Function computation via subspace coding[☆]

Nikhil Karamchandani[b,c], Lorenzo Keller[a,*], Christina Fragouli[a], Massimo Franceschetti[b]

[a] *École Polytechnique Fédérale de Lausanne (EPFL), CH-1015, Lausanne, Switzerland*
[b] *University of California, San Diego, La Jolla, CA 92093*
[c] *University of California Los Angeles, Los Angeles, CA 90095-1594*

Abstract

This paper considers function computation in a network where intermediate nodes perform randomized network coding, through appropriate choice of the subspace codebooks at the source nodes. Unlike traditional network coding for computing functions, that requires intermediate nodes to be aware of the function to be computed, our designs are transparent to the intermediate node operations.

Keywords: Function computation; Network coding; Subspace coding; Wireless sensor network

1. Introduction

In sensor networks, the need for energy efficiency has stimulated research efforts towards in-network aggregation and function computation. Information-theoretic studies have focused on the design of *optimal* schemes for different target functions and network classes, see for example [1], [2], [3]. On the other hand, recent work [4], [5] has pointed out the need to have *simple* coding schemes, since "systems are hard to develop and debug". They advocate a solution where most nodes in the network perform the same operations regardless of the function to be computed, and the onus of guaranteeing successful computation is on a few special nodes that are allowed to vary their operation.

[☆] This work was partially funded by ERC Project NOWIRE (ERC-2009-StG-240317) and NSF CNS-0916778

[*] Corresponding author

Email addresses: nikhil@ee.ucla.edu (Nikhil Karamchandani), lorenzo.keller@epfl.ch (Lorenzo Keller), christina.fragouli@epfl.ch (Christina Fragouli), massimo@ece.ucsd.edu (Massimo Franceschetti)

Motivated by the above considerations, we consider the problem of computing functions in a network where multiple sources are connected to a single sink via relays. The sources may have several different possible codebooks, and can select which one to employ depending on the function to be computed. Given a certain target function, each source transmits a codeword corresponding to its observed message. The relay nodes, however, perform the same linear operations, for example randomized network coding (which is a practical and efficient way of transmitting data in a network [6]) irrespective of the target function, i.e., the vectors inserted by the sources are randomly combined and forwarded towards the sink, using linear coefficients that are unknown to both the sources and the sink. The sink then proceeds to evaluate the target function of the source messages.

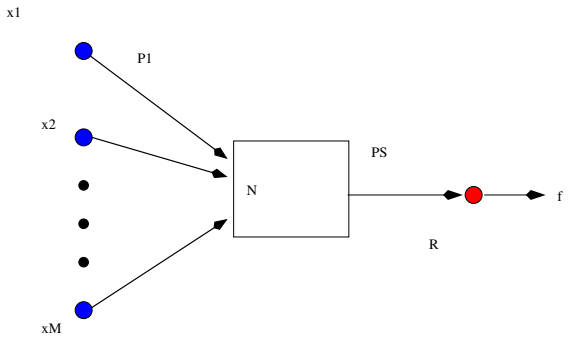
Following [7], [8], [9], we use subspace coding for computing functions in our network model. Given a target function, we assume that each source uses a codebook consisting of subspaces. Each source message is mapped to a subspace in the codebook. When a source generates a message, it injects the basis vectors of the corresponding subspace into the network. The network operation is abstracted by assuming that the sink collects enough linear combinations of these vectors to learn the joint span of the injected subspaces. Given this information, the sink then attempts to compute the target function of the source messages. Our objective is to design codebooks which minimize the number of symbols each source needs to transmit, while guaranteeing successful function computation by the sink.

Thus, we envision a network architecture where intermediate network nodes always perform the same operations for information transfer, which leads to a simple implementation. At the same time, the sink has the flexibility to utilize the network to learn different functions of the source data by informing the source nodes to employ the corresponding codebooks. In this paper we are interested in non-coherent communication; this allows for a low complexity asynchronous operation of the network [7]. Another approach would be to consider the coherent case as done in [10].

We note that a scheme which optimizes the intermediate node operations according to the function to be computed might need fewer transmissions. However, it would be more complex to implement, would require topology knowledge, and might be sensitive to the employed communication protocol. In contrast, our approach is transparent both to the topology and the employed communication protocol: the only requirement we impose is that we gather sufficient linearly independent combinations. As a result, our protocol would be very well suited to dynamically changing topologies, and could be applied without change on top of very different communication protocols.

The paper is organized as follows. Section 2 presents the problem formulation. In Section 3, we present various lower bounds on the number of symbols each source needs to transmit to evaluate an arbitrary function. In Section 4, we discuss various example target functions. In particular, we provide lower bounds as well as near-optimal coding schemes for the *identity*, T*-threshold*, *maximum* and K*-largest values* functions. Finally, in Section 3, we present a constructive scheme to evaluate arbitrary functions.

2. Problem Formulation and Notation



We consider a set of N sources $\sigma_1, \sigma_2, \ldots, \sigma_N$ connected to a sink ρ via a network $\mathcal{N}$. Each source σ_i is either inactive or observes a message $x_i \in \mathcal{A}$, where $\mathcal{A}$ is a finite alphabet. For ease of notation, when a source σ_i is inactive we will set $x_i = \phi$. The sink needs to compute a *target function* f of the source messages, where f is of the form

$$f \ : \ (\mathcal{A} \cup \{\phi\})^N \longrightarrow \mathcal{B}.$$

Some example target functions are defined below.

Definition 2.1.

- The *identity* target function has $\mathcal{B} = (\mathcal{A} \cup \{\phi\})^N$ and is defined by

$$f\left(x_1, \ldots, x_N\right) = \left(x_1, \ldots, x_N\right).$$

- For $m \geq 1$, the *arithmetic sum* target function has $\mathcal{A} = \{1, \ldots, m\}$, $\mathcal{B} = \{0, 1, \ldots, mN\}$, and is defined by

$$f\left(x_1, \ldots, x_N\right) = x_1 + x_2 + \cdots + x_N$$

where '+' denotes ordinary integer summation. For any $a \in \mathcal{A} \cup \{\phi\}$, we set $a + \phi = a$.

- Let $\mathcal{A}$ be an ordered set. The *maximum* target function has $\mathcal{B} = \mathcal{A}$ and is defined by

$$f(x_1, \ldots, x_N) = \max\{x_1, \ldots, x_N\}.$$

  For any $a \in \mathcal{A} \cup \{\phi\}$, we set $\max\{a, \phi\} = a$.

- The *parity* target function has $\mathcal{A} = \mathcal{B} = \{0, 1\}$, and is defined by

$$f(x_1, \ldots, x_N) = x_1 \oplus x_2 \oplus \ldots \oplus x_N$$

  where $\oplus$ denotes mod-2 addition. Again, for any $a \in \mathcal{A} \cup \{\phi\}$ we set $a \oplus \phi = a$.

- The *majority* target function has $\mathcal{A} = \mathcal{B} = \{0, 1\}$, and is defined by

$$f(x_1, \ldots, x_N) = \begin{cases} 1 & \text{if } |\{i : x_i = 1\}| > |\{i : x_i = 0\}| \\ 0 & \text{otherwise.} \end{cases}$$

We consider operation using subspace coding. We denote a subspace by π and the union of two subspaces π_1, π_2 is defined as $\pi_1 + \pi_2 = \{\mathbf{x}+\mathbf{y} : \mathbf{x} \in \pi_1, \mathbf{y} \in \pi_2\}$. We write $\pi_1 + \pi_2 + ...\pi_m$ as $\sum_{i=1}^{m} \pi_i$. The network operates as follows.

- At each source, every alphabet symbol is mapped to a subspace, which serves as the corresponding codeword. Thus, each source σ_i has an associated codebook $\mathcal{C}_i = \left\{\pi_i^j\right\}_{j \in \mathcal{A}}$ where π_i^j is a d-dimensional subspace[1] of an l-dimensional vector space $\mathbb{F}_q^l$ where $d, l \geq 1$ are design parameters. When the source σ_i is active and observes a message $x_i \in \mathcal{A}$, it injects into the network $\mathcal{N}$ a set of d vectors from $\mathbb{F}_q^l$ which span the subspace $\pi_i^{x_i}$. When the source is σ_i inactive, it does not make any transmissions and hence we set $\pi_i^{\phi} = \{0\}$.

- The sink ρ receives from the network $\mathcal{N}$ a set of vectors from $\mathbb{F}_q^l$ which span the union of the input subspaces[2] i.e., ρ observes $\sum_{i=1}^{N} \pi_i^{x_i}$.

[1] Although restricting our code design to subspaces of equal dimension may not always be optimal, it significantly simplies the design, and is a standard approach in the literature [7, 11].

[2] In practice, networks operate in rounds. The duration of a round can be chosen large enough

- The sink uses the received information to compute the value of $f(x_1, x_2, \ldots, x_N)$.

A (d, l) *feasible code for computing* f is a collection of codebooks $\{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_N\}$ such that each π_i^j in the codebooks is a d-dimensional subspace of $\mathbb{F}_q^l$ and the sink can compute the value of $f(x_1, x_2, \ldots, x_N)$ for any choice of input messages $x_1, x_2, \ldots, x_N$ where each $x_i \in \mathcal{A} \cup \{\phi\}$, i.e. for all $x_1, x_2, \ldots, x_N$ and $x_1', x_2', \ldots, x_N'$, if

$$f(x_1, x_2, \ldots, x_N) \neq f(x_1', x_2', \ldots, x_N'),$$

then $\sum_{i=1}^N \pi_i^{x_i} \neq \sum_{i=1}^N \pi_i^{x_i'}$.

For a (d, l) feasible code for computing f, each source transmits at most $d \cdot l$ symbols from $\mathbb{F}_q$, and we thus consider the associated cost[3] to be $d \cdot l$. Our code design seeks to achieve

$$\mathcal{E}_{\min}(f) = \inf \{d \cdot l : \exists \text{ a } (d, l) \text{ feasible code for computing } f\}.$$

We begin by showing that for the purpose of minimizing the cost $d \cdot l$, it suffices to consider codes which use one-dimensional subspaces.

Theorem 2.2. *Given any (d, l) feasible code for computing a target function f, there also exists a $(1, d \cdot l)$ feasible code for computing f.*

Proof. Partition $(\mathcal{A} \cup \{\phi\})^N$ into $P_1, P_2, \ldots, P_{|\mathcal{B}|}$ so that each P_i consists of all the input vectors which result in the same function value. Then a necessary and sufficient condition for successful computation is that the sink should receive different union subspaces for any two input vectors $\mathbf{x}$ and $\mathbf{y}$ if they belong to distinct partitions.

Let $\{\pi_i^j \subseteq \mathbb{F}_q^l : i \in \{1, \ldots, N\}, j \in \mathcal{A}\}$ denote the collection of d-dimensional subspaces associated with the given (d, l) feasible code for computing f. The above necessary condition implies that these subspaces satisfy a collection of inequalities, each of the form

$$\sum_{i=1}^N \pi_i^{a_i} \neq \sum_{i=1}^N \pi_i^{b_i} \tag{1}$$

to ensure that the sink receives enough linear independent combinations to span the union of the input subspaces.

[3] In this work, the field size q is considered to be fixed and hence not included in the cost.

where each $a_i, b_i \in \mathcal{A} \cup \{\phi\}$. Now corresponding to each d-dimensional subspace π_i^j in the above code, construct a one-dimensional subspace $\widehat{\pi}_i^j \subseteq \mathbb{F}_q^{d \cdot l}$ by concatenating the d basis vectors of π_i^j into a single vector. It can be verified that the collection of one-dimensional subspaces $\{\widehat{\pi}_i^j : i \in \{1, \ldots, N\}, j \in \mathcal{A}\}$ constructed this way also satisfy all the inequalities that the original code satisfied, i.e., if (1) holds for some $\{a_i\}, \{b_i\}$, then

$$\sum_{i=1}^{N} \widehat{\pi}_i^{a_i} \neq \sum_{i=1}^{N} \widehat{\pi}_i^{b_i}. \tag{2}$$

Since (1) holds, there exists at least one basis vector, say $v_1 \in \pi_1^{a_1}$, which is not in $\sum_{i=1}^{N} \pi_i^{b_i}$. This immediately implies that $\widehat{\pi}_1^{a_1}$ cannot be an element of $\sum_{i=1}^{N} \widehat{\pi}_i^{b_i}$, which proves (2).

Thus the collection of one-dimensional subspaces $\{\widehat{\pi}_i^j : i \in \{1, \ldots, N\}, j \in \mathcal{A}\}$ ensures that for any two input vectors in distinct partitions, the sink receives different union subspaces. Since this is sufficient for function computation, we have shown that we can construct a $(1, d \cdot l)$ feasible code from any (d, l) feasible code. ■

In the sequel, we will only consider codes which use one-dimensional subspaces. We will denote the dimension of any subspace π by $\dim(\pi)$. Also, for any vector $\mathbf{x}$, the j-th component will be denoted by $(\mathbf{x})_j$. Consider a set of indices $I = \left(i_1, i_2, \ldots, i_{|I|}\right) \subseteq \{1, \ldots, N\}$. For any $\mathbf{a} = \left(a_1, a_2, \ldots, a_{|I|}\right) \in (\mathcal{A} \cup \{\phi\})^{|I|}$ and any vector $\mathbf{x} \in (\mathcal{A} \cup \{\phi\})^N$, let $\mathbf{x}(I, \mathbf{a}) = (x_1, x_2, \ldots, x_N)$ denote a vector which is obtained from $\mathbf{x}$ by substituting the components corresponding to the index set I with values from the vector $\mathbf{a}$ and retaining all the other components. That is, for each $j \in \{1, \ldots, |I|\}$, $(\mathbf{x}(I, \mathbf{a}))_{i_j} = (\mathbf{a})_j$ and for each $k \notin I$, $(\mathbf{x}(I, \mathbf{a}))_k = (\mathbf{x})_k$. We conclude this section with a lemma that is often used in the subsequent sections.

Lemma 2.3. *If there exist one-dimensional subspaces $\pi_1, \pi_2, \ldots, \pi_K \subseteq \mathbb{F}_q^l$ and a subspace $\pi^* \subseteq \mathbb{F}_q^l$ such that*

$$\pi_i \not\subseteq \pi^* + \sum_{j<i} \pi_j \quad \forall\, i \in \{1, \ldots, K\} \tag{3}$$

then $l \geq K$.

Proof. (3) implies that the basis vectors for the K one-dimensional subspaces are linearly independent. The result then follows. ■

3. Lower bounds

The number of d-dimensional subspaces of $\mathbb{F}_q^l$ for any $d \leq l, l \geq 1$ is given by the gaussian binomial coefficient

$$\begin{bmatrix} l \\ d \end{bmatrix}_q = \frac{(q^l-1)(q^{l-1}-1)\cdots(q^{l-d+1}-1)}{(q^d-1)(q^{d-1}-1)\cdots(q-1)}. \tag{4}$$

We have the following upper bound on the number of d-dimensional subspaces..

Lemma 3.1. *The number of d-dimensional subspaces of $\mathbb{F}_q^l$ is at most $4q^{d(l-d)}$ [7, Lemma 4].*

Recall that $\mathbf{x}(I, \mathbf{a})$ denotes a vector which is obtained from $\mathbf{x}$ by substituting the components corresponding to the index set I with values from the vector $\mathbf{a}$ and retaining all the other components. Consider a target function with the following property.

Function property $\mathbf{P}$: There exists $k \in \{1, \ldots, N\}$ and $\mathbf{x} \in (\mathcal{A} \cup \{\phi\})^N$ such that for any distinct $a, b \in \mathcal{A}$,

$$f(\mathbf{x}(\{k\}, a)) \neq f(\mathbf{x}(\{k\}, b)).$$

Examples : The identity function and arithmetic sum function satisfy property $\mathbf{P}$.

We have the following simple lower bound for functions which satisfy property $\mathbf{P}$.

Lemma 3.2. *For any target function f which satisfies property $\mathbf{P}$,*

$$\mathcal{E}_{min}(f) \geq \log_q \left(|\mathcal{A}| \, (q-1) + 1 \right).$$

Proof. From the definition of property $\mathbf{P}$, there exists k such that source σ_k must assign a distinct one-dimensional subspace to each $a \in \mathcal{A}$. From (4), we have

$$\begin{aligned} &\frac{q^l - 1}{q-1} \geq |\mathcal{A}| \\ \implies &l \geq \log_q \left(|\mathcal{A}| \, (q-1) + 1 \right). \end{aligned}$$

■

Consider a target function with the following property.

Function property $\mathbf{Q}(k)$: There exists $\mathbf{x} \in (\mathcal{A} \cup \{\phi\})^N$ and a collection of k distinct indices $i_1, i_2, \ldots, i_k$ such that for every $j \in \{1, 2, \ldots, k\}$, we have $(\mathbf{x})_j \neq \phi$ and

$$f\left(\mathbf{x}(\{i_1, \ldots, i_{j-1}\}, \{\phi, \ldots, \phi\})\right) \neq f\left(\mathbf{x}(\{i_1, \ldots, i_{j-1}, i_j\}, \{\phi, \ldots, \phi\})\right). \quad (5)$$

Example 3.3.

- *The identity function satisfies property* $\mathbf{Q}(N)$ *by choosing each* $(\mathbf{x})_j$ *equal to any element of the alphabet* $\mathcal{A}$*.*
- *The arithmetic sum function satisfies property* $\mathbf{Q}(N)$ *by choosing each* $(\mathbf{x})_j$ *equal to some non-zero element of the alphabet* $\mathcal{A}$*.*
- *The parity function satisfies property* $\mathbf{Q}(N)$ *by choosing each* $(\mathbf{x})_j$ *equal to* 1*.*
- *The majority function satisfies property* $\mathbf{Q}(N)$ *by choosing* $(\mathbf{x})_j$ *equal to* 1 *if* j *is even and* 0 *otherwise when* N *is even and vice-versa when* N *is odd.*

Lemma 3.4. *For any target function* f *which satisfies property* $\mathbf{Q}(k)$*,*

$$\mathcal{E}_{min}(f) \geq k.$$

Proof. Let

$$\Pi^c = \sum_{t \notin \{i_1, i_2, \ldots, i_k\}} \pi_t^{(\mathbf{x})_t}.$$

From (5), any feasible code for computing f should satisfy for every $j \in \{1, 2, \ldots, k\}$

$$\pi_{i_j}^{(\mathbf{x})_{i_j}} + \sum_{m=j+1}^{k} \pi_{i_m}^{(\mathbf{x})_{i_m}} + \Pi^c \neq \sum_{m=j+1}^{k} \pi_{i_m}^{(\mathbf{x})_{i_m}} + \Pi^c$$

$$\implies \pi_{i_j}^{(\mathbf{x})_{i_j}} \not\subseteq \sum_{m=j+1}^{k} \pi_{i_m}^{(\mathbf{x})_{i_m}} + \Pi^c.$$

Then using Lemma 2.3 for the collection of k one-dimensional subspaces $\pi_{i_1}^{(\mathbf{x})_{i_1}}, \ldots, \pi_{i_k}^{(\mathbf{x})_{i_k}}$ and the subspace Π^c, the result follows. ∎

We borrow the following definition from [12].

Definition 3.5. For any target function $f : (\mathcal{A} \cup \{\phi\})^N \longrightarrow \mathcal{B}$, any index set $I \subseteq \{1, 2, \ldots, N\}$, and any $\mathbf{a}, \mathbf{b} \in (\mathcal{A} \cup \{\phi\})^{|I|}$, we write $\mathbf{a} \equiv \mathbf{b}$ if for every $\mathbf{x} \in (\mathcal{A} \cup \{\phi\})^N$, we have $f(\mathbf{x}(I, \mathbf{a})) = f(\mathbf{x}(I, \mathbf{b}))$.

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

Definition 3.6. For every f and I, let $R_{I,f}$ denote the total number of equivalence classes induced by $\equiv$ and let

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

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

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

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

The value of $R_{I,f}$ is independent of the choice of $\Phi_{I,f}$.

Example 3.7.

- *Let f be the identity target function. Then for every $\mathbf{a}, \mathbf{b} \in (\mathcal{A} \cup \{\phi\})^{|I|}$ we have $\mathbf{a} \equiv \mathbf{b}$ if and only if $\mathbf{a} = \mathbf{b}$. The number of distinct equivalence classes is*
$$R_{I,f} = (|\mathcal{A}| + 1)^{|I|}.$$
- *Let $\mathcal{A} = \{1, \ldots, m\}$ for some $m \geq 1$ and let f be the arithmetic sum target function. For $\mathbf{a}, \mathbf{b} \in (\mathcal{A} \cup \{\phi\})^{|I|}$, we have $\mathbf{a} \equiv \mathbf{b}$ if and only if*
$$\sum_{i=1}^{|I|} (\mathbf{a})_i = \sum_{i=1}^{|I|} (\mathbf{b})_i$$
and the number of such possible sums is
$$R_{I,f} = m\,|I| + 1.$$

[4] Witsenhausen [13] represented this equivalence relation in terms of the independent sets of a characteristic graph and his representation has been used in various problems related to function computation [14, 15]. Although $\equiv$ is defined with respect to a particular index set I and a function f, we do not make this dependence explicit – the values of I and f will be clear from the context.

Lemma 3.8. *For any target function f,*

$$\mathcal{E}_{min}(f) \geq \max_{I} \max \left\{ \frac{\sqrt{\log_q (R_{I,f})}}{3}, \frac{\log_q(R_{I,f})}{3\,|I|} \right\}.$$

Proof. Consider any $I = \{i_1, i_2, \ldots, i_{|I|}\}$. For any $\mathbf{a}, \mathbf{b} \in (\mathcal{A} \cup \{\phi\})^{|I|}$ such that $\mathbf{a} \not\equiv \mathbf{b}$, any feasible code should satisfy

$$\sum_{j \in \{1,\ldots,|I|\}} \pi_{i_j}^{(\mathbf{a})_j} \neq \sum_{j \in \{1,\ldots,|I|\}} \pi_{i_j}^{(\mathbf{b})_j}. \tag{6}$$

Note that for any I and $\mathbf{a} \in (\mathcal{A} \cup \{\phi\})^{|I|}$, $\dim \left(\sum_{j \in \{1,\ldots,|I|\}} \pi_{i_j}^{(\mathbf{a})_j} \right) \leq |I|$ since it is composed of the union of at most $|I|$ one-dimensional subspaces. Then from Definition 3.5 and (6), there exist at least $R_{I,f}$ distinct subspaces, each with dimension at most $|I|$. From Lemma 3.1, we have

$$4 \cdot \sum_{j=1}^{\min\{l,|I|\}} q^{j(l-j)} \quad \geq \quad R_{I,f}. \tag{7}$$

This implies that

$$\begin{aligned} & 4 \cdot \sum_{j=1}^{l} q^{j(l-j)} \geq R_{I,f} \\ \Longrightarrow & 4l \cdot q^{\left(\frac{l}{2}\right)^2} \geq R_{I,f} \\ \Longrightarrow & \log_q(4l) + \left(\frac{l}{2}\right)^2 \geq \log_q(R_{I,f}). \end{aligned}$$

Since $\log_q(4l) \leq 2l^2$, we have

$$\begin{aligned} & 3l^2 \geq \log_q(R_{I,f}) \\ \Longrightarrow & l \geq \frac{\sqrt{\log_q(R_{I,f})}}{3}. \end{aligned}$$

From (7), we also have

$$\begin{aligned} & 4 \cdot \sum_{j=1}^{|I|} q^{j(l-j)} \geq R_{I,f} \\ \Longrightarrow & 4\,|I| \cdot q^{\hat{d}(l-\hat{d})} \geq R_{I,f} \text{ with } \hat{d} = \underset{j \in \{1,|I|\}}{\operatorname{argmax}} q^{j(l-j)} \\ \Longrightarrow & \log_q(4\,|I|) + \hat{d}(l-\hat{d}) + \geq \log_q(R_{I,f}). \end{aligned}$$

Since $\log_q(4\,|I|) \leq 2\,|I|$ and $\hat{d} \leq |I|$, we have

$$\begin{aligned} & 2\,|I|\,l + |I|\,l \geq \log_q(R_{I,f}) \\ \Longrightarrow & l \geq \frac{\log_q(R_{I,f})}{3\,|I|}. \end{aligned}$$

■

Example 3.9.

- *For the arithmetic sum target function f, we get*

$$\mathcal{E}_{min}(f) \geq \frac{\sqrt{\log_q(N\,|\mathcal{A}|+1)}}{3}.$$

Comment : Note that when $|\mathcal{A}| \gg N$, the bounds in the above examples are better than the ones presented earlier in the section.

4. Bounds for specific functions

Any target function can be computed by first reconstructing all the source messages at the sink (i.e., computing the identity function $f(x_1, x_2, \ldots, x_N) = (x_1, x_2, \ldots, x_N)$ with each $x_i \in \mathcal{A} \cup \{\phi\}$) and then deriving the function value. Hence, the following lemma provides an upper bound on the cost for computing any function f.

Lemma 4.1. *There exists a $(1, l)$ feasible code for computing the identity function such that*

$$l = N + \lceil \log_q |\mathcal{A}| \rceil.$$

Proof. It is easy to see that this can be achieved simply by using coding vectors of length N, where each source σ_i when active uses the basis vector $\mathbf{e_i}$ as its coding vector and appends this to the information packet that consists of $\lceil \log_q |\mathcal{A}| \rceil$ symbols. ■

In the previous section, we provided lower bounds on $\mathcal{E}_{\min}(f)$ for arbitrary functions. Functions for which the lower bound is of the same order as $N + \lceil \log_q |\mathcal{A}| \rceil$ are hard to compute in the sense that it is almost optimal (up to a constant factor) to first recover all the source messages and then compute the function. For example, when $N \geq \log_q |\mathcal{A}|$, this is true for the arithmetic sum function, the parity function, and the majority function. Next, we discuss some example target functions.

4.1. T-threshold Function

Let $\mathcal{A} = \{1\}$. The T-threshold function is defined as[5]

$$f(x_1, x_2, \ldots, x_N) = \begin{cases} 1 & \text{if } x_1 + x_2 + \ldots + x_N \geq T \\ 0 & \text{otherwise.} \end{cases}$$

Lemma 4.2. *There exists a $(1, l)$ feasible code for computing the T-threshold function with $T < (1 - 1/q)N$, such that*

$$l \leq N H_q \left(\frac{T}{N} \right)$$

where H_q is the q-ary entropy function defined as

$$H_q(x) = x \log_q \left(\frac{q-1}{x} \right) + (1-x) \log_q \left(\frac{1}{1-x} \right) \ \forall\, x \in (0, 1).$$

Proof. Consider the scheme in Figure 1. The scheme uses a $l \times N$ parity check matrix of a q-code with minimum distance $d_{min} = T + 1$. From the Gilbert-Varshamov bound [16], there exists such a matrix with

$$l \leq N H_q \left(\frac{T}{N} \right).$$

[5]The function computes whether the number of active sources is at least T or not.

- Let $\mathbf{H}$ be the $l \times N$ parity check matrix of a q-ary code with minimum distance $d_{min} = T + 1$.
- Source σ_i uses $C_i = \{\mathbf{h_i}\}$, where $\mathbf{h_i}$ is a column of $\mathbf{H}$.
- If the dimension of the subspace that the sink receives is less than T, it outputs 0. Otherwise, it outputs 1.

Figure 1: A $(1, l)$ code for the T-threshold function

■

Comment : For a constant T, $O\left(NH_q\left(\frac{T}{N}\right)\right) = O\left(T \log_q N\right)$. Thus, while computing the identity function requires the cost to grow linearly with the number of sources N, the T-threshold function requires only logarithmic growth. We have the following matching lower bound.

Lemma 4.3. *For the T-threshold function f with $T < N/2$,*

$$\mathcal{E}_{min}(f) \geq \frac{N}{2} H_q\left(\frac{T}{2N}\right).$$

Proof. Consider two possible input vectors $(x_1, x_2, \ldots, x_N)$ and $(y_1, y_2, \ldots, y_N)$ such that

$$x_i = 1 \; \forall \; i \in \{1, 2, \ldots, T\} \text{ and } x_i = \phi \text{ otherwise}$$
$$y_i = 1 \; \forall \; i \in \{2, 3, \ldots, T\} \text{ and } y_i = \phi \text{ otherwise .}$$

Note that

$$1 = f(x_1, x_2, \ldots, x_N) \neq f(y_1, y_2, \ldots, y_N) = 0$$

and hence it is necessary for any feasible code for computing f that

$$\pi_1^1 + \sum_{i=2}^{T} \pi_i^1 \neq \sum_{i=2}^{T} \pi_i^1 \Longrightarrow \pi_1^1 \not\subseteq \sum_{i=2}^{T} \pi_i^1.$$

The same argument can be extended to get the following necessary condition. For

any subset $(i_1, i_2, \ldots, i_T)$ of $\{1, 2, \ldots, N\}$,

$$\pi^1_{i_j} \not\subseteq \sum_{k \neq j} \pi^1_{i_k} \text{ for every } j \in \{1, 2, \ldots, T\}.$$

Denote a basis vector for π^1_i by $\mathbf{v_i}$. From the necessary condition on the subspaces $\pi^1_1, \pi^1_2, \ldots, \pi^1_N$, any collection of T vectors from $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_N}$ are linearly independent. The $l \times N$ matrix with the vectors $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_N}$ as columns corresponds to the parity check matrix of a q-ary linear code of length N and minimum distance at least $T + 1$. Using the bounds in [16], for $T < N/2$ we have

$$l \;\geq\; NH_q\left(\frac{T}{2N}\right) - \frac{1}{2}\log_q\left(4T\left(1 - \frac{T}{2N}\right)\right).$$

The result then follows since

$$\frac{1}{2}\log_q\left(4T\left(1 - \frac{T}{2N}\right)\right) \leq \frac{N}{2}H_q\left(\frac{T}{2N}\right). \tag{8}$$

For $N \leq 11$, (8) can be verified numerically. Let $N \geq 12$. Then (8) holds if we show that for every $1 \leq T < N/2$,

$$N \cdot \frac{T}{2N}\ln\left(\frac{2N}{T}\right) \geq \ln(4T) \text{ or equivalently,}$$
$$T\ln\left(\frac{2N}{T}\right) - 2\ln(4T) \geq 0. \tag{9}$$

For $T = 1$, (9) holds since $N \geq 8$. Differentiating the left-hand side of (9) with respect to T, we get

$$\ln(2N) - \ln(T) - 1 - \frac{2}{T}$$

which is greater than zero since $N \geq 12$ and $T \leq N/2$. Thus, (9) is true for every $1 \leq T < N/2$ and thus (8) holds. ■

4.2. Maximum Function

Lemma 4.4. *There exists a $(1, l)$ feasible code for computing the maximum function such that*

$$l \leq \min\left\{|\mathcal{A}|, N + \lceil\log_q|\mathcal{A}|\rceil\right\}.$$

Proof. Consider the following two schemes for computing the maximum function.
• *A* $(1, |\mathcal{A}|)$ *scheme*: Let $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v}_{|\mathcal{A}|}$ be linearly independent vectors of length $|\mathcal{A}|$ each. For every source σ_i, let $\mathcal{C}_i = \left(\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v}_{|\mathcal{A}|}\right)$. This scheme has $l = |\mathcal{A}|$.
• *A* $(1, N + \lceil \log_q |\mathcal{A}| \rceil)$ *scheme*: We can compute the identity function with $l = N + \lceil \log_q |\mathcal{A}| \rceil$ and hence can compute the maximum function also. This scheme is useful if $\mathcal{A} \geq N$. ■

Comment : Thus when $|\mathcal{A}| \ll N$, the first scheme is much more efficient than reconstructing all the source messages.

Lemma 4.5. *For the maximum target function* f*,*

$$\mathcal{E}_{min}(f) \geq \min\{|\mathcal{A}|, N\}.$$

Proof. Let $\mathcal{A} = \left(a_1, a_2, \ldots, a_{|\mathcal{A}|}\right)$ be an ordered set (in decreasing order) and let $M = \min\{|\mathcal{A}|, N\}$. Consider an input vector $\mathbf{x}$ such that

$$(\mathbf{x})_i = a_i \; \forall \; i \in \{1, \ldots, M\} \text{ and } (\mathbf{x})_i = \phi \text{ otherwise}$$

and a collection of M indices $i_1, i_2, \ldots, i_M$ such that each $i_j = j$. Then from (5), it can be easily verified that the maximum target function f satisfies property $\mathbf{Q}(M)$. The result then follows from Lemma 3.4. ■

4.3. K-largest Values Function

Let $\mathcal{A} = (a_1, a_2, \ldots, a_{|\mathcal{A}|})$ be an ordered set (in increasing order). For any given input vector $(x_1, x_2, \ldots, x_N)$, let $(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_N)$ denote the vector which is a permutation of the input vector and satisfies $\hat{x}_i \geq \hat{x}_{i+1}$ for each i. Then the K-largest values function is given by

$$f\left(x_1, x_2, \ldots, x_N\right) = \left(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_K\right).$$

Lemma 4.6. *There exists a* $(1, l)$ *feasible code for computing the* K*-largest values function with* $K < N/2$*, such that*

$$l \leq |\mathcal{A}| \cdot N H_q\left(\frac{K}{2N}\right).$$

- Let $\mathbf{H}$ be the $(l/\left|\mathcal{A}\right|) \times N$ parity check matrix of a q-ary code with minimum distance $K+1$.
- If source σ_i takes value a_j from the alphabet $\mathcal{A}$, then it transmits a vector which is all zero except the $(j-1) \times (l/\left|\mathcal{A}\right|) + 1$ to $j \times (l/\left|\mathcal{A}\right|)$ elements, which take values from the i-th column of $\mathbf{H}$.
- Each vector in the union subspace Π that the sink receives is parsed into $|\mathcal{A}|$ sub-vectors of length $l/\left|\mathcal{A}\right|$.
- Let $\Pi_j \subseteq \mathbb{F}_q^{l/|\mathcal{A}|}$ denote the subspace spanned by collecting the j-th sub-vector of each vector in Π.
- Thus by calculating $\dim(\Pi_{|\mathcal{A}|})$, $\dim(\Pi_{|\mathcal{A}|-1}) \ldots$, the sink can compute the K largest values.

Figure 2: A $(1, l)$ code for K-largest values function

Proof. Consider the scheme in Figure 2.

Again using the Gilbert-Varshamov bound [16] we can prove that there exists a parity check matrix such that

$$\frac{l}{|\mathcal{A}|} \leq N H_q\left(\frac{K}{2N}\right).$$

■

Comment : Again, for constant $|\mathcal{A}|$ and K, the cost only grows logarithmically with the number of sources N.

Lemma 4.7. *For the K-largest values target function f with $K < N/2$,*

$$\mathcal{E}_{min}(f) \geq \frac{N}{2} H_q\left(\frac{K}{2N}\right).$$

Proof. If the receiver can correctly compute the K-largest values, then it can also deduce if the number of active sources is greater than K or not. Thus, it can also compute the T-threshold function with the threshold $T = K$. The result then follows from Lemma 4.3. ■

5. A general scheme for computation

We now present a general method to compute functions under our network model. We will illustrate the method for boolean functions of the form $f : \mathcal{A}^N \to \{0, 1\}$. For a general function, the output can be considered as a string of bits and the above scheme can be used separately to compute each bit of the output.

Since f has boolean output, it can be written as

$$f(x_1, x_2, \ldots, x_N) = \sum_{i=1}^{s} \prod_{j=1}^{N} B_{ij}$$

where s is some integer such that $1 \leq s \leq |\mathcal{A}|^N$; $\{B_{ij}\}$ are boolean variables such that the value of B_{ij} depends only on x_j; and the sum and product represent boolean OR and AND. By taking the complement, we have

$$\overline{f(x_1, x_2, \ldots, x_N)} = \prod_{i=1}^{s} \sum_{j=1}^{N} \overline{B_{ij}}.$$

Given any input x_j, source j creates a vector v_j of length s such that i-th component is $\overline{B_{ij}}$. Each source j then sends the corresponding vector v_j into the network and the sink collects linear combinations of these vectors. If the i-th component of any of the vectors in the union subspace at the sink is 1, then a boolean variable A_i is assigned the value 1. This implies that

$$A_i = \sum_{j=1}^{N} \overline{B_{ij}}$$

and hence,

$$f(x_1, x_2, \ldots, x_N) = \overline{\prod_{i=1}^{s} A_i}.$$

Thus, we have a $(1, s)$ scheme to compute any function f with binary output.

Comment : Since the cost associated with the above code is s, the above scheme is efficient when the number of input vectors for which the function value is 1 (or 0) is much smaller than the total number of possible input vectors.

We now present an example to illustrate the above method.

Example 5.1. *Let $\mathcal{B} = \{1, 2, \ldots, K\}$ and let the source alphabet $\mathcal{A}$ be the power*

set of $\mathcal{B}$, i.e, $\mathcal{A} = 2^{\mathcal{B}}$. Then the set cover function is defined as

$$f(x_1, x_2, \ldots, x_N) = \begin{cases} 1 & \text{if } \mathcal{B} \not\subseteq \bigcup_{i=1}^{N} x_i \\ 0 & \text{otherwise.} \end{cases}$$

In words, each source observes a subset of $\mathcal{B}$ and the sink needs to compute if the union of the source messages covers $\mathcal{B}$. Define the boolean variable $\mathbb{1}_A$ as follows.

$$\mathbb{1}_A = \begin{cases} 1 & \text{if } A \text{ is true} \\ 0 & \text{otherwise.} \end{cases}$$

Then the function f can be rewritten as

$$f(x_1, x_2, \ldots, x_N) = \sum_{i=1}^{K} \prod_{j=1}^{N} \mathbb{1}_{\{i \notin x_j\}}.$$

Then using the scheme described in this section, the set cover function can be computed using a $(1, K)$ code with $d \cdot l = \log_2 |\mathcal{A}| = K$. This scheme is in-fact optimal in terms of the smallest possible cost for any feasible code.

6. Conclusions

In this paper we investigated function computation in a network where intermediate nodes perform randomized network coding, through appropriate choice of the subspace codebooks at the source nodes. Unlike traditional function computation, that requires intermediate nodes to be aware of the function to be computed, our designs are transparent to the intermediate node operations. Under this setup, we provided lower bounds on the number of transmissions required by each source to ensure successful computation. For several practically relevant target functions, such as the identity, maximum, and threshold functions, we designed schemes whose performance is close to optimal.

References

[1] A. Giridhar and P. R. Kumar, "Computing and communicating functions over sensor networks," *IEEE Journal on Selected Areas in Communication*,

vol. 23, no. 4, pp. 755–764, Apr. 2005.

[2] ——, "Towards a theory of in-network computation in wireless sensor networks," *IEEE Communications Magazine*, vol. 44, no. 4, pp. 98–107, Apr. 2006.

[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, Feb. 2011.

[4] 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.

[5] O. Gnawali, K. Jang, J. Paek, M. Vieira, R. Govindan, B. Greenstein, A. Joki, D. Estrin, and E. Kohler, "The tenet architecture for tiered sensor networks," in *Proceedings of the ACM Conference on Embedded Networked Sensor Systems (SenSys)*, Oct 2006, pp. 153–166.

[6] T. Ho, M. Medard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A random linear network coding approach to multicast," *IEEE Transactions on Information Theory*, vol. 52, no. 10, pp. 4413–4430, Oct. 2006.

[7] R. Koetter and F. R. Kschischang, "Coding for errors and erasures in random network coding," *IEEE Transactions on Information Theory*, vol. 54, no. 8, pp. 3579–3591, Aug 2008.

[8] M. Jafari Siavoshani, C. Fragouli, and S. Diggavi, "Noncoherent multisource network coding," in *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, Jul 2008, pp. 817–821.

[9] C. Fragouli, M. Jafari Siavoshani, S. Mohajer, and S. Diggavi, "On the capacity of non-coherent network coding," in *Proceedings of the IEEE International Symposium on Information Theory (ISIT)*, Jun 2009, pp. 273–277.

[10] L. Keller, N. Karamchandani, and C. Fragouli, "Function computation over linear channels," in *Proceedings of the IEEE International Symposium on Network Coding (NetCod)*, 2010.

[11] D. Silva, F. R. Kschischang, and R. Koetter, "A rank-metric approach to error control in random network coding," *IEEE Transactions on Information Theory*, vol. 54, no. 9, pp. 3951–3967, Sep 2008.

[12] 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, Feb. 2011.

[13] H. Witsenhausen, "The zero-error side information problem and chromatic numbers," *IEEE Transactions on Information Theory*, vol. 22, no. 5, pp. 592–593, Sep. 1976.

[14] V. Doshi, D. Shah, M. Medard, and S. Jaggi, "Graph coloring and conditional graph entropy," in *Proceedings of the Fortieth Asilomar Conference on Signals, Systems and Computers*, 2006, pp. 2137–2141.

[15] A. Orlitsky and J. R. Roche, "Coding for computing," *IEEE Transactions on Information Theory*, vol. 47, no. 3, pp. 903–917, Mar. 2001.

[16] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. North-Holland Mathematical Library, 1977.