunath & J. Radhakrishnan. *A Tight Lower Bound for Parity in Noisy Communication Networks*. In Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete Algorithms, pages 1056–1065, 2008.

[Feng 04] H. Feng, M. Effros & S. Savari. *Functional source coding for networks with receiver side information*. In Proceedings of the forty-second Allerton Conference on Computation, Communication and Control, pages 1419–1427, 2004.

[Ford 56] L. R. Ford & D. R. Fulkerson. *Maximal flow through a network*. Canadian Journal of Mathematics, pages 399–404, 1956.

[Gallager 88] R. G. Gallager. *Finding Parity in a Simple Broadcast Network*. IEEE Transactions on Information Theory, vol. 34, no. 2, pages 176–180, March 1988.

[Gamal 87] A. El Gamal. *Reliable Communication of Highly Distributed Information*. In T. M. Cover & B. Gopinath, editeurs, Open Problems in Communication and Computation, pages 60–62. Springer-Verlag, 1987.

[Giridhar 05] A. Giridhar & P. R. Kumar. *Computing and Communicating Functions over Sensor Networks*. IEEE Journal on Selected Areas in Communication, vol. 23, no. 4, pages 755–764, April 2005.

[Goyal 08] N. Goyal, G. Kindler & M. Saks. *Lower Bounds for the Noisy Broadcast Problem*. SIAM Journal on Computing, vol. 37, no. 6, pages 1806–1841, March 2008.

[Gupta 00] P. Gupta & P. R. Kumar. *The Capacity of Wireless Networks*. IEEE Transactions on Information Theory, vol. 46, no. 2, pages 388–404, March 2000.

[Hardy 79] G. H. Hardy & E. M. Wright. An introduction to the theory of numbers. Oxford University Press, fifth edition, 1979.

[Harvey 04] N. J. A. Harvey, R. D. Kleinberg & A. R. Lehman. *Comparing network coding with multicommodity flow for the k-pairs communication problem*. M.I.T. LCS, Tech. Rep. 964, 2004.

[Harvey 06] N. J. A. Harvey, R. Kleinberg & A. R. Lehman. *On the capacity of information networks*. IEEE Transactions on Information Theory & IEEE/ACM Transactions on Networking (joint issue), vol. 52, no. 6, pages 2345–2364, June 2006.

[Hill 90] R. Hill. A first course in coding theory. Oxford University Press, 1990.

[Hoeffding 63] W. Hoeffding. *Probability Inequalities for Sums of Bounded Random Variables*. Journal of the American Statistical Association, vol. 58, no. 301, pages 13–30, March 1963.

[Hoffman 71] K. M. Hoffman & R. Kunze. Linear algebra. Prentice Hall, 1971.

[Hu 63] T. C. Hu. *Multi-commodity network flows*. Operations Research, pages 344–360, 1963.

[Hungerford 97] T.W. Hungerford. Algebra. Springer-Verlag, 1997.

[Jain 03] K. Jain, M. Mahdian & M. R. Salavatipour. *Packing Steiner Trees*. In Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms, pages 266–274, 2003.

[Karamchandani 09] N. Karamchandani, R. Appuswamy & M. Franceschetti. *Distributed Computation of Symmetric Functions with Binary Inputs*. In Proceedings of the IEEE Information Theory Workshop, pages 76–80, 2009.

[Kempe 03] D. Kempe, A. Dobra & J. Gehrke. *Gossip-Based Computation of Aggregate Information*. In Proceedings of the forty-fourth annual IEEE Symposium on Foundations of Computer Science, pages 482–491, 2003.

[Koetter 03] R. Koetter & M. Médard. *An algebraic approach to network coding*. IEEE/ACM Transactions on Networking, vol. 11, no. 5, pages 782–795, 2003.

[Körner 79] J. Körner & K. Marton. *How to encode the modulo-two sum of binary sources*. IEEE Transactions on Information Theory, vol. 25, no. 2, pages 29–221, March 1979.

[Kowshik 09] H. Kowshik & P. R. Kumar. *Zero-Error Function Computation in Sensor Networks*. In Proceedings of the IEEE Conference on Decision and Control, pages 3787–3792, 2009.

[Kushilevitz 97] E. Kushilevitz & N. Nisan. Communication complexity. Cambridge University Press, 1997.

[Lehman 03] A. R. Lehman & E. Lehman. *Complexity classification of network information flow problems*. In Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms, pages 142–150, 2003.

[Leighton 99] T. Leighton & S. Rao. *Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms*. Journal of the ACM, vol. 46, no. 6, pages 787–832, November 1999.

[Li 03] S. R. Li, R. W. Yeung & N. Cai. *Linear network coding*. IEEE Transactions on Information Theory, vol. 49, 2003.

[Ma 08] N. Ma & P. Ishwar. *Two-terminal distributed source coding with alternating messages for function computation*. In Proceedings of the IEEE International Symposium on Information Theory, pages 51–55, 2008.

[Ma 09] N. Ma, P. Ishwar & P. Gupta. *Information-Theoretic Bounds for Multiround Function Computation in Collocated Networks*. In Proceedings of the IEEE International Symposium on Information Theory, pages 2306–2310, 2009.

[Menger 27] K. Menger. *Zur allgemeinen Kurventheorie*. Fund. Math., page 96Ű115, 1927.

[Mosk-Aoyama 08] D. Mosk-Aoyama & D. Shah. *Fast Distributed Algorithms for Computing Separable Functions*. IEEE Transactions on Information Theory, vol. 54, no. 7, pages 2997–3007, July 2008.

[Nazer 07] B. Nazer & M. Gastpar. *Computing over Multiple-Access Channels*. IEEE Transactions on Information Theory, vol. 53, no. 10, pages 3498–3516, October 2007.

[Nebe 06] G. Nebe, E. M. Rains & N. J. A. Sloane. Self-dual codes and invariant theory. Springer, 2006.

[Ngai 04] C. K. Ngai & R. W. Yeung. *Network coding gain of combination networks*. In Proceedings of the IEEE Information Theory Workshop, pages 283–287, 2004.

[Orlitsky 01] A. Orlitsky & J. R. Roche. *Coding for computing*. IEEE Transactions on Information Theory, vol. 47, no. 3, pages 903–917, March 2001.

[Paek 09] J. Paek, B. Greenstein, O. Gnawali, K. Jang, A. Joki, M. Vieira, J. Hicks, D. Estrin, R. Govindan & E. Kohler. *The tenet architecture for tiered sensor networks*. ACM Transactions on Sensor Networks, 2009.

[Rai 09] B. K. Rai & B. K. Dey. *Feasible alphabets for communicating the sum of sources over a network*. In Proceedings of the IEEE International Symposium on Information Theory, pages 1353–1357, 2009.

[Rai 10a] B. K. Rai & B. K. Dey. *Sum-networks: System of polynomial equations, unachievability of coding capacity, reversibility, insufficiency of linear network coding*. available at *http://arxiv.org/abs/0906.0695*, 2010.

[Rai 10b] B. K. Rai, B. K. Dey & S. Shenvi. *Some bounds on the capacity of communicating the sum of sources*. In ITW 2010, Cairo, 2010.

[Ramamoorthy 08] A. Ramamoorthy. *Communicating the sum of sources over a network*. In Proceedings of the IEEE International Symposium on Information Theory, pages 1646–1650, 2008.

[Schwartz 80] J. T. Schwartz. *Fast probabilistic algorithms for verification of polynomial identities*. J. ACM., vol. 27, pages 701–717, 1980.

[Subramanian 07] S. Subramanian, P. Gupta & S. Shakkottai. *Scaling Bounds for Function Computation over Large Networks*. In Proceedings of the IEEE International Symposium on Information Theory, pages 136–140, 2007.

[Vazirani 04] V. V. Vazirani. Approximation algorithms. Springer, first edition, 2004.

[West 01] D. B. West. Introduction to graph theory. Prentice-Hall, 2001.

[Witsenhausen 76] H. Witsenhausen. *The zero-error side information problem and chromatic numbers*. IEEE Transactions on Information Theory, vol. 22, no. 5, pages 592–593, September 1976.

[Yamamoto 82] H. Yamamoto. *Wyner - Ziv theory for a general function of the correlated sources*. IEEE Transactions on Information Theory, vol. 28, no. 5, pages 803–807, September 1982.

[Yao 79] A. C. Yao. *Some complexity questions related to distributive computing*. In Proceedings of the eleventh annual ACM Symposium on Theory of Computing, pages 209–213, 1979.

[Yeung 02] R. W. Yeung. A first course in information theory. Springer, 2002.

[Ying 07] L. Ying, R. Srikant & G. E. Dullerud. *Distributed Symmetric Function Computation in Noisy Wireless Sensor Networks*. IEEE Transactions on Information Theory, vol. 53, no. 12, pages 4826–4833, December 2007.