Linear Codes, Target Function Classes, and Network Computing Capacity

Rathinakumar Appuswamy, *Member, IEEE*, Massimo Franceschetti, *Senior Member, IEEE*, Nikhil Karamchandani, *Member, IEEE*, and Kenneth Zeger, *Fellow, IEEE*

***Abstract*—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, semi-injective, and linear target functions over finite fields. Computing capacity bounds and achievability are given with respect to these target function classes for network codes that use routing, linear coding, or nonlinear coding.**

***Index Terms*—Capacity, cut-set bound, function computation, information theory, network coding.**

I. Introduction

Network coding [1] is concerned with 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 [1], [10], [11], whether linear codes are sufficient to achieve the capacity [6], [7], [14], [16], and cut-set upper bounds on the capacity and the tightness of such bounds [10], [11], [23].

Network computing, on the other hand, considers a more general problem in which each receiver node demands a target function of the source messages [4], [8], [15], [17], [21], [22]. 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 [4], 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. Analogous to coding capacity for network coding, the notion of computing capacity was defined for network computing as the supremum of achievable rates of computing the network's target function, i.e., the maximum number of times f can be computed per network usage. The objective was to characterize the computing capacity for any given network and target function. 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. 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 (see, e.g., [19]).

In contrast to [4], this paper focuses on *linear network computing*, where the encoding operations performed by the nodes in the network are restricted to be linear. Specifically, we compare the linear computing capacity with the (nonlinear) computing capacity and the routing computing capacity for different classes of target functions in single-receiver networks. Such classes include reducible, 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 each of the above target function classes and for network codes that use routing, linear coding, or nonlinear coding.

The performance of linear codes has been studied previously in the context of network coding. For example, it is known that linear codes are sufficient to achieve the coding capacity for multicast networks [1], but they are not sufficient in general to achieve the coding capacity for nonmulticast networks [6]. In the context of 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 [20]. However, it has been shown that for single-receiver networks, linear coding is sufficient for solvability when computing a scalar linear target function [3], [21].

Our specific contributions will be summarized next.

A. Contributions

Section II gives many of the formal definitions used in this paper (e.g., target function classes and computing capacity types). In Sections III and IV, 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 semi-injective, reducible, and linear functions. The relationships between these classes is illustrated in Fig. 1. Throughout this paper, we emphasize the main results as theorems, while the other results are stated as propositions.

Section III studies the performance of linear codes for network computing. We show that if a target function is not reducible, then the linear computing capacity and routing com-

Manuscript received May 07, 2011; revised October 12, 2012; accepted January 28, 2013. Date of publication March 08, 2013; date of current version August 14, 2013. This work was supported by the National Science Foundation and the UCSD Center for Wireless Communications. This paper was presented in part at the 2009 IEEE International Symposium on Information Theory and in part at the 2011 IEEE International Symposium on Information Theory.

R. Appuswamy is with IBM Research Almaden, San Jose, CA 95120 USA (e-mail: rappusw@us.ibm.com).

M. Franceschetti and K. Zeger are with the Department of Electrical and Computer Engineering, University of California, San Diego, La Jolla, CA 92093-0407 USA (e-mail: massimo@ece.ucsd.edu; zeger@ucsd.edu).

N. Karamchandani is with the Department of Electrical Engineering, University of California, Los Angeles, CA 90095-1594 USA (e-mail: nikhil@ee.ucla.edu).

Communicated by M. Gastpar, Associate Editor for Shannon Theory.

Digital Object Identifier 10.1109/TIT.2013.2252052

TABLE I
SUMMARY OF OUR MAIN RESULTS FOR CERTAIN CLASSES OF TARGET FUNCTIONS The quantities $\mathcal{C}_{\rm cod}(\mathcal{N},f), \mathcal{C}_{\rm lin}(\mathcal{N},f)$, and $\mathcal{C}_{\rm rout}(\mathcal{N},f)$ 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 $\mathcal{A}$ indicate constraints on the target function f and the source alphabet $\mathcal{A}$, respectively.

Result	f	$\mathcal{A}$	Location
$\forall f\ \forall \mathcal{N}\ \mathcal{C}_{\rm lin}(\mathcal{N},f)=\mathcal{C}_{\rm rout}(\mathcal{N},f)$	non-reducible	field	Theorem III.6(a)
	semi-injective	ring	
$\forall f\ \exists \mathcal{N}\ \mathcal{C}_{\rm lin}(\mathcal{N},f)>\mathcal{C}_{\rm rout}(\mathcal{N},f)$	reducible	ring	Theorem III.6(b)
$\forall f\ \exists \mathcal{N}\ \mathcal{C}_{\rm cod}(\mathcal{N},f)>\mathcal{C}_{\rm lin}(\mathcal{N},f)$	non-injective & non-reducible	field	Proposition III.7
$\exists f\ \exists \mathcal{N}\ \mathcal{C}_{\rm cod}(\mathcal{N},f)>\mathcal{C}_{\rm lin}(\mathcal{N},f)>\mathcal{C}_{\rm rout}(\mathcal{N},f)$	reducible	ring	Proposition III.8
$\forall f\ \forall \mathcal{N}\ \mathcal{C}_{\rm cod}(\mathcal{N},f)=\mathcal{C}_{\rm lin}(\mathcal{N},f)$	linear	field	Theorem IV.5

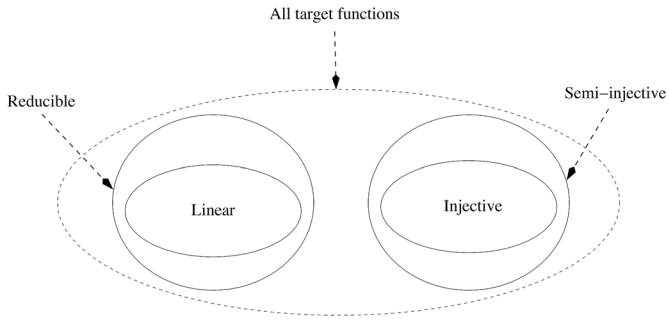

Fig. 1. Decomposition of the space of all target functions into various classes.

puting capacity are equal whenever the source alphabet is a finite field [see Theorem III.6(a)]; the same result also holds for semi-injective target functions over rings. Conversely, if a target function is reducible, then there always exists a network where the linear computing capacity is larger than the routing computing capacity [see Theorem III.6(b)]. Thus, the advantage of linear coding over routing for network computing is strongly influenced by whether the target function is reducible or not. Propositions III.7 and III.8 investigate the benefit of nonlinear codes over linear codes for computing reducible and nonreducible target functions and show that in both cases, there exist networks for which the (nonlinear) computing capacity is greater than the linear computing capacity. In particular, Proposition III.8 shows that even if the target function is reducible, linear codes may not achieve the full (nonlinear) computing capacity of a network. However, as we show in the next section, there are reducible target functions for which linear codes are indeed optimal.

Section IV focuses on computing linear target functions over finite fields, which form a subclass of reducible target functions. Thus, from Theorem III.6(b), there may be a computing capacity gain in using linear codes over routing. In this section, we show that for linear target functions over finite fields, linear network codes in fact achieve the full (nonlinear) computing capacity in an arbitrary network (see Theorem IV.5). We note that this result was obtained independently in [20] and [21].

In Section V, we study an example network that illustrates various concepts discussed in the previous sections and also provides some interesting additional results for network computing. Specifically, we study the reverse butterfly network—obtained by reversing the direction of all the edges in the multicast butterfly network (the butterfly network studied in [1] 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 (see Proposition V.1) and the computing capacity (see Theorem V.2). 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.

Finally, in Appendix A, we demonstrate that the performance of optimal linear codes may depend on how "linearity" is defined (see Proposition VI.2). Specifically, we show that the linear computing capacity of a network varies depending on the particular ring over which linearity is defined on the source alphabet.

Our main results are summarized in Table I.

II. 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 $s=|S|$ 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 $\mathcal{E}_{\rm in}(u)$ and $\mathcal{E}_{\rm out}(u)$ 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 ith component of x. For any index set $I=\{i_1,i_2,\ldots,i_q\}\subseteq\{1,2,\ldots,m\}$ with $i_1<i_2<\cdots<i_q$, 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 $\mathbb{F}_q$ denote a finite field of order q. A superscript t will denote the transpose for vectors and matrices.

A. Target Functions

A *target function* is a mapping

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

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

TABLE II
DEFINITIONS OF SOME TARGET FUNCTIONS

Target function f	Alphabet $\mathcal{A}$	$f(x_1, \ldots, x_s)$	Comments
identity	arbitrary	$(x_1, \ldots, x_s)$	$\mathcal{B} = \mathcal{A}^s$
arithmetic sum	$\{0, 1, \ldots, q-1\}$	$x_1 + x_2 + \cdots + x_s$	'+' is ordinary integer addition, $\mathcal{B} = \{0, 1, \cdots, s(q-1)\}$
mod r sum	$\{0, 1, \ldots, q-1\}$	$x_1 \oplus x_2 \oplus \ldots \oplus x_s$	$\oplus$ is mod r addition, $\mathcal{B} = \mathcal{A}$
linear	any ring	$a_1x_1 + a_2x_2 + \ldots + a_sx_s$	arithmetic in the ring, $\mathcal{B} = \mathcal{A}$
maximum	any ordered set	$\max\{x_1, \ldots, x_s\}$	$\mathcal{B} = \mathcal{A}$

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 II.

Definition II.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 $x \in \mathcal{A}^s$,

$$g(xT) = f(x). \tag{1}$$

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 II.2: Suppose the alphabet is $\mathcal{A} = \mathbb{F}_2$ and the target function is

$$f : \mathbb{F}_2^3 \longrightarrow \{0, 1\}$$

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 $g(y_1, y_2) = y_1y_2$, 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 II.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 III.2 and Lemma III.3) to be nonreducible.

B. 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 individual components

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

are called *messages*.[2]

Definition II.4: A (k, n) *network code in a network* $\mathcal{N}$ consists of the following:

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

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

$$h^{(e)} : \prod_{\hat{e} \in \mathcal{E}_{\text{in}}(v)} \mathcal{A}^n \longrightarrow \mathcal{A}^n \qquad \text{otherwise.}$$

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

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

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.

Definition II.5: Suppose in a network $\mathcal{N}$, the in-edges of the receiver are $e_1, e_2, \ldots, e_{|\mathcal{E}_{\text{in}}(\rho)|}$. 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

$$\psi\left(z_{e_1}, \ldots, z_{e_{|\mathcal{E}_{\text{in}}(\rho)|}}\right)_j = f\Big(\left(\alpha(\sigma_1)_j, \ldots, \alpha(\sigma_s)_j\right)\Big). \tag{2}$$

[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.

In words, the receiver constructs a vector of k alphabet symbols, such that for each $i \in \{1, 2, \ldots, k\}$, the ith component of the receiver's computed vector equals[4] the value of the desired target function f, applied to the ith components of the source message vectors, for any choice of message generator α. 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 [5]. We use an analogous definition for the computing capacity.

Definition II.6: The *computing capacity* of a network $\mathcal{N}$ with respect to a target function f is

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

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 II.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 $\mathcal{A}$.

Remark II.8: Note that Definition II.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 (see Theorem IV.5).

Definition II.9: The *linear computing capacity* of a network $\mathcal{N}$ with respect to target function f is

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

Remark II.10: 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 and over which the encoding operations are linear. In Appendix A, we illustrate this possibility with a specific computing problem.

The *routing computing capacity* $\mathcal{C}_{\mathrm{rout}}(\mathcal{N}, f)$ is defined similarly by restricting the encoding functions to routing, i.e., at each node, the codeword sent over an out-edge consists of symbols either received by the node, or generated by it if it is a source. We call the quantity $\mathcal{C}_{\mathrm{cod}}(\mathcal{N}, f) - \mathcal{C}_{\mathrm{lin}}(\mathcal{N}, f)$ the *computing capacity gain* of using nonlinear coding over linear coding. Similar "gains," such as $\mathcal{C}_{\mathrm{cod}}(\mathcal{N}, f) - \mathcal{C}_{\mathrm{rout}}(\mathcal{N}, f)$ and $\mathcal{C}_{\mathrm{lin}}(\mathcal{N}, f) - \mathcal{C}_{\mathrm{rout}}(\mathcal{N}, f)$ are defined.

Definition II.11: 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 $\mathcal{N}$.

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 [23]. Let $\mathcal{C}_{\mathrm{rout}}(\mathcal{N})$ 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 [23, Theorem 4.2] that for any single-receiver network $\mathcal{N}$:

$$\mathcal{C}_{\mathrm{rout}}(\mathcal{N}) = \min_{C \in \Lambda(\mathcal{N})} \frac{|C|}{|I_C|}. \tag{3}$$

Remark II.12: Recall that f depends on every source in the network nontrivially. It is easy to see that if the intermediate nodes in a network are restricted to perform routing, then every component of every source message must be received by ρ in order to compute f. Since any routing code that computes the identity target function can also be used to compute any target function f, we have

$$\mathcal{C}_{\mathrm{rout}}(\mathcal{N}, f) = \mathcal{C}_{\mathrm{rout}}(\mathcal{N}). \tag{4}$$

This motivates the use of coding for computing functions in networks.

III. Linear Network Codes for Computing Target Functions

Remark II.12 implies 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 III-A, 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 computing 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 computing capacity whenever the target function is noninjective. Hence, if the target function is not reducible, such capacity gain must be obtained from nonlinear coding. Section III-A also 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.

Note that, in general, the linear computing capacity would depend on the algebraic structure imposed on the alphabet. Proposition VI.2 in the Appendix illustrates this through an example where linear computing capacity is evaluated for computing the same target function over a network for two different ring alphabets of cardinality 4. It is shown that the linear computing capacity over one ring is strictly greater than the other.

[4]Note that the computation has to be performed with zero-error.

A. Reducible and Nonreducible 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 III.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 $f^{-1}(\{f(x)\}) = \{x\}$.

Example III.2: All injective functions are semi-injective. If f is the arithmetic sum target function (recall the definition from Table II), 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.

The following lemma follows easily from the definition of reducible functions in Definition II.1.

Lemma III.3: If alphabet $\mathcal{A}$ is a ring, then semi-injective target functions are not reducible.

Lemma III.4 states an alternative characterization of reducible target functions when the source alphabet is a finite field and will be used in Theorem III.6. A proof of the lemma is relegated to Appendix C.

Lemma III.4: Let $\mathcal{N}$ be a network with target function $f : \mathcal{A}^s \longrightarrow \mathcal{B}$ and a finite field alphabet $\mathcal{A}$. The function f is reducible if and only if there exists a nonzero $d \in \mathcal{A}^s$ such that for each $a \in \mathcal{A}$ and each $x \in \mathcal{A}^s$

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

Note that if the alphabet $\mathcal{A}$ is not a finite field, then the assertion in Lemma III.4 may not be true; the following example illustrates this point.

Example III.5: 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 $a \in \mathcal{A}$:

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

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

Theorem III.6 establishes that for a network with a finite field alphabet and a target function which is not reducible, the 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 multireceiver networks have a larger linear capacity than their routing capacity [1]. However, all single-receiver networks are known to achieve their coding capacity with routing [23]. For network computing, the next theorem shows that with nonreducible target functions there is no advantage to using linear coding over routing.[5]

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

Theorem III.6: Let $f : \mathcal{A}^s \longrightarrow \mathcal{B}$ be a target function.

(a) Let $\mathcal{N}$ be a network with source 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

$$C_{\text{lin}}(\mathcal{N}, f) = C_{\text{rout}}(\mathcal{N}, f)\,.$$

(b) Let $\mathcal{A}$ be a ring. If f is reducible, then there exists a network $\mathcal{N}$ with source alphabet $\mathcal{A}$ such that

$$C_{\text{lin}}(\mathcal{N}, f) > C_{\text{rout}}(\mathcal{N}, f)\,.$$

Proof of (a)*:* Since any routing code is, in particular, a linear code

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

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

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

$$\sum_{i=1}^{s} w^{(i)} M(e)_i = \sum_{i=1}^{s} y^{(i)} M(e)_i. \tag{5}$$

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 mth row of $w - y$, then $d \neq 0$. For the case where $\mathcal{A}$ is a field and f is not reducible, by Lemma III.4, there exist $a \in \mathcal{A}$ and $x \in \mathcal{A}^s$ such that $ad \neq 0$ and

$$f(ad + x) \neq f(x). \tag{6}$$

In the case where $\mathcal{A}$ is a ring with identity and f is semi-injective, there exists an $x \in \mathcal{A}^s$ such that

$$\{x\} = f^{-1}(\{f(x)\})$$

which, in turn, is true if and only if for each nonzero $d \in \mathcal{A}^s$, we have $f(d + x) \neq f(x)$. So (6) still holds with $a = 1$.

Let u be any $k \times s$ matrix over $\mathcal{A}$ whose mth row is x and let $v = u + a(w - y)$. From (6), the target function f differs on the mth rows of u and v. Thus, the vectors on the in-edges of the

receiver ρ must differ between two cases: (1) when the source messages are the columns of u, and (2) when the source 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 jth 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

$$\begin{aligned}&\sum_{i=1}^{s} u^{(i)}M(e)_i \\ &= \sum_{i=1}^{s} u^{(i)}M(e)_i + a\sum_{i=1}^{s}(w^{(i)} - y^{(i)}))M(e)_i \quad [\text{from (5)}] \\ &= \sum_{i=1}^{s} v^{(i)}M(e)_i \end{aligned} \tag{7}$$

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 jth 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)] 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

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

Proof of (b)*:* 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

$$g(xT) = f(x) \text{ for every } x \in \mathcal{A}^s \text{ [from Definition II.1]}. \tag{8}$$

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

$$\begin{aligned}\psi(xT) &= g(xT) && [\text{from } \psi = g] \\ &= f(x) && [\text{from (8)}].\end{aligned}$$

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

$$\begin{aligned}\frac{k}{n} &= \frac{1}{\lambda} \\ &> 1/s && [\text{from } \lambda \leq s-1] \\ &= \mathcal{C}_{\text{rout}}(\mathcal{N}) && [\text{from (8)}]\end{aligned}$$

which is sufficient to establish the claim. ■

Theorem III.6(a) showed that there cannot be linear computing gain for networks whose target functions are not reducible. Conversely, Theorem III.6(b) shows 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. Next, Propositions III.7 and III.8 investigate the benefit of nonlinear codes over linear codes for both classes of target functions and show that there exist networks for which the (nonlinear) computing capacity is greater than the linear computing capacity.

Proposition III.7: 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

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

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

$$\begin{aligned}&\mathcal{C}_{\text{lin}}(\mathcal{N}, f) \\ &= 1/s && [\text{from Theorem III.6(a), (3)}] \\ &< \min\left\{1,\ \tfrac{1}{\log_{|\mathcal{A}|}|f(\mathcal{A}^s)|}\right\} && [\text{from } s \geq 2,\ |f(\mathcal{A}^s)| < |\mathcal{A}|^s] \\ &\leq \mathcal{C}_{\text{cod}}(\mathcal{N}, f) && [\text{from Lemma VI.3}].\end{aligned}$$

We note that the last inequality also follows from [4, Theorem III.3] which characterizes the computing capacity for any target function over a tree network. ■

The same proof as above also holds if the alphabet $\mathcal{A}$ is a ring with identity and the target function f is semi-injective but not injective. Note that the above result does not hold for injective functions. This is easy to see since the problem of computing injective functions is essentially equivalent to the network coding problem of recovering all the source messages at the receiver and, as mentioned before, all single-receiver networks are known to achieve their coding capacity with just routing [23].

Proposition III.8: There exists a network $\mathcal{N}$ and a reducible target function f such that

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

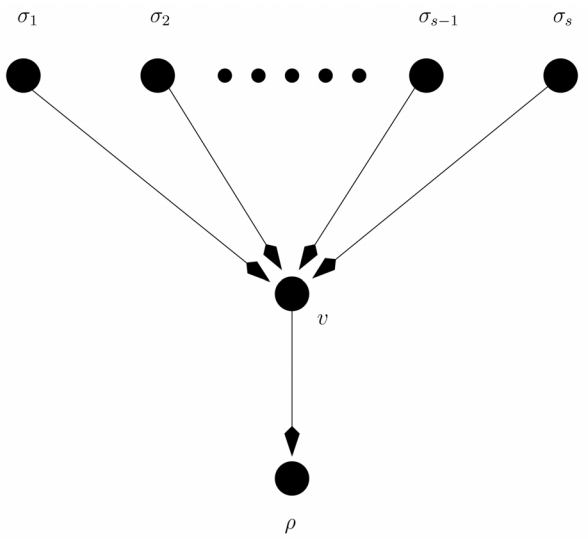

Fig. 2. Network $\mathcal{N}_{2,s}$ has sources $\sigma_1, \sigma_2, \ldots, \sigma_s$, each connected to the relay v by an edge and v is connected to the receiver by an edge.

Proof: Let $\mathcal{N}$ denote the network $\mathcal{N}_{2,3}$ shown in Fig. 2 with $s = 3$, alphabet $\mathcal{A} = \mathbb{F}_2$, and let f be the target function in Example II.3.

The routing computing capacity is given by

$$\mathcal{C}_{\text{rout}}(\mathcal{N}, f) = 1/3 \quad [\text{from (3)}]. \tag{9}$$

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

$$\mathcal{C}_{\text{cod}}(\mathcal{N}, f) \geq 1. \tag{10}$$

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

$$(\, \alpha(\sigma_1) \quad \alpha(\sigma_2) \quad \alpha(\sigma_3)\,)\ M$$

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

$$(\, \alpha(\sigma_1) \quad \mathbf{0} \quad \alpha(\sigma_2)\,)\ M$$

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),

$$\begin{aligned} \frac{k}{n} &\leq \mathcal{C}_{\text{lin}}(\mathcal{N}_{2,2}, g) \\ &= \mathcal{C}_{\text{rout}}(\mathcal{N}_{2,2}, g) \qquad [\text{from Theorem III.6(a)}] \\ &= 1/2 \qquad\qquad [\text{from (3)}]. \end{aligned}$$

Consequently

$$\mathcal{C}_{\text{lin}}(\mathcal{N}, f) \leq 1/2. \tag{11}$$

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

$$(\, \alpha(\sigma_1) \quad \alpha(\sigma_2) \quad \alpha(\sigma_3)\,)\ M$$

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 (11), we get

$$\mathcal{C}_{\text{lin}}(\mathcal{N}, f) = 1/2. \tag{12}$$

The claim of the theorem now follows from (9), (10), and(12). ■

The above proposition shows that, even if the target function is reducible, linear codes may not achieve the full (nonlinear) computing capacity of a network. However, as we will see in the next section, there are reducible target functions for which linear codes are indeed optimal.

IV. Computing Linear Target Functions

Recall from Table II that for any ring $\mathcal{A}$, a linear target function is defined as $f(x) = a_1 x_1 + a_2 x_2 + \cdots + a_s x_s$ for any $x \in \mathcal{A}^s$, with $a_i \in \mathcal{A}$ for all i and arithmetic over the ring. We showed in the previous section that for reducible target functions, there may be a computing capacity gain in using linear codes over routing. In this section, we show that for a special subclass of reducible target functions, namely linear target functions over finite fields, linear network codes achieve the full (nonlinear) computing capacity. As mentioned before, this result has been obtained independently in [20] and [21]. 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

$$\alpha(\sigma) = (\alpha(\sigma)_1, \alpha(\sigma)_2, \ldots, \alpha(\sigma)_c) \in \mathbb{F}_{q^k}$$

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

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

$$\begin{cases} \sum_{\hat{e} \in \mathcal{E}_{\text{in}}(u)} \gamma_{\hat{e}}^{(e)} z_{\hat{e}} + \sum_{j=1}^{c} \beta_j^{(e)} \alpha(u)_j, & \text{if } u \in S \\ \sum_{\hat{e} \in \mathcal{E}_{\text{in}}(u)} \gamma_{\hat{e}}^{(e)} z_{\hat{e}}, & \text{otherwise} \end{cases} \tag{13}$$

for some $\gamma_{\hat{e}}^{(e)}, \beta_j^{(e)} \in \mathbb{F}_{q^n}$ and we use a decoding function ψ whose jth component output ψ_j is

$$\sum_{e \in \mathcal{E}_{\text{in}}(\rho)} \delta_j^{(e)} z_e \quad \text{for all } j \in \{1, 2, \ldots, c\} \tag{14}$$

for certain $\delta_j^{(e)} \in \mathbb{F}_{q^n}$. 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 $e_j \in \mathcal{E}_{\text{out}}(\sigma)$, let $x_1^{(e_j)}, \ldots, x_c^{(e_j)}$ be variables, and for each $e_j \in \mathcal{E}_{\text{in}}(\rho)$, 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 shorthand notation $\mathbb{F}[y]$ to mean the ring of polynomials $\mathbb{F}[\ldots, y_{e_i}^{(e_j)}, \ldots]$ and similarly for $\mathbb{F}[x, y, w]$.

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

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

$$(A_\tau(x))_{i,j} = \begin{cases} x_i^{(e_j)}, & \text{if } e_j \in \mathcal{E}_{\text{out}}(\sigma_\tau) \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

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

$$(F(y))_{i,j} = \begin{cases} y_{e_i}^{(e_j)}, & \text{if } e_i, e_j \in \mathcal{E} \text{ and } head(e_i) = tail(e_j) \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

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

$$(B(w))_{i,j} = \begin{cases} w_i^{(e_j)}, & \text{if } e_j \in \mathcal{E}_{\text{in}}(\rho) \\ 0, & \text{otherwise.} \end{cases} \tag{17}$$

Consider an (nc, n) linear code of the form in (13) and (14).

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 IV.1. Let I denote the identity matrix of suitable dimension.

Lemma IV.1. (see [14, Lemma 2]): The matrix $I - F(y)$ is invertible over the ring $\mathbb{F}_q[y]$.

Lemma IV.2. (see [14, Theorem 3]): For $s = 1$ and for all $\tau \in \{1, \ldots, s\}$, the decoder in (14) satisfies

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

Lemma IV.3. (see [2, Theorem 1.2]): Let $\mathbb{F}$ be an arbitrary field, and let $g = g(x_1, \ldots, x_m)$ be a polynomial in $\mathbb{F}[x_1, \ldots, x_m]$. 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 $\mathbb{F}$ 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

$$M_\tau(x, y, w) = A_\tau(x)(I - F(y))^{-1} B(w)^t \tag{18}$$

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

Lemma IV.4: If for all $\tau \in \{1, 2, \ldots, s\}$,

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

in the ring $\mathbb{F}_q[x, y, w]$, then there exists an integer $n > 0$ and vectors β, γ, δ over $\mathbb{F}_{q^n}$ 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 $\mathbb{F}_{q^n}$.

Proof: The quantity

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

is a nonzero polynomial in $\mathbb{F}_q[x, y, w]$ and therefore also in $\mathbb{F}_{q^n}[x, y, w]$ 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 IV.3 implies there exist vectors β, γ, δ (whose components correspond to the components of the vector variables x, y, w) over $\mathbb{F}_{q^n}$ such that

$$\det\left(\prod_{\tau=1}^{s} M_\tau(\beta, \gamma, \delta)\right) \neq 0 \tag{19}$$

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. ■

Theorem IV.5: Let the alphabet $\mathcal{A}$ be the finite field $\mathbb{F}_q$. If $\mathcal{N}$ is a network with a linear target function f over $\mathbb{F}_q$ that depends on every source nontrivially, then

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

Proof: We have

$$\begin{aligned} \mathcal{C}_{\text{lin}}(\mathcal{N}, f) &\leq \mathcal{C}_{\text{cod}}(\mathcal{N}, f) \\ &\leq \min_{C \in \Lambda(\mathcal{N})} |C| \end{aligned} \tag{20}$$

where the last inequality follows immediately from [4, Theorem II.1] and by noting that the quantity $R_{I_{C,f}}$, defined in [4, Definition 1.5], is equal to $|\mathcal{A}|$ for linear target functions. 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 $\alpha = \min_{C \in \Lambda(\mathcal{N})} |C|$.

From Lemma IV.1, the matrix $I - F(y)$ is invertible over the ring $\mathbb{F}_q[x, y, w]$ and therefore also over $\mathbb{F}_{q^n}[x, y, w]$. Since any minimum cut between the source σ_τ and the receiver ρ has at least c edges, it follows from [14, Theorem 2][6] that $\det(M_\tau(x, y, w)) \neq 0$ for every $\tau \in \{1, 2, \ldots, s\}$. From Lemma IV.4, there exists an integer $n > 0$ and vectors β, γ, δ over $\mathbb{F}_{q^n}$ 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

$$\hat{A}_\tau(\beta) = a_\tau \left(M_\tau(\beta, \gamma, \delta)\right)^{-1} A_\tau(\beta). \tag{21}$$

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 $\mathbb{F}_{q^n}$ computed by the receiver ρ is

$$\begin{aligned}
\psi &= \sum_{\tau=1}^{s} \alpha(\sigma_\tau)\, \hat{A}_\tau(\beta)(I - F(\gamma))^{-1} B(\delta)^t \\
&\qquad\qquad [\text{from Lemma IV.2 and linearity}] \\
&= \sum_{\tau=1}^{s} \alpha(\sigma_\tau)\, a_\tau \left(M_\tau(\beta, \gamma, \delta)\right)^{-1} A_\tau(\beta)(I - F(\gamma))^{-1} B(\delta)^t \\
&\qquad\qquad [\text{from (21)}] \\
&= \sum_{\tau=1}^{s} a_\tau\, \alpha(\sigma_\tau) \qquad [\text{from (18)}] \\
&= (f(\alpha(\sigma_1)_1, \ldots, \alpha(\sigma_s)_1), \ldots, f(\alpha(\sigma_1)_c, \ldots, \alpha(\sigma_s)_c))
\end{aligned}$$

which proves that the linear code achieves a computing rate of α. Along with (20), this lower bound proves the theorem statement. ■

Theorem IV.5 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.

V. Reverse Butterfly Network

In this section, we study an example network that illustrates various concepts discussed previously in this paper and also provides some interesting additional results for network computing. The network $\mathcal{N}_3$ shown in Fig. 3(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 Fig. 3(a).

Proposition V.1: Let the alphabet $\mathcal{A}$ be $\{0, 1, \ldots, q-1\}$. The routing computing capacity and the linear computing capacity

[6]Using the implication (1) $\Longrightarrow$ (3) in [14, Theorem 2].

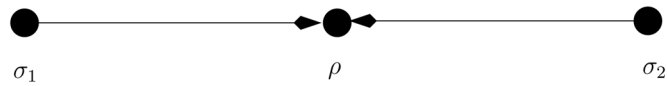

Fig. 3. Network where there is no benefit to using linear coding over routing for computing f.

(over the ring $\mathbb{Z}_q$) of the reverse butterfly network $\mathcal{N}_3$ with the arithmetic sum target function

$$f : \mathcal{A}^2 \longrightarrow \{0, 1, \ldots, 2(q-1)\}$$

are

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

Proof: Since f is a nonreducible target function from Example III.2 and Lemma III.3, we have

$$\begin{aligned}
\mathcal{C}_{\text{lin}}(\mathcal{N}_3, f) &= \mathcal{C}_{\text{rout}}(\mathcal{N}_3, f) \qquad [\text{from Theorem III.6(a)}] \\
&= 1. \qquad\qquad [\text{from (3), (4)}]
\end{aligned}$$

■

Theorem V.2: Let the alphabet $\mathcal{A}$ be $\{0, 1, \ldots, q-1\}$. The computing capacity of the reverse butterfly network $\mathcal{N}_3$ with the arithmetic sum target function

$$f : \mathcal{A}^2 \longrightarrow \{0, 1, \ldots, 2(q-1)\}$$

is

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

Remark V.3: The computing capacity $\mathcal{C}_{\text{cod}}(\mathcal{N}_3, f)$ obtained in Theorem V.2 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 computingcapacity are known to be independent of the coding alphabet used [5]. 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 V.4: The ratio of the coding capacity to the routing capacity for the multicast butterfly network with two messages was computed in [5] 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. Thus, this is also an illustration of Proposition III.7.

Remark V.5: Recall that capacity is defined as the supremum of the set of rational numbers k/n such that a (k, n) code that computes the target function exists. It was pointed out in [5] that it remains an open question whether the coding capacity of a network can be irrational. Our Theorem V.2 demonstrates that the computing capacity of a network (e.g., the reverse butterfly network) with unit capacity links can be irrational when

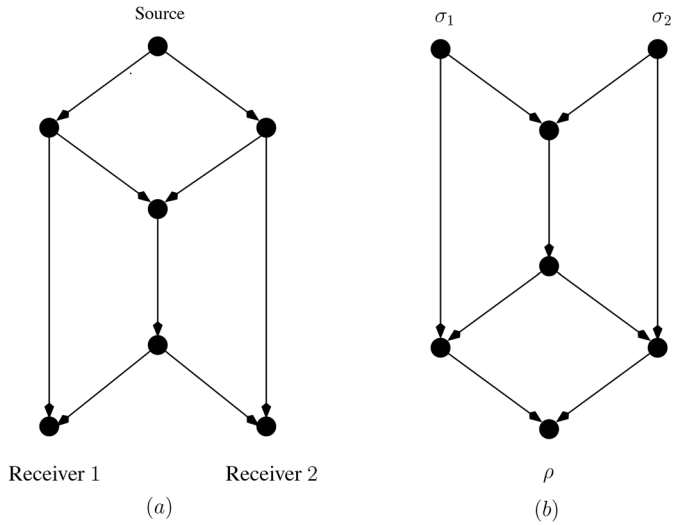

Fig. 4. Butterfly network and its reverse $\mathcal{N}_3$.

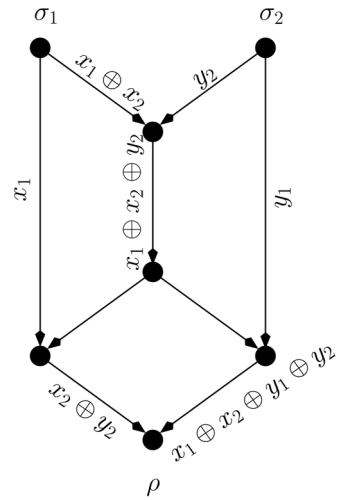

Fig. 5. Reverse butterfly network with a linear code over $\mathbb{Z}_q$ that computes the mod q sum target function.

the target function to be computed is the arithmetic sum target function of the source messages.

The following lemma is used to prove Theorem V.2. The code construction used below was obtained in [3] and is an instance of the class of codes proposed in [20].

Lemma V.6: Let the alphabet $\mathcal{A}$ be $\{0, 1, \ldots, q-1\}$. The computing capacity and the linear computing capacity (over the ring $\mathbb{Z}_q$) of the reverse butterfly network $\mathcal{N}_3$ with the mod q sum target function f are

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

Proof: The upper bound of 2 on $\mathcal{C}_{\text{cod}}(\mathcal{N}_3, f)$ follows from [4, Theorem II.1]. To establish the achievability part, we construct a linear code over the ring $\mathbb{Z}_q$. Let $k = 2$ and $n = 1$. Consider the code shown in Fig. 4, 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 linear code achieves a rate of 2.

Remark V.7: It follows from the definition of reducible functions in Definition II.1 that the $\mod q$ sum target function above is reducible. Thus, Theorem III.6(b) applies and there exist networks for which the linear computing capacity is strictly greater than the routing computing capacity. In fact, it is easy to check that the reverse butterfly network $\mathcal{N}_3$ is one such network since its routing computing capacity is 1 from Proposition V.1, whereas the linear computing capacity is 2 from the above result.

Proof of Theorem V.2: We have

$$\mathcal{C}_{\text{cod}}(\mathcal{N}, f) \leq 2/\log_q(2q-1) \text{ [from [4, TheoremII.1]]}.$$

To establish the lower bound, we use the fact that the 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 V.6 (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.

Fig. 5 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 Fig. 4 over $\hat{\mathcal{A}}$, and finally converts the result back to $\mathcal{A}$. Similarly, the receiver node T computes the componentwise arithmetic sum of the source message vectors $\alpha(\sigma_1)$ and $\alpha(\sigma_2)$ using

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

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

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

■

TABLE III
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

APPENDIX

A. Linear Coding Over Different Ring Alphabets

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 III. We consider different rings R of size 4 for $\mathcal{A}$ and evaluate the linear computing capacity of the network $\mathcal{N}_1$ shown in Fig. 6 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

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

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

$$z(\alpha(\sigma_1), \alpha(\sigma_2)) = \alpha(\sigma_1) M_1 + \alpha(\sigma_2) M_2. \tag{22}$$

Let $m_{i,1}, \ldots, m_{i,k}$ denote the row vectors of M_i, for $i \in \{1,2\}$. Throughout this section, we will denote the zero vector by $\mathbf{0}$ to distinguish it from the scalar 0 and the length of the vector will be clear from context.

Lemma VI.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 III, 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 the zero vector $\mathbf{0}$, then $\alpha(\sigma_1) = \alpha(\sigma_2) \in \{0,2\}^k$.

Proof: If $\alpha(\sigma_1) = \alpha(\sigma_2) = \mathbf{0}$, then ρ receives a $\mathbf{0}$ by (22) and must decode a 0 since $f((0,0)) = 0$ (from Table III). Thus, ρ always decodes a 0 upon receiving the zero vector $\mathbf{0}$. But $f((x_1,x_2)) = 0$ if and only if $x_1 = x_2$ (from Table III), so whenever ρ receives a $\mathbf{0}$, 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 $z(\alpha(\sigma_1), \alpha(\sigma_2)) = \mathbf{0}$ and $\alpha(\sigma_1)_j \notin \{0,2\}$ for some $j \in \{1,2,\ldots,k\}$. Since $\alpha(\sigma_1)_j$ is invertible in $\mathbb{Z}_4$ (it is either 1 or 3), we have from (22) that

$$\begin{aligned} & m_{1,j} \\ &= \sum_{\substack{i=1 \\ i \neq j}}^{k} -\alpha(\sigma_1)_j^{-1}\, \alpha(\sigma_1)_i\, m_{1,i} + \sum_{i=1}^{k} -\alpha(\sigma_1)_j^{-1}\, \alpha(\sigma_2)_i\, m_{2,i} \\ &= y^{(1)} M_1 + y^{(2)} M_2 \end{aligned} \tag{23}$$

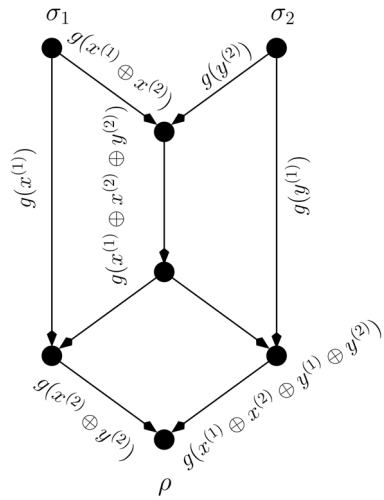

Fig. 6. Reverse butterfly network with a code that computes the arithmetic sum target function. $\bigoplus$ denotes $\bmod(2q-1)$ addition.

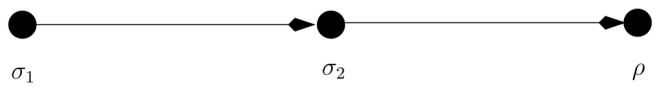

Fig. 7. Network $\mathcal{N}_1$ has two sources σ_1 and σ_2 and a receiver ρ .

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

$$\begin{aligned} y_i^{(1)} &= \begin{cases} -\alpha(\sigma_1)_j^{-1}\, \alpha(\sigma_1)_i, & if\ i \neq j \\ 0, & if\ i = j \end{cases} \\ y_i^{(2)} &= -\alpha(\sigma_1)_j^{-1}\, \alpha(\sigma_2)_i\,. \end{aligned} \tag{24}$$

Also, define the k-dimensional vector x by

$$x_i = \begin{cases} 0, & \text{if } i \neq j \\ 1, & \text{if } i = j. \end{cases} \tag{25}$$

We have from (22) that $z(x, \mathbf{0}) = m_{1,j}$, and from (22) and (23) 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

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

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

Proposition VI.2: The network $\mathcal{N}_1$ in Fig. 6 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 III satisfies

$$\mathcal{C}_{\text{lin}}(\mathcal{N}_1, f)^{\mathbb{Z}_4} \leq \frac{2}{3}$$
$$\mathcal{C}_{\text{lin}}(\mathcal{N}_1, f)^{\mathbb{Z}_2 \times \mathbb{Z}_2} = 1.$$

(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 (22), we have $z(x, \mathbf{0}) = \mathbf{0}$ whenever $xM_1 = \mathbf{0}$. Since $f((0,0)) \neq f((x_i, 0))$ (whenever $x_i \neq 0$), it must therefore be the case that $xM_1 = \mathbf{0}$ only when $x = \mathbf{0}$, or in other words, the rows of M_1 must be independent, so $n \geq k$. Thus

$$\mathcal{C}_{\text{lin}}(\mathcal{N}, f)^{\mathbb{Z}_2 \times \mathbb{Z}_2} \leq 1. \tag{26}$$

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 addition over $\mathcal{A}$. For any $x \in \mathcal{A}^2$, the value $f(x)$, as defined in Table III, 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 (26), 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 (22). Let $\mathcal{K}$ denote the collection of all message vector pairs $(\alpha(\sigma_1), \alpha(\sigma_2))$ such that $z(\alpha(\sigma_1), \alpha(\sigma_2)) = \mathbf{0}$. Define the $2k \times n$ matrix

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

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

Then

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

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

B. Lower Bound On the Computing Capacity

Lemma VI.3: The computing capacity of the network $\mathcal{N}_{2,s}$ shown in Fig. 2, with respect to a target function $f : \mathcal{A}^s \longrightarrow \mathcal{B}$, satisfies

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

Proof: Suppose

$$\log_{|\mathcal{A}|} |f(\mathcal{A}^s)| < 1. \tag{27}$$

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 (27)]. 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

$$\log_{|\mathcal{A}|} |f(\mathcal{A}^s)| \geq 1. \tag{28}$$

Then, any rate less than $1/(\log_{|\mathcal{A}|} |f(\mathcal{A}^s)|)$ can be achieved as follows. Since the desired rate is less than 1, we can still assume that each source sends its message vector to node v. Since representing $f(\mathcal{A}^s)$ needs $\log_{|\mathcal{A}|} |f(\mathcal{A}^s)|$ symbols from the alphabet $\mathcal{A}$, node v can communicate it to the receiver ρ at any rate less than $1/(\log_{|\mathcal{A}|} |f(\mathcal{A}^s)|)$. ■

C. Proof of Lemma III.4

Definition VI.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 VI.5:[7] If $\mathcal{A}$ is a finite field and $\mathcal{M}$ is a subspace of vector space $\mathcal{A}^s$, then $(\mathcal{M}^{\perp})^{\perp} = \mathcal{M}$.

Proof of Lemma III.4: Let g and T, respectively, be the map and matrix associated with the reducible function f, as indicated in Definition II.1. It then follows easily that there exists a nonzero $d \in \mathcal{A}^s$ such that $dT = 0$ and hence

$$f(ad + x) = g((ad + x)T) = g(xT) = f(x). \tag{29}$$

Conversely, suppose that there exists a nonzero d such that (29) holds for every $a \in \mathcal{A}$ and every $x \in \mathcal{A}^s$ and let $\mathcal{M}$ be the 1-D subspace of $\mathcal{A}^s$ spanned by d. Then

$$f(t + x) = f(x) \text{ for every } t \in \mathcal{M}, x \in \mathcal{A}^s. \tag{30}$$

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 $\mathcal{R}_T$

[7]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 (see, e.g., [12, Theorem 7.5] or [18, Corollary 3.2.3]). An analogous result for orthogonal complements over inner product spaces is well known in linear algebra (see, e.g., [13, Theorem 5, p. 286]).

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

$$g(y) = g(xT) = f(x). \tag{31}$$

Note that if $y = x^{(1)}T = x^{(2)}T$ for $x^{(1)} \neq x^{(2)}$, then $x^{(1)} - x^{(2)} \in (\mathcal{M}^{\perp})^{\perp} = \mathcal{M}$ from the definition of T and Lemma VI.5. This implies that

$$f(x^{(1)}) = f((x^{(1)} - x^{(2)}) + x^{(2)}) = f(x^{(2)}). \quad [\text{from}(30)]$$

Thus, g is well defined. Then, from (31) and Definition II.1, f is reducible. ■

REFERENCES

[1] R. Ahlswede, N. Cai, S. Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inf. Theory*, vol. IT-46, no. 4, pp. 1204–1216, Jul. 2000.

[2] N. Alon, "Combinatorial nullstellensatz," *Comb. Probab. Comput.*, no. 8, pp. 7–29, 1999.

[3] R. Appuswamy, M. Franceschetti, N. Karamchandani, and K. Zeger, "Network computing capacity for the reverse butterfly network," in *Proc. IEEE Int. Symp. Inf. Theory*, Jun./Jul. 2009, pp. 259–262.

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

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

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

[7] R. Dougherty, C. Freiling, and K. Zeger, "Linear network codes and systems of polynomial equations," *IEEE Trans. Inf. Theory*, vol. 54, no. 5, pp. 2303–2316, May 2008.

[8] A. Giridhar and P. R. Kumar, "Toward a theory of in-network computation in wireless sensor networks," *IEEE Commun. Mag.*, vol. 44, no. 4, pp. 98–107, Apr. 2006.

[9] G. H. Hardy and E. M. Wright, *An Introduction to the Theory of Numbers*. Oxford, U.K.: Oxford Univ. Press, 1979.

[10] N. J. A. Harvey, R. D. Kleinberg, and A. L. Rasala, Comparing network coding with multicommodity flow for the k-pairs communication problem Comput. Sci. Artif. Intell. Lab., Massachusetts Inst. Technol., Cambridge, USA, Tech. Rep. 964, 2004.

[11] N. J. A. Harvey, R. Kleinberg, and A. R. Lehman, "On the capacity of information networks," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2345–2364, Jun. 2006.

[12] R. Hill, *A First Course in Coding Theory*. Oxford, U.K.: Oxford Univ. Press, 1990.

[13] K. M. Hoffman and R. Kunze, *Linear Algebra*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1971.

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

[15] H. Kowshik and P. R. Kumar, "Zero-error function computation in sensor networks," in *Proc. IEEE Conf. Decision Control*, 2009, pp. 3787–3792.

[16] S. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. Inf. Theory*, vol. 49, no. 2, pp. 371–381, Feb. 2003.

[17] B. Nazer and M. Gastpar, "Computing over multiple-access channels," *IEEE Trans. Inf. Theory*, vol. 53, no. 10, pp. 3498–3516, 2007.

[18] G. Nebe, E. M. Rains, and N. J. A. Sloane, *Self-Dual Codes and Invariant Theory*. New York, NY, USA: Springer, 2006.

[19] J. Paek, B. Greenstein, O. Gnawali, K. Jang, A. Joki, M. Vieira, J. Hicks, D. Estrin, R. Govindan, and E. Kohler, "The tenet architecture for tiered sensor networks," *ACM Trans. Sensor Netw.*, 2009.

[20] B. K. Rai and B. K. Dey, Sum-Networks: System of Polynomial Equations, Unachievability of Coding Capacity, Reversibility, Insufficiency of Linear Network Coding 2009 [Online]. Available: http://arxiv.org/abs/0906.0695

[21] B. K. Rai, B. K. Dey, and S. Shenvi, "Some bounds on the capacity of communicating the sum of sources," in *Proc. IEEE Inf. Theory Workshop*, Jan. 2010, pp. 1–5.

[22] A. Ramamoorthy, "Communicating the sum of sources over a network," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2008, pp. 1646–1650.

[23] A. R. Lehman and E. Lehman, "Complexity classification of network information flow problems," in *Proc. 15th Annu. ACM-SIAM Symp. Discrete Algorithms*, 2003, pp. 142–150.

Rathinakumar Appuswamy, biography not available at the time of publication.

Massimo Franceschetti, biography not available at the time of publication.

Nikhil Karamchandani, biography not available at the time of publication.

Kenneth Zeger, biography not available at the time of publication.