Function computation via subspace coding

Nikhil Karamchandani* Lorenzo Keller† Christina Fragouli† Massimo Franceschetti*
*Dept. of Electrical and Computer Engineering, UCSD, USA
†School of Computer and Communication Sciences, EPFL, Switzerland
{Email : nikhil@ucsd.edu, lorenzo.keller@epfl.ch, christina.fragouli@epfl.ch, massimo@ece.ucsd.edu}

***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.**

I. Introduction

In sensor networks, the need for energy efficiency has stimulated research efforts towards in-network aggregation and function computation, see for example [1], [2]. Recent work [3], [4] has also 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.

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 [5]) 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 [6]–[8], 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. Here we focus on non-coherent communication where we have no knowledge about the network transformation; in [9] we look at the case where this transformation is fixed and known.

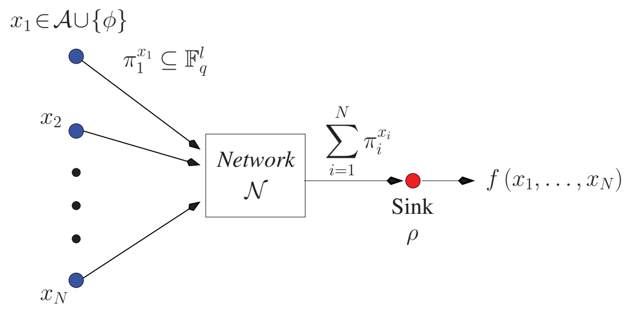
We note that a scheme which optimizes the intermediate node operations according to the function to be computed might need fewer transmissions. However, it would be more complex to implement, would require topology knowledge, and might be sensitive to the employed communication protocol. In contrast, our approach is transparent both to the topology and the employed communication protocol: the only requirement we impose is that we gather sufficient linear 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 II presents the problem formulation. In Section III, we present a simple scheme to compute the *identity* function, i.e., to reconstruct all the source messages, and then describe a class of "hard" functions for which it is optimal (in an order sense) to first compute the identity and then compute the function value. We continue by designing near-optimal coding schemes for some "easy" functions, i.e., functions which can be computed by transmitting less symbols by the sources than what is required to compute the identity: these are the T-*threshold*, *maximum* and K-*largest values* functions considered Section IV. In Section V, we present a lower bound on the number of symbols each source needs to transmit to evaluate an arbitrary function, and a constructive scheme to evaluate arbitrary functions.

II. 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

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


$x_1 \in \mathcal{A} \cup \{\phi\}$
$\pi_1^{x_1} \subseteq \mathbb{F}_q^l$
x_2
x_N
Network $\mathcal{N}$
$\sum_{i=1}^{N} \pi_i^{x_i}$
Sink ρ
$f(x_1, \ldots, x_N)$

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 $\pi_1 + \pi_2 = \{\mathbf{x} + \mathbf{y} : \mathbf{x} \in \pi_1, \mathbf{y} \in \pi_2\}$. 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 $\pi_i^{\phi} = \emptyset$.
- 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 $\sum_{i=1}^{N} \pi_i^{x_i}$.
- 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 $x_i \in \mathcal{A} \cup \{\phi\}$.

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

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

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 II.1. *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.*

[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 [6], [10].

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

Proof: From the collection of d-dimensional subspaces associated with the given (d, l) code, construct a collection of one-dimensional subspaces by concatenating the d basis vectors of each subspace. The proof follows by showing that the resulting collection also forms a feasible code for computing f. Details of the proof can be found in [11]. ∎

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 $I = (i_1, i_2, \ldots, i_{|I|}) \subseteq \{1, \ldots, N\}$. For any $\mathbf{a} = (a_1, a_2, \ldots, a_{|I|}) \in (\mathcal{A} \cup \{\phi\})^{|I|}$ 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 II.2. *If there exist one-dimensional subspaces $\pi_1, \pi_2, \ldots, \pi_K \subseteq \mathbb{F}_q^l$ such that*

$$\pi_i \not\subseteq \sum_{j<i} \pi_j \quad \forall\, i \in \{1, \ldots, K\}$$

then $l \geq K$.

III. Functions which are maximally hard to compute

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 III.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 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. ∎

Consider the case $N \geq \log_q |\mathcal{A}|$. Next, we present a class of functions for which the cost is required to grow linearly with respect to the number of sources N. Thus, the number of transmissions that each source makes for the computation of such functions is almost the same (in the order sense) as that required to reconstruct all the source messages. For any vector $\mathbf{x} \in (\mathcal{A} \cup \{\phi\})^N$, let $I_{\mathbf{x}}$ denote the index set corresponding to the components which are not ϕ. Then, consider a target function f which satisfies the following property with some constant $\alpha \in (0, 1]$.

Function property $\mathbf{P}(\alpha)$ *:* There exists a vector $\mathbf{x}^* =$

$(x_1^*, x_2^*, \ldots, x_N^*)$ with $|I_{\mathbf{x}^*}| \geq \alpha N$ such that for each $k \in I_{\mathbf{x}^*}$,

$$f\left(\mathbf{x}^*(\{k\}, \phi)\right) \neq f\left(\mathbf{x}^*\right). \tag{1}$$

Recall that $\mathbf{x}^*(\{k\}, \phi)$ denotes the vector obtained from $\mathbf{x}^*$ by setting x_k^* equal to ϕ and retaining all the other components. This implies that the function value is sensitive to whether any specific source σ_k is active or not.

Example III.2.

- *The identity function satisfies property $\mathbf{P}(1)$ by choosing each x_i^* equal to any element of the alphabet $\mathcal{A}$.*
- *The arithmetic sum function satisfies property $\mathbf{P}(1)$ by choosing each x_i^* equal to some non-zero element of the alphabet $\mathcal{A}$.*
- *The parity function ($\mathcal{A} = \{0, 1\}$) satisfies property $\mathbf{P}(1)$ by choosing each x_i^* equal to 1.*

Lemma III.3. *Let f be a function which satisfies the property $\mathbf{P}(\alpha)$. Then, $\mathcal{E}_{min}(f) \geq \alpha N$.*

Proof: From (1), any feasible code for computing f must satisfy the following condition. For each $k \in I_{\mathbf{x}^*}$,

$$\pi_k^{x_k^*} + \sum_{j \neq k} \pi_j^{x_j^*} \neq \sum_{j \neq k} \pi_j^{x_j^*} \Longrightarrow \pi_k^{x_k^*} \not\subseteq \sum_{j \neq k} \pi_j^{x_j^*}.$$

Since $|I_{\mathbf{x}^*}| \geq \alpha N$, the result follows from[4] Lemma II.2. ∎

Comment : Lemma V.1 provides a general lower bound on $\mathcal{E}_{\text{min}}(f)$ for arbitrary functions. Functions for which the lower bound is of the same order as $N + \lceil \log_q |\mathcal{A}| \rceil$ are also maximally hard to compute.

IV. Bounds for specific functions

A. T-threshold Function

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

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

Lemma IV.1. *There exists a $(1, l)$ feasible code for computing the T-threshold function with $T < N/2$, such that*

$$l \leq N H_q\left(\frac{T}{2N}\right)$$

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 binary code with minimum distance $d_{min} = T + 1$. From [12], [13], there exists such a matrix with

$$l \leq N H_q\left(\frac{T}{2N}\right).$$

∎

[4]From Lemma II.2, the result can be proven for a relaxed albeit more complicated version of the property $\mathbf{P}(\alpha)$.

[5]For any integer a, we set $a + \phi = a$. Thus, 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 binary 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.

Fig. 1. A $(1, l)$ code for the T-threshold function

Comment : For a constant T, $O\left(N H_q\left(\frac{T}{2N}\right)\right) = O\left(T \log_q N\right)$. 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.

Lemma IV.2. *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 $\{1, \ldots, N\}$,

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

Denote a basis vector for π_i^1 by $\mathbf{v_i}$. From the necessary condition on the subspaces $\pi_1^1, \pi_2^1, \ldots, \pi_N^1$, 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 for a linear code of length N and minimum distance at least $T + 1$. The result then follows by the using the bounds in [12], [13]. Details can be found in [11]. ∎

B. Maximum Function

Lemma IV.3. *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[6].

- *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

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

$\mathcal{C}_i = \left(\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v}_{|\mathcal{A}|}\right)$. 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 IV.4. *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 increasing order) and let $M = \min\{N, |\mathcal{A}|\}$. Consider two possible input vectors $(x_1, x_2, \ldots, x_N)$ and $(y_1, y_2, \ldots, y_N)$ such that

$$x_i = a_i \ \forall\ i \in \{1, \ldots, M\} \text{ and } x_i = \phi \text{ otherwise}$$
$$y_i = a_i \ \forall\ i \in \{1, \ldots, M-1\} \text{ and } y_i = \phi \text{ otherwise} .$$

Note that

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

and hence any feasible code for computing f must satisfy

$$\sum_{i=1}^{M-1} \pi_i^{a_i} + \pi_M^{a_M} \neq \sum_{i=1}^{M-1} \pi_i^{a_i} \implies \pi_M^{a_M} \not\subseteq \sum_{i=1}^{M-1} \pi_i^{a_i}.$$

The same argument can be extended to get the following necessary condition. For any subset $(i_1, i_2, \ldots, i_M)$ of $\{1, \ldots, N\}$ and any ordered subset (in increasing order) $(a_{j_1}, a_{j_2}, \ldots, a_{j_M})$ of $\mathcal{A}$,

$$\pi_{i_k}^{a_{j_k}} \not\subseteq \sum_{m<k} \pi_{i_m}^{a_{j_m}}.$$

Then the result follows from Lemma II.2. ■

C. 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

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

Lemma IV.5. *There exists a $(1, l)$ feasible code for computing the K-largest values function with $K < N/2$, such that*

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

Proof: Consider the scheme in Figure 2.

Again from [12], [13], 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 IV.6. *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 IV.2. ■

- Let $\mathbf{H}$ be the $(l/|\mathcal{A}|) \times N$ parity check matrix of a binary 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 $(j-1) \times (l/|\mathcal{A}|) + 1$ to $j \times (l/|\mathcal{A}|)$ 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 $l/|\mathcal{A}|$.
- 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 $\dim(\Pi_{|\mathcal{A}|})$, $\dim(\Pi_{|\mathcal{A}|-1})$ ..., the sink can compute the K largest values.

Fig. 2. A $(1, l)$ code for K-largest values function

V. Arbitrary Functions

A. A general lower bound

For any $\mathbf{x} \in (\mathcal{A} \cup \{\phi\})^N$ and $I \subseteq \{1, \ldots, N\}$, let

$$R_I^{\mathbf{x}}(f) = \left|\left\{f\left(\mathbf{x}(I, \mathbf{a})\right) : \ \mathbf{a} \in (\mathcal{A} \cup \{\phi\})^{|I|}\right\}\right| \tag{2}$$

denote the number of distinct values that the function takes when only the arguments corresponding to I are varied and all the others are held fixed according to $\mathbf{x}$.

Lemma V.1. *For any target function f,*

$$\mathcal{E}_{min}(f) \geq$$
$$\max_{\substack{I, \mathbf{x} : \\ R_I^{\mathbf{x}}(f) > 1}} \ \max\left\{\frac{\sqrt{\log_q\left(R_I^{\mathbf{x}}(f) - 1\right)}}{3}, \frac{\log_q\left(R_I^{\mathbf{x}}(f) - 1\right)}{3\,|I|}\right\}.$$

Proof: We skip the proof of this lemma here due to lack of space. Details can be found in [11]. ■

Example V.2.

- *For the identity target function f, the above bound gives*
$$\mathcal{E}_{min}(f) \geq \frac{\log_q |\mathcal{A}|}{3}.$$
- *For the arithmetic sum target function f, we get*
$$\mathcal{E}_{min}(f) \geq \frac{\sqrt{\log_q N\,|\mathcal{A}|}}{3}.$$

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

B. 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 $f : (\mathcal{A} \cup \{\phi\})^N \to \{0,1\}$. 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 $1 \le s \le |\mathcal{A}|^N$; $\{B_{ij}\}$ 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 V.3. *Let $\mathcal{B} = \{1, \ldots, K\}$ 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 $\mathbf{1}_A$ as follows.

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

Then the function f can be rewritten as

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

Then using the scheme described in this section, the set cover function can be computed using a $(1, K)$ code. This scheme is in-fact optimal in terms of the smallest possible cost for any feasible code.

VI. Conclusions

In this paper we investigated function computation in a network where the intermediate node operation result in an unknown linear trasnformation of the source data, 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 and oblivious to the network topology.

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

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

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

[10] 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.

[11] N.Karamchandani, L.Keller, C.Fragouli, and M.Franceschetti, "Function computation via subspace coding," *EPFL Technical Report ARNI-REPORT-2010-001, http://infoscience.epfl.ch/record/143339*, 2010.

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

[13] L. Keller, M. Siavoshani, C. Fragouli, K. Argyraki, and S. Diggavi, "Identity aware sensor networks," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, Apr 2009, pp. 2177–2185.