Linear Coding for Network Computing

Rathinakumar Appuswamy, Massimo Franceschetti,
Nikhil Karamchandani, and Kenneth Zeger

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

I. 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 [1], [9], [10], whether linear codes are sufficient to achieve the capacity [6], [7], [11], [13], and cut-set upper bounds on the capacity and the tightness of such bounds [9], [10], [19].

Network computing, on the other hand, considers a more general problem in which each receiver node demands a target function of the source messages [3], [8], [12], [14], [16], [18]. Most problems in network coding are applicable to network computing as well. Network computing problems arise in applications such as sensor networks and vehicular networks.

In [3], 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 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. [15]).

In network coding, 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 non-multicast networks [6]. 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 [17]. However, it is known that for single-receiver networks, linear coding is sufficient for solvability when computing a scalar linear target function [4], [16]. Analogous to the coding capacity for network coding, the notion of computing capacity was defined for network computing in [8] 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, and semi-injective functions. 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.

A. Contributions

In Section III, 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 1. 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 III.4); 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 III.5), 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 III.7). 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 computing capacity (Theorem III.8). 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

This work was supported by the National Science Foundation

The authors are with the Department of Electrical and Computer Engineering, University of California, San Diego, La Jolla, CA 92093-0407. Emails: rathnam@ucsd.edu, massimo@ece.ucsd.edu, nikhil@ucsd.edu, zeger@ucsd.edu

This paper appeared at the the IEEE International Symposium on Information Theory (ISIT), in St. Petersburg, Russia, July-August 2011.

routing computing capacity. (Theorem III.10). Due to lack of space, many of the proofs have been omitted. The interested reader can find them in [4].

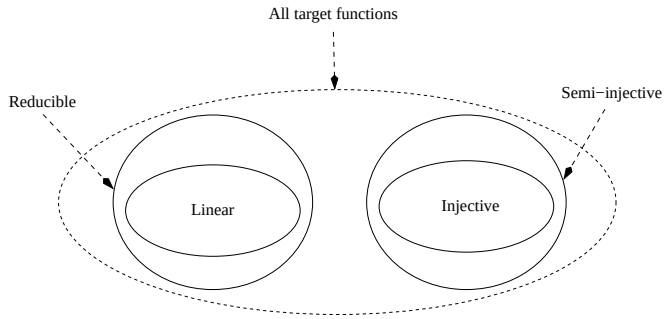

Fig. 1. Decomposition of the space of all target functions into various classes.

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 distinct *source nodes* and a single *receiver* $\rho \in \mathcal{V}$. 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 $\mathcal{E}_{in}(u)$ and $\mathcal{E}_{out}(u)$ 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 i-th component of x. 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

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).

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 $x \in \mathcal{A}^s$,

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

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 components

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

are called *messages*

Definition II.2. A (k, n) *network code in a network* $\mathcal{N}$ consists of *encoding functions* $h^{(e)}$, for every out-edge $e \in \mathcal{E}_{out}(v)$ of every node $v \in \mathcal{V} - \rho$, and a *decoding function* ψ.

Furthermore, given a (k, n) network code, every edge $e \in \mathcal{E}$ carries a vector z_e of at most n alphabet symbols[1], 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 II.3. Suppose in a network $\mathcal{N}$, the in-edges of the receiver are $e_1, e_2, \ldots, e_{|\mathcal{E}_{in}(\rho)|}$. 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

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

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 [5]. We use an analogous definition for the computing capacity.

Definition II.4. The *computing capacity* of a network $\mathcal{N}$ with respect to a target function f is

$$\mathcal{C}_{\text{cod}}(\mathcal{N}, f) \ = \ \sup\Big\{\frac{k}{n} \ : \ \exists\ (k, n) \text{ network code that computes } f \text{ in } \mathcal{N}\Big\}. \tag{3}$$

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.5. 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 $\mathcal{A}$.

The *linear computing capacity* $\mathcal{C}_{lin}(\mathcal{N}, f)$ and the *routing computing capacity* $\mathcal{C}_{rout}(\mathcal{N}, f)$ are defined similarly

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

by restricting the encoding functions to be linear functions (over $\mathcal{A}$) and routing, respectively. 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.

III. Linear network codes for computing target functions

It turns out that if intermediate network nodes are restricted to use only 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 non-injective. Hence, if the target function is not reducible, such computing capacity gain must be obtained from nonlinear coding. Section III-B 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.

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

Theorem III.4 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 [19]. For network computing, the next theorem shows that with non-reducible target functions there is no advantage to using linear coding over routing.[2]

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

Theorem III.4. *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*

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

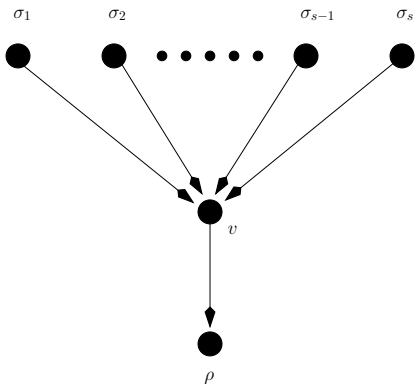

Fig. 2. Network $\mathcal{N}_s$ has sources $\sigma_1, \sigma_2, \ldots, \sigma_s$, each connected to the relay v by an edge and v is connected to the receiver by an edge.

Theorem III.4 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 III.5, 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 II 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 III.5. *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 [19, 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. ∎

Theorem III.4 showed that there cannot be linear computing gain for networks whose target functions are not reducible, and Theorem III.5 showed that the same is true for target functions that are injective. However, Theorem III.7 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 III.7 we only consider the case when there are two or more sources. We need the following lemma first.

Lemma III.6. *The computing capacity of the network $\mathcal{N}_s$ shown in Figure 2, with respect to a target function $f : \mathcal{A}^s \longrightarrow$*

$\mathcal{B}$, *satisfies*

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

Theorem 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*

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

Proof: If $\mathcal{N}$ is the network $\mathcal{N}_s$ shown in Figure 2 with alphabet $\mathcal{A}$, then

$$\begin{aligned}
&\mathcal{C}_{\text{lin}}(\mathcal{N}, f) = 1/s && \text{[from Theorem III.4]}\\
&< \min\left\{1,\ \frac{1}{\log_{|\mathcal{A}|}|f\left(\mathcal{A}^s\right)|}\right\} && \text{[from } s \geq 2,\ |f\left(\mathcal{A}^s\right)| < |\mathcal{A}|^s\text{]}\\
&\leq \mathcal{C}_{\text{cod}}(\mathcal{N}, f)\,. && \text{[from Lemma III.6]}
\end{aligned}$$

■

B. Reducible target functions

In Theorem III.8, we prove a converse to Theorem III.4 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 III.10 shows that, even if the target function is reducible, linear codes may not achieve the full (nonlinear) computing capacity of a network.

Theorem III.8. *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*

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

For target functions that are not reducible, any improvement on achievable rate of computing using coding must be provided by nonlinear codes (by Theorem III.4). 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), while for certain other reducible target functions, nonlinear codes might provide a strictly larger achievable computing rate compared to linear codes.

Remark III.9. 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.

Theorem III.7 shows that for every non-injective, non-reducible target function, some network has a nonlinear computing gain over linear coding, and Theorem III.8 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 III.10. *There exists a network $\mathcal{N}$ and a reducible target function f such that:*

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

REFERENCES

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

[2] R. Appuswamy, M. Franceschetti, N. Karamchandani, and K. Zeger, "Network computing capacity for the reverse butterfly network," *IEEE International Symposium on Information Theory (ISIT)*, pp. 259–262, 2009.

[3] R. Appuswamy, M. Franceschetti, N. Karamchandani, and K. Zeger, "Network coding for computing: cut-set bounds", *IEEE Transactions on Information Theory*, vol. 57, no. 2, pp. 1015–1030, February 2011.

[4] R. Appuswamy, M. Franceschetti, N. Karamchandani, and K. Zeger, "Linear codes, target function classes, and network computing capacity," available at *http://arxiv.org/abs/1101.0085*, 2011.

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

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

[7] R. Dougherty, C. Freiling, and K. Zeger, "Linear network codes and systems of polynomial equations", *IEEE Transactions on Information 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 Communications Magazine*, vol. 44, no. 4, pp. 98–107, 2006.

[9] N. J. A. Harvey, R. Kleinberg, and A. Rasala Lehman, "On the capacity of information networks," *IEEE Transactions on Information Theory & IEEE/ACM Transactions on Networking (joint issue)*, vol. 52, no. 6, pp. 2345–2364, 2006.

[10] N. J. A. Harvey, R. D. Kleinberg, and A. Rasala Lehman, "Comparing network coding with multicommodity flow for the k-pairs communication problem," *MIT LCS, Tech. Rep. 964*, 2004.

[11] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Transactions on Networking*, vol. 11, no. 5, pp. 78—795, October 2003.

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

[13] S. R. Li, R. W. Yeung, and N. Cai, "Linear network coding", *IEEE Transactions on Information Theory*, vol. 49, pp. 371–381, 2003.

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

[15] 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 Transactions on Sensor Networks (TOSN)*, 2009.

[16] B. K. Rai, B. K. Dey, S. Shenvi, "Some bounds on the capacity of communicating the sum of sources," *IEEE International Symposium on Information Theory (ISIT)*, Cairo, Egypt, 2010.

[17] B. K. Rai, and 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*, 2009.

[18] A. Ramamoorthy, "Communicating the sum of sources over a network," *IEEE International Symposium on Information Theory (ISIT)*, Toronto, Canada, 2008.

[19] A. Rasala Lehman and E. Lehman, "Complexity classification of network information flow problems," *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 142–150, 2003.