Linearity and Solvability in Multicast Networks

Randall Dougherty, Christopher Freiling, and Kenneth Zeger, *Fellow, IEEE*

Abstract—**It is known that for every solvable multicast network, there exists a large enough finite-field alphabet such that a scalar linear solution exists. We prove: i) every binary solvable multicast network with at most two messages has a binary scalar linear solution; ii) for more than two messages, not every binary solvable multicast network has a binary scalar linear solution; iii) a multicast network that has a solution for a given alphabet might not have a solution for all larger alphabets.**

Index Terms—**Coding, flows, network information theory, routing.**

I. Introduction

A multicast network is a directed multigraph containing a single source node and a collection of destination nodes. The source node has a set of messages from a fixed alphabet and each destination node tries to recover all of the messages.

Each node in the graph receives an alphabet symbol on each of its in-edges and transmits a symbol on each of its out-edges. Each transmitted symbol is computed at the node by a fixed encoding operation, which is a function of the symbols received. A multicast network is solvable with respect to the alphabet if the encoding operations can be assigned in such a way that the source messages can always be recovered at each destination node, using decoding operations. The encoding and decoding operations in a network collectively constitute a "code," a powerful alternative to simple network "switching" (see [19] for more background).

Ahlswede *et al.* [1] introduced the concept of network coding and gave a condition based on the well-known max-flow min-cut theorem for the solvability of multicast networks. Li *et al.* [11] studied network codes that are (scalar) linear when the message alphabet is the underlying set of a finite field. They showed that any solvable (acyclic) multicast network has a linear solution provided the finite field alphabet chosen is of large enough cardinality. Although no bounds were given in their paper on the necessary alphabet size to achieve a linear coding solution, one can deduce roughly that the alphabet had to be about the number of edges raised to the power of the number of messages. Algorithms for constructing linear codes (when they exist) over finite fields were given in [7], [5], [9]–[11].

Koetter and Médard [10] characterized linear solvability of multicast networks in an algebraic manner and showed how to calculate the linear solution guaranteed by [11]. They also showed that linear solutions exist for solvable multicast networks with some finite-field alphabet whose size is a power of two and which is at most as large as the number of source messages times the number of destination nodes. Jaggi *et al.* [9] and Ho *et al.* [8] improved the alphabet size bound of [10] by showing that every solvable multicast network with at least two destination nodes has a linear solution with some finite-field alphabet of size at most equal to the number of destination nodes.

Feder *et al.*[6] showed that to achieve a linear solution, some solvable multicast networks asymptotically require finite-field alphabets to be at least as large as twice the square root of the number of destination nodes. They also provide some upper bounds on the minimum alphabet size for linear solutions. The lower bound proof in [6] shows the solvability of a specially constructed network, and bounds its alphabet size for a linear solution, but no nonlinear solution of lower alphabet size is given. Also, the networks they use are not solvable over the binary field. Rasala Lehman and Lehman [15] constructed a similar multicast network as in [6] (in independent work) and also achieved essentially the same square root lower bound as in [6]. The result in [15] actually shows that the square root lower bound on alphabet size applies to finding any solution, not just a linear solution. They also gave an example of a multicast network which is solvable over a ternary alphabet but which has no linear solution for alphabets of cardinality less than five. Furthermore, it was shown in [15] that the problems of determining the minimum alphabet size for both linear and nonlinear solutions to a multicast network are NP-hard.

It was additionally shown in [15] that there exist nonmulticast networks which are solvable, but which have no linear solution for any finite field alphabet size. Independently, Riis [16] constructed a nonmulticast network which is solvable over the binary field but which has no binary linear solution. Médard *et al.* [14] present an example (attributed to Koetter) of a nonmulticast network which has no linear solution, but does have a certain nonlinear solution which, under a broader definition of "solution" that allows coding over multiple time units, can be viewed as a "vector-linear" solution.[1] The authors of [14] conjecture that every solvable network has a vector-linear solution, and give a coding theorem which may be useful in analyzing nonmulticast network solutions. Riis notes in [16] that

Manuscript received January 5, 2004; revised May 13, 2004. This work was supported by the Institute for Defense Analyses, the National Science Foundation, and Ericsson. The material in this paper was presented in part at the Conference on Information Sciences and Systems, Princeton University, Princeton, NJ, March 2004.

R. Dougherty is with the Center for Communications Research, San Diego, CA 92121-1969 USA (e-mail: rdough@ccrwest.org).

C. Freiling is with the Department of Mathematics, California State University, San Bernardino, San Bernardino, CA 92407-2397 USA (e-mail: cfreilin@csusb.edu).

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

Communicated by M. Médard, Associate Editor for Communications.

Digital Object Identifier 10.1109/TIT.2004.834751

[1]Without the linearity requirement, solutions over multiple time units can be viewed as solutions over one time unit but with higher cardinality alphabets.

his linearly unsolvable nonmulticast network has a three-dimensional vector-linear solution over the binary field (attributed to Koetter). He also showed in [16] that any solvable multicast network has a vector-linear solution with binary components, and that there exist solvable nonmulticast networks which can achieve vector-linear solutions only if the alphabet size grows linearly with the number of nodes in the network.

Implementation of network codes for large alphabet sizes may be difficult due to complexity constraints. For example, a network that transmits finite-field elements along its links might be implemented by representing field elements as binary vectors (if the field size is a power of two) and then transmitting the binary vectors either all at once or bit by bit over multiple units of time. The finite-field arithmetic used in implementing a linear code mandates that all such bits in a binary vector arrive before the arithmetic can be performed. Thus, either a large delay or a large transmission bandwidth on each link would be required. This motivates the use of a small alphabet, such as a binary field.

Although linear solutions are guaranteed by [11] for large enough finite-field alphabets, the results in [11] do not guarantee the existence of a linear solution for a solvable multicast network if the alphabet size is fixed, nor do they consider alphabets whose cardinalities are not integer powers of primes. To our best knowledge, there are no prior results in the literature, other than the previously mentioned lower bounds in [6] and [15], and the ternary alphabet nonlinear solution in [15], about the existence or nonexistence of linear solutions for solvable multicast networks with a fixed finite-field alphabet. In recent independent work, Riis (in collaboration with Ahlswede) [16] showed there exists a multicast network with five messages that is solvable over the binary field but which has no linear solution over the binary field. Their network is based on the nonlinear Nordstrom–Robinson $(12, 5, 5)$ error-correcting code. It has been unknown whether binary solvable multicast networks with three or four messages necessarily are linearly solvable.

In this paper, we address the fixed-alphabet issue for binary alphabets; namely, when each edge in a multicast network carries only a single bit. First, for completeness, we prove (Theorem III.1) that if a multicast network with at most two messages is solvable over the binary field then it has a linear solution over the binary field.[2] Our proof shows precisely how to convert a nonlinear solution into a linear solution. In contrast, we show that this result need not be true if there are more than two messages. That is, we prove (Theorem IV.3) that if the number of messages is greater than two, then there exists a multicast network that is solvable over the binary field but does not have a linear solution over the binary field. Our proof is constructive; we specify such a network, give a nonlinear solution, and then prove it has no linear solution. Finally, we show (Theorem V.2) that a multicast network that has a solution for a given alphabet might not have a solution for all larger alphabets. In this case, we do not restrict our attention necessarily to finite-field alphabets. Our proof demonstrates an interesting connection between network solvability and a 45–year-old theorem on the existence of orthogonal latin squares. Proofs of all lemmas are given in Section VI.

[2]David Karger has communicated to us that he has also obtained Theorem III.1 in independent work.

II. Preliminaries

In this section, we give formal definitions of various terms used in the results to follow.

A *multicast network* is a 4-tuple $(\mathcal{V}, \mathcal{E}, s, \mathcal{D})$, where $(\mathcal{V}, \mathcal{E})$ is a finite connected directed acyclic multigraph, $s \in \mathcal{V}$, and $\mathcal{D} \subset \mathcal{V}$. The elements of $\mathcal{V}$ are called *nodes* and the elements of $\mathcal{E}$ are called *edges*. Since $\mathcal{E}$ is a multiset, parallel edges are allowed. The elements of $\mathcal{V} - \mathcal{D} - \{s\}$ are *interior nodes*, the node s is a *source node*, and the elements of $\mathcal{D}$ are *destination nodes*. Every edge $(a, b) \in \mathcal{E}$ is called an *in-edge* of node b and an *out-edge* of node a.

For each node $n \in \mathcal{V}$, let $\mathcal{E}_i(n)$ and $\mathcal{E}_o(n)$ be the set of in-edges and out-edges of n, respectively. The cardinality $|\mathcal{E}_i(n)|$ (respectively, $|\mathcal{E}_o(n)|$) is called the *in-degree* (respectively, *out-degree*) of node n. Let

$$m = |\mathcal{E}_o(s)|$$

and

$$\mathcal{E}_o(s) = \{e_{s,1}, \dots, e_{s,m}\}.$$

For a multicast network we also require that $|\mathcal{E}_i(n)| \geq 1$ for all $n \in \mathcal{V} - \{s\}$, so that the source is the only node with no in-edges.

Let $\mathcal{A}$ be a finite set, with at least two elements, called an *alphabet*. For a given multicast network, alphabet $\mathcal{A}$, and edge $e \in \mathcal{E}$, a *valuation* of the edge e is a mapping

$$v_e : \mathcal{A}^m \to \mathcal{A}.$$

We extend the definition of v_e to ordered sets of edges $U = \{e_1, \dots, e_k\}$ by defining $v_U = (v_{e_1}, \dots, v_{e_k})$. Let

$$\mu = v_{\mathcal{E}_o(s)}$$

and note that $|\mu| = m$. The elements of μ are called *messages* and are said to be *emitted* by the source. A specific instance of the set of messages is any m-tuple $a \in \mathcal{A}^m$, and for such an instance the valuation of any edge $e \in \mathcal{E}$ yields $v_e(a) \in \mathcal{A}$. The valuation of an edge in a network indicates the information carried by the edge as a function of the messages emitted by the source. A decoding *valuation* is a mapping

$$v_{n,x} : \mathcal{A}^m \to \mathcal{A}$$

for $n \in \mathcal{D}$, $x \in \mu$. The edge valuations and decoding valuations will collectively be referred to as the "valuations."

A *code* is a collection of *encoding operations*

$$\phi_e : \mathcal{A}^{|\mathcal{E}_i(n)|} \to \mathcal{A}$$

for all $n \in \mathcal{V} - \mathcal{D} - \{s\}$ and $e \in \mathcal{E}_o(n)$, and *decoding operations*

$$\phi_{n,x} : \mathcal{A}^{|\mathcal{E}_i(n)|} \to \mathcal{A}$$

for all $n \in \mathcal{D}$ and $x \in \mu$. The encoding operations assign to each out-edge of a node an element of $\mathcal{A}$ as a function of the

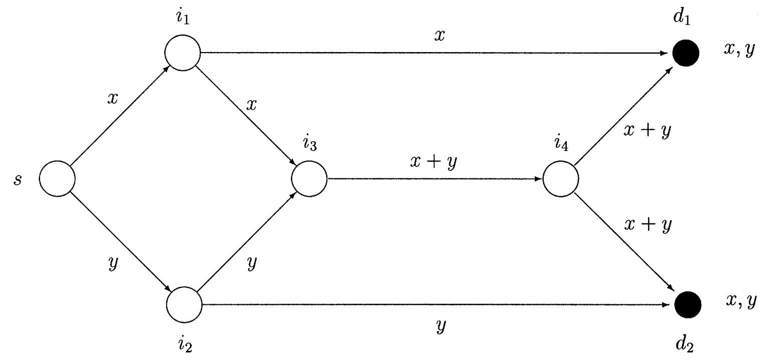


Fig. 1. Example of a multicast network with messages x, y, source node s, interior nodes i_1, i_2, i_3, i_4, and destination nodes d_1, d_2. The valuations define a code, which is a (linear) solution since the destination nodes can recover the messages x and y. The decoding operations for nodes d_1 and d_2 are subtractors or projections, whereas all encoding operations are adders or projections.

elements of $\mathcal{A}$ assigned to the in-edges of the same node. The decoding operations at a given node generate elements of $\mathcal{A}$, in correspondence with the source messages, and as functions of the elements of $\mathcal{A}$ assigned to the in-edges of the node. The goal of the encoding operations on out-edges of interior nodes is to help transfer the source messages through the multicast network to the destination nodes, and the goal of the decoding operations at destination nodes is to produce each of the source messages. A mapping $f : \mathcal{A}^k \rightarrow \mathcal{A}$ is a *projection* function if for some $i \in \{1, \dots, k\}$, $f(u_1, \dots, u_k) = u_i$ for all $u_1, \dots, u_k \in \mathcal{A}$.

A collection of valuations are *induced* by a code if the edge valuations satisfy $\phi_e(v_{\mathcal{E}_i(n)}) = v_e$, for all $n \in \mathcal{V} - \mathcal{D} - \{s\}$ and all $e \in \mathcal{E}_o(n)$; and the decoding valuations satisfy $\phi_{n,x}(v_{\mathcal{E}_i(n)}) = v_{n,x}$ for all $n \in \mathcal{D}$ and all $x \in \mu$; and $v_{e_{s,i}}(x_1, \dots, x_m) = x_i$ for $i = 1, \dots, m$. If a node n has in-edges $\mathcal{E}_i(n) = \{e_1, \dots, e_i\}$ and an out-edge e, then the composition $\phi_e(v_{e_1}, \dots, v_{e_i}) : \mathcal{A}^m \rightarrow \mathcal{A}$ is a function, which when given a specific instance of messages, produces a specific alphabet element assigned to the edge e. An analogous statement holds for decoding operations. For a connected acyclic graph, each code induces exactly one valuation.

A *solution* is a code such that $v_{n,x} = x$, for each destination node $n \in \mathcal{D}$ and each message $x \in \mu$, for the induced valuation. If $n \in \mathcal{D}$ and $v_{n,x} = x$, then n is said to *recover* the message x. A solution allows each destination node to recover all the messages emitted by the source. A multicast network is said to be *solvable* for a given alphabet if there exists a solution.

It can be useful to impose an algebraic structure on $\mathcal{A}$, such as a ring or a field. In such a case, a code is *linear* (respectively, *affine*) with respect to $\mathcal{A}$, if for each $e \in \mathcal{E}$, $n \in \mathcal{D}$, and $x \in \mu$, the functions ϕ_e and $\phi_{n,x}$ are linear (respectively, affine) over $\mathcal{A}$. If $|\mathcal{A}| = 2$ then $\mathcal{A}$ is identified with the binary field Z_2, a solution is called a *binary solution*, and the network said to be *binary solvable*. In any linear solution, all induced valuations are clearly linear functions of the source messages. A function of the type $f : Z_2^k \rightarrow Z_2$ is said to be a *k-input Boolean function*.

While solutions to a network can be described either as "linear" or "nonlinear" with respect to specific rings of cardinality $|\mathcal{A}|$, it is most common to assume $\mathcal{A}$ is a finite field when $|\mathcal{A}|$ is a power of a prime.

Fig. 1 shows an example of a multicast network described in [1]. A linear solution is shown in the example, where each edge valuation is written as a function of the messages x and y emitted by the source.

For a multicast network with a given code, for any edge $e \in \mathcal{E}$ the induced valuation v_e can be viewed as a function of the messages. Also, if $\mathcal{A}$ is a finite field, then any function $f : \mathcal{A}^k \rightarrow \mathcal{A}$ can be written as a polynomial of degree at most $k(|\mathcal{A}| - 1)$ using the Lagrange interpolation formula

$$f(x_1, \dots, x_k) = \sum_{(v_1, \dots, v_k) \in \mathcal{A}^k} \prod_{i=1}^{k} \prod_{u \in \mathcal{A} \setminus \{v_i\}} (x_i - u)(v_i - u)^{-1} f(v_1, \dots, v_k).$$

Thus, for a finite-field alphabet, each mapping v_e is always a polynomial function of the elements of μ, with coefficients in the field $\mathcal{A}$. In what follows we will frequently make use of such polynomial representations.

The ith argument of a function $f : \mathcal{A}^k \rightarrow \mathcal{A}^j$ is said to be *recoverable* from f if there exists a mapping $g : \mathcal{A}^j \rightarrow \mathcal{A}$ such that $g(f(u_1, \dots, u_k)) = u_i$ for all $(u_1, \dots, u_k) \in \mathcal{A}^k$. A set of functions $f_1, \dots, f_j : \mathcal{A}^m \rightarrow \mathcal{A}$ is *complete* if every argument is recoverable from the function $(f_1, \dots, f_j) : \mathcal{A}^m \rightarrow \mathcal{A}^j$.

For any multicast network, define a *topological ordering of the nodes* to be a bijection $\sigma : \{1, 2, \dots, |\mathcal{V}|\} \rightarrow \mathcal{V}$ such that $\sigma^{-1}(a) < \sigma^{-1}(b)$ whenever $(a, b) \in \mathcal{E}$. Such a function σ always exists since the digraph $(\mathcal{V}, \mathcal{E})$ is acyclic. Note that since the source is the only node with in-degree zero, it must be the case that $\sigma(1) = s$. Also, define a *topological ordering of the edges* to be any bijection $\hat{\sigma} : \{1, 2, \dots, |\mathcal{E}|\} \rightarrow \mathcal{E}$ satisfying, for some topological ordering of the nodes σ, $\hat{\sigma}^{-1}(e) < \hat{\sigma}^{-1}(e')$ whenever $e \in \mathcal{E}_o(u)$, $e' \in \mathcal{E}_o(u')$, and $\sigma^{-1}(u) < \sigma^{-1}(u')$.

For an alphabet $\mathcal{A}$ that is a ring, a function $f : \mathcal{A}^k \rightarrow \mathcal{A}$ is *homogeneous* if $f(\{0\}^k) = 0$. A solution is *homogeneous* if all induced valuations are homogeneous. The valuations in a homogeneous solution for a finite-field alphabet, when expressed as polynomials, do not have additive constant terms. The following lemma allows us to restrict attention to homogeneous solutions when analyzing the solvability of a multicast network.

Lemma II.1: For a given finite-field alphabet and multicast network, if there exists a solution, then there exists a homogeneous solution. Furthermore, if there exists an affine solution, there there exists a linear solution.

We have chosen various terminologies in this paper for convenience and note that alternatives may be found in the literature.[3] In the figures in this paper, destination nodes are depicted as solid circles, and interior nodes are depicted as hollow circles.

III. Linear Codes Suffice for two Messages

In this section, we show how to create a linear solution from a nonlinear solution if the multicast network has at most two messages, and the alphabet is the binary field Z_2.

The main idea of the proof of Theorem III.1 below can be seen intuitively by an example. Suppose some encoding operation is, say, the product (i.e., logical "AND") of its two inputs, and suppose the input edges have valuations $xy+x$ and $x+y$, where x and y are the source messages. If we discard the nonlinear portions of the inputs and output, then the inputs become x and $x+y$ and the output becomes x, which agrees with the linearization of the product $(xy+x)(x+y)$. This effect can be achieved by replacing the nonlinear encoding operation by a linear encoding operation which is the projection function that copies its first input to its output. Our proof shows that every such nonlinear encoding operation can be replaced by a linear edge function, such that if the inputs were linearized, then so would the output be linearized. Then, an induction argument finishes the proof.

Theorem III.1: Every binary solvable multicast network with at most two messages has a binary linear solution.

Proof: A multicast network with exactly one message is solvable if and only if there is a directed path from the source node to every destination node, in which case a linear solution exists: simply label every edge in all such paths with the single message.

By Lemma II.1, one may assume without loss of generality that a solution to a solvable multicast network is homogeneous. So assume we have a binary solvable multicast network with exactly two messages x and y and a homogeneous solution.

It suffices to prove that every binary solvable multicast network that has at most two messages and only two-input encoding operations has a linear solution. This is because any k-input Boolean function is logically equivalent to some circuit consisting of only two-input Boolean functions, the linearity of which implies the linearity of the multiple-input configuration. By homogeneity, any encoding operation with exactly one input is either the identity function or else the constant 0. These can, respectively, be thought of as two-input Boolean functions of the form $f(a,b)=a$ and $f(a,b)=0$. Thus, we assume all encoding operations and decoding operations have exactly two inputs.

Consider a given nonlinear solution. There are 16 two-input Boolean functions, half of which are homogeneous. Hence, for all $e \in \mathcal{E}$, the induced edge valuation v_e lies in the set

$$\mathcal{P} = \{0, x, y, x+y, xy, xy+x, xy+y, xy+x+y\}.$$

For any $f \in \mathcal{P}$, let $\mathrm{L}(f) \in \mathcal{P}$ be the linear function defined by

$$\mathrm{L}(f) = \begin{cases} f, & \text{if } f \text{ is linear} \\ xy+f, & \text{if } f \text{ is not linear.} \end{cases}$$

The function $\mathrm{L}(f)$ "linearizes" f by deleting the xy term if it appeared in the polynomial. For each edge $e=(a,b)$ with $\mathcal{E}_i(a)=\{e_1,e_2\}$, define a function $\tau_e : Z_2^2 \to Z_2$ by

$$\tau_e = \begin{cases} \pi_1, & \text{if } v_{e_1}=v_{e_2} \text{ or } v_{e_1} \neq v_{e_2} = x+y \\ \pi_2, & \text{if } v_{e_2} \neq v_{e_1} = x+y \\ 0, & \text{otherwise} \end{cases}$$

where π_1 and π_2 are the projection functions defined by

$$\pi_1(a,b) = a$$
$$\pi_2(a,b) = b$$

for all a, b. The given solution determines the polynomials v_e, for all $e \in \mathcal{E}$. Thus, for every edge e, the solution determines the function τ_e, which is either the projection π_1, the projection π_2, or the constant 0 function. In all three cases, τ_e is linear.

The function τ_e gives the linear part of the product of two linear functions of x and y. One can verify by examining all cases that

$$\mathrm{L}(fg) = \tau_e(\mathrm{L}(f), \mathrm{L}(g))$$

for any $f, g \in \mathcal{P}$.

For each $e \in \mathcal{E}$, suppose the encoding operation at e is

$$\phi_e(a,b) = C_1 a + C_2 b + C_3 ab$$

for all $a, b \in \{0,1\}$, and some $C_1, C_2, C_3 \in \{0,1\}$. Define

$$\hat{\phi}_e(a,b) = C_1 a + C_2 b + C_3 \tau_e(a,b)$$

for $a, b \in \{0,1\}$. The (nonlinear) product in ϕ_e is replaced by the linear function τ_e in $\hat{\phi}_e$. It follows that, for any $e=(a,b) \in \mathcal{E}$ if $\mathcal{E}_i(a)=\{e_1,e_2\}$, then

$$\begin{aligned} &\hat{\phi}_e(\mathrm{L}(v_{e_1}), \mathrm{L}(v_{e_2})) \\ &\quad = C_1\mathrm{L}(v_{e_1}) + C_2\mathrm{L}(v_{e_2}) + C_3\tau_e(\mathrm{L}(v_{e_1}), \mathrm{L}(v_{e_2})) \\ &\quad = C_1\mathrm{L}(v_{e_1}) + C_2\mathrm{L}(v_{e_2}) + C_3\mathrm{L}(v_{e_1}v_{e_2}) \\ &\quad = \mathrm{L}(C_1v_{e_1} + C_2v_{e_2} + C_3v_{e_1}v_{e_2}) \\ &\quad = \mathrm{L}(\phi_e(v_{e_1}, v_{e_2})). \end{aligned} \tag{1}$$

[3] Some definitions of multicast network used by other authors do not assume the graph to be acyclic. For convenience, in this paper we build the acyclicity assumption into the definition. In the literature, sometimes what we call nodes are called *vertices*, edges are called *links* or *channels*, destination nodes are called *terminal nodes* or *sinks*, projection functions are called *switches* or *routing* functions, and solutions are called *instantaneous solutions* or *scalar solutions*. Sometimes in the literature a network is said to have a solution if it is solvable (in our sense) for some alphabet, as opposed to being solvable with respect to a specific alphabet. Also, sometimes multicast networks are defined with sources that emit arbitrary functions of the messages, instead of just the messages themselves.

For each $e \in \mathcal{E}$, replace ϕ_e by $\hat{\phi}_e$, and perform analogous replacements for each decoding operation at the destination nodes.

Let σ be a topological ordering of the edges. For each $e \in \mathcal{E}$ let v_e and $\hat{v}_e$ be the edge valuations induced by the codes $\{\phi\}$ and $\{\hat{\phi}\}$, respectively (and similarly for decoding valuations). Then we can prove by induction that

$$\hat{v}_{\sigma(k)} = \mathrm{L}(v_{\sigma(k)})$$

for all k, where the inductive step follows from (1). The same argument also applies to the decoding valuations.

The decoding valuations in a solution are linear, so

$$\hat{v}_{n,x} = \mathrm{L}(v_{n,x}) = v_{n,x}$$

for all demand nodes n and messages x. Thus, we have achieved a linear solution. □

IV. Linear Codes do not Suffice for More Than Two Messages

In this section, we construct in Theorem IV.3 a specific binary solvable multicast network, and then show it has no binary linear solution. This demonstrates that Theorem III.1 is tight in the sense that for binary solvable multicast networks, if the number of messages is upper-bounded by anything larger than two, then one cannot generally guarantee a binary linear solution. In this section, all arithmetic is performed over the binary field Z_2.

The network constructed in Theorem IV.3 has three messages and its nodes have in-degree at most four. The intuition behind the network construction is that we use a slight variation of the circuit in Fig. 1 as a building block to force the valuations of various edges to be linear functions, and then use these forced signals as inputs in combination with nonlinear majority voting functions at destination nodes. The combination of linear functions and majority vote functions makes it impossible to replace the majority vote functions by linear functions.

To motivate the network used in Theorem IV.3, consider a small piece of the network consisting of the four destination nodes shown in Fig. 2. The network has three binary messages x, y, z and each destination node demands all three messages. First, we will guarantee that any solution to the network forces the edges labeled in Fig. 2 by linear functions of x, y, z to have those labels as their valuations. Note that with the nonlinear assignments (where M is majority voting)

$$\begin{aligned} Q &= \mathrm{M}(x,y,z) \\ R &= \mathrm{M}(y,z,x+y+z) \end{aligned}$$

the demands of all four nodes in Fig. 2 are met, since in such case the following relations hold:

$$\begin{aligned} x &= Q + (x+z) + (x+z)(y+z) \\ &= R + (x+z)(x+y) + (x+z) + (x+y) \\ &= Q + [y + (x+y+z)](Q+R+1) \\ &= Q + [z + (x+y+z)](Q+R+1) \\ y &= Q + (y+z) + (x+z)(y+z) \\ &= R + (x+z) + (x+z)(x+y) \\ &= Q + [z + (x+y+z)](Q+R) \\ z &= Q + (x+z)(y+z) \\ &= R + (x+y) + (x+z)(x+y) \\ &= Q + [y + (x+y+z)](Q+R). \end{aligned}$$

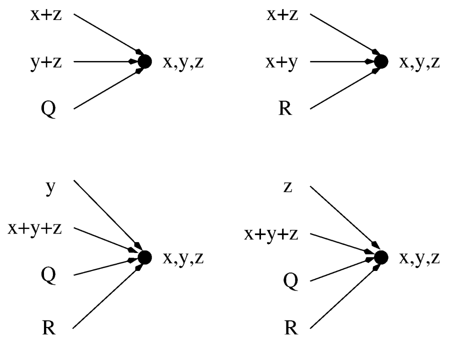


Fig. 2. Four destination nodes in a network.

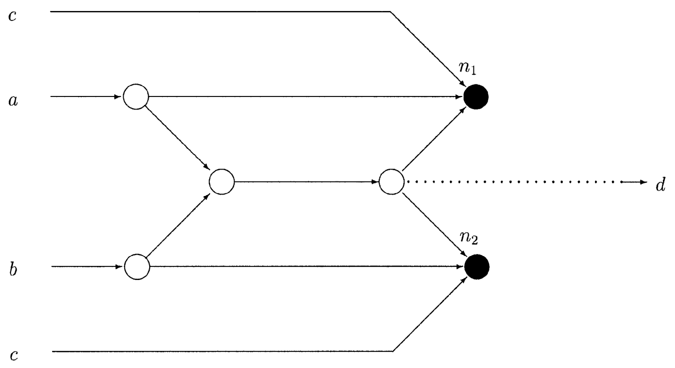


Fig. 3. Diagram of the circuit $\mathcal{G}_{a,b,c}$ where a, b, c, d are valuations of the indicated edges. The nodes n_1 and n_2 are destination nodes and the other nodes are interior nodes. The quantity d is called the *output* of the circuit.

Now, suppose there was a linear solution. What possible linear functions could the edges labeled Q and R carry? It is clear that the first destination node cannot have its demands met unless $Q \notin \{0, x+y, y+z, z+x\}$. Thus, we have narrowed down the set of possible linear functions for Q to $\{x, y, z, x+y+z\}$. An identical argument using the second destination node gives the same set of possibilities for R. Now, if $Q = x+y+z$, then the node in the third figure shows that $R \in \{x, z\}$ and the node in the fourth figure shows that $R \in \{x, y\}$. Thus, we conclude that if $Q = x+y+z$, then $R = x$. In an intuitive sense, this makes it difficult for a linear solution to allow $Q = x+y+z$, for it would force the value of R. By continuing along these lines, we can create so many difficult constraints, that eventually, no possible linear assignments will exist, and yet majority-linear assignments will still be satisfactory.

The circuit shown in Fig. 3 will be used as a building block in part of Theorem IV.3.

Lemma IV.1: If the circuit $\mathcal{G}_{a,b,c}$ appears in a multicast network with three messages, then for any homogeneous binary solution and any complete set of linear induced edge valuations $\{a, b, c\}$, the induced edge valuation d must be $a+b$ or the constant 0.

The *majority vote* function is a three-input Boolean function M defined for all $a, b, c \in Z_2$ by

$$\mathrm{M}(a,b,c) = ab + bc + ca.$$

A three-input Boolean function f is *majority-linear* if there exist three-input linear Boolean functions a, b, c such that $f = \mathrm{M}(a,b,c)$.

The following facts will be used frequently:

- $\mathrm{M}(a, a+1, b) = b$
- $\mathrm{M}(a,b,c) = c + (a+c)(b+c)$
- $\mathrm{M}(a,a,b) = a$

for all $a, b, c \in Z_2$.

Lemma IV.2: If a, b, c, and d are linear functions such that $\{a,b,c\}$ and $\{a,b,d\}$ are complete, then the following are also complete:

a) $\{a+b, a+c, \mathrm{M}(a,b,c)\}$
b) $\{a, b, \mathrm{M}(a,b,c), \mathrm{M}(a,d,a+b+d)\}$
c) $\{b, \mathrm{M}(a,b,c), \mathrm{M}(a,a+b,c), \mathrm{M}(c,a+b,a+c)\}$
d) $\{b, \mathrm{M}(a,b,c), \mathrm{M}(a,b+c,c), \mathrm{M}(b,a+b,a+c)\}$
e) $\{b, \mathrm{M}(b,c,a+b+c), \mathrm{M}(c,a+b,b+c), \mathrm{M}(c,a+b,a+c)\}$.

Theorem IV.3: For every $m \geq 3$, there exists a binary solvable multicast network with m messages that does not have a binary linear solution.

Proof: We prove the result for $m = 3$; it is straightforward to extend it to all $m \geq 3$ by adding to our network $m-3$ nodes which each receive one out-edge from the source, and then copy their inputs to all destination nodes. Throughout this proof, edge valuations will be called "linear" (respectively, "majority-linear") to mean that they are linear (respectively, majority-linear) functions of the source messages x, y, z. Define a multicast network $\mathcal{N}_1$ with three messages x, y, z and a code over the binary alphabet to consist of the following components:

i) A source node s with its three out-edges labeled x, y, z.
ii) Circuits $\mathcal{G}_{x,y,z}$, $\mathcal{G}_{y,z,x}$, $\mathcal{G}_{z,x,y}$ with output encoding operations $x+y$, $y+z$, $x+z$, respectively.
iii) Circuit $\mathcal{G}_{x+y,z,y}$ with output encoding operation $x+y+z$.
iv) A three-input, one-output interior node $n^{(1)}(A,B,C)$ for each complete set of linear inputs $\{A,B,C\}$, and with output encoding operation $\mathrm{M}(A,B,C)$.
v) A three-input destination node $n^{(2)}(A,B,M_1)$ for each complete set of inputs $\{A,B,M_1\}$, where A and B are linear and M_1 is majority-linear.
vi) A four-input destination node $n^{(3)}(A,B,M_1,M_2)$ for each complete set of inputs $\{A,B,M_1,M_2\}$, where A and B are linear and M_1 and M_2 are majority-linear.
vii) A four-input destination node $n^{(4)}(A,B,C)$ for each complete set of inputs

$$\{B, \mathrm{M}(A,B,C), \mathrm{M}(A,A+B,C), \mathrm{M}(C,A+B,A+C)\}$$

for all linear complete sets $\{A,B,C\}$.
viii) A four-input destination node $n^{(5)}(A,B,C)$ for each complete set of inputs

$$\{B, \mathrm{M}(A,B,C), \mathrm{M}(A,B+C,C), \mathrm{M}(B,A+B,A+C)\}$$

for all linear complete sets $\{A,B,C\}$.
ix) A four-input destination node $n^{(6)}(A,B,C)$ for each complete set of inputs

$$\{B, \mathrm{M}(B,C,A+B+C), \\ \mathrm{M}(A+B,B+C,C), \mathrm{M}(C,A+B,A+C)\}$$

for all linear complete sets $\{A,B,C\}$.
x) A one-input, multiple-output interior node $n^{(7)}(A)$ for each A that is linear or majority-linear, with each encoding operation being the identity function.

The network $\mathcal{N}_1$ is illustrated in Figs. 4 and 5. The nodes defined in item x) are used to make sure that multiple uses of any given linear or majority-linear edge valuation A in $\mathcal{N}_1$ come from the same place. That is, if two nodes in $\mathcal{N}_1$ each have A as an input edge valuation, then the corresponding edges are out-edges of the same node $n^{(7)}(A)$. Thus, $n^{(7)}(A)$ nodes are defined for $A = x, y, z$ after item i), for $A = x+y$, $y+z$, $x+z$ after item ii), for $A = x+y+z$ after item iii), and for all majority-linear A after item iv). All items in $\mathcal{N}_1$ can be constructed from items previously defined in $\mathcal{N}_1$ and the corresponding nodes in x).

The code described in the definition of the circuit $\mathcal{N}_1$ is a solution. This follows immediately from the fact that every destination node is defined to have a complete set of inputs.

We next show that the multicast network $\mathcal{N}_1$ does not have a binary linear solution. Assume, to the contrary, that there exists a linear solution. Then each edge of the digraph which is labeled by some (possibly nonlinear) function gets replaced by some linear function. Also, no edge already labeled with a linear function can be replaced by anything other than itself, since otherwise a destination node in some $\mathcal{G}$-circuit would not be able to recover the source messages. So only the majority-linear functions can potentially get replaced by linear functions in $\mathcal{N}_1$.

Note that if A, B, C, D are linear functions and $\mathrm{M}(A,B,C)$ is replaced by D somewhere, then $\mathrm{M}(A,B,C)$ must be replaced by D everywhere in $\mathcal{N}_1$.

We use the notation

$$\mathrm{M}(A,B,C) \rightsquigarrow D$$

to mean that the nonlinear edge valuation $\mathrm{M}(A,B,C)$ occurs in the given solution for $\mathcal{N}_1$ and is replaced everywhere it occurs by the linear valuation D in the linearly labeled network. If S is a set of linear functions then the notation

$$\mathrm{M}(A,B,C) \rightsquigarrow S$$

will mean that there exists $D \in S$ such that $\mathrm{M}(A,B,C) \rightsquigarrow D$.

We next state a lemma which constrains the linear replacements that are possible for various majority-linear functions. The remainder of the proof of the theorem will show that the constraints become so restrictive that, in fact, they lead to the conclusion that no such linear replacements are possible in a solution.

Lemma IV.4: In the multicast network $\mathcal{N}_1$, the following replacement rules hold.

a) For every complete set of linear functions $\{A,B,C\}$, it cannot be the case that $\mathrm{M}(A,B,C) \rightsquigarrow 0$.
b) $\mathrm{M}(A,B,C) \rightsquigarrow \{A, B, C, A+B+C\}$ for all complete sets of linear functions $\{A,B,C\}$.

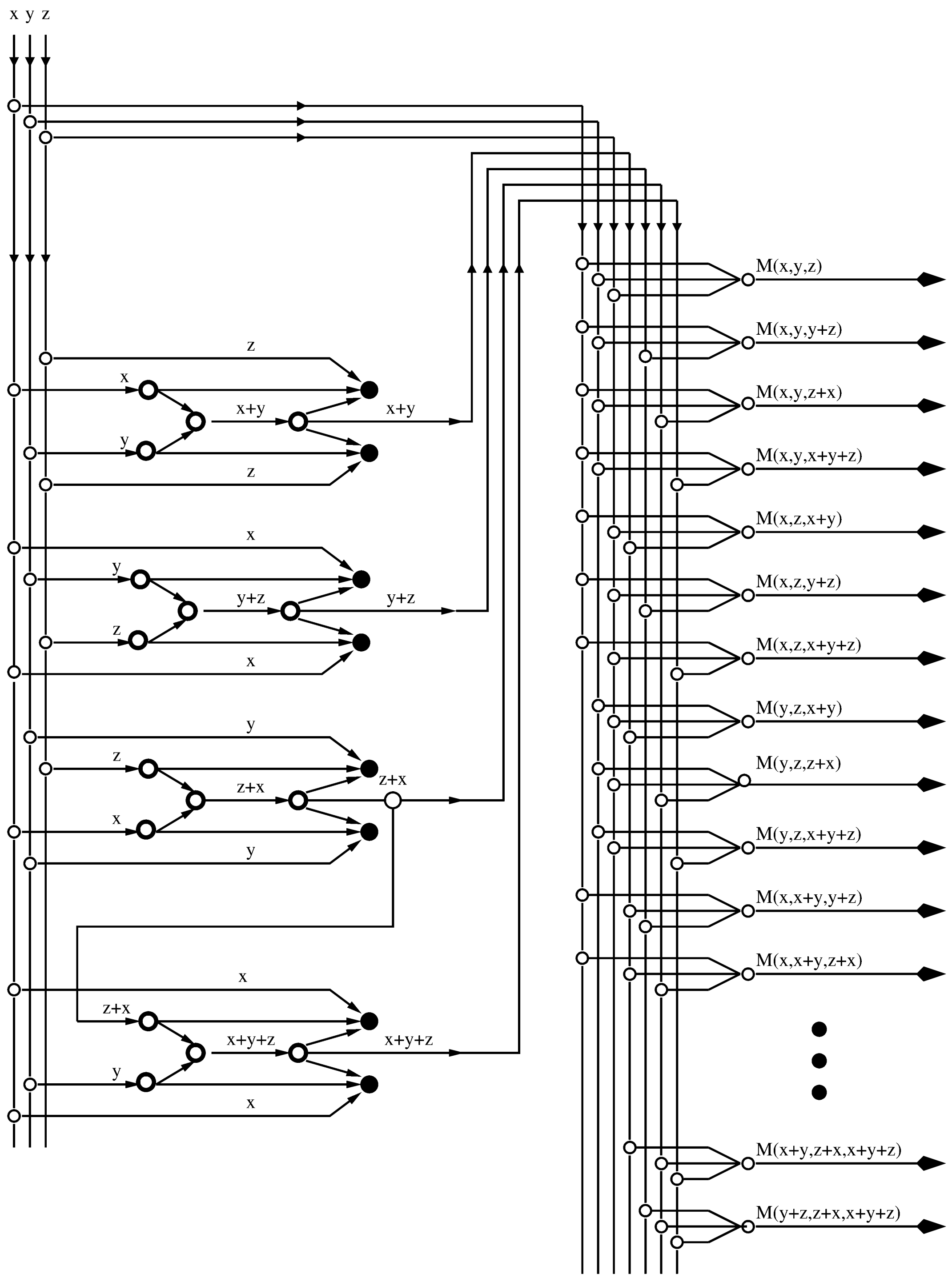


Fig. 4. The circuit on the left produces edge valuations which are forced in a solution to be the linear functions $x+y$, $y+z$, $z+x$, and $x+y+z$, where x, y, z are the messages. The circuit on the right produces all edges in the network corresponding to majority-linear functions of the form $\mathrm{M}(A,B,C)$, for all complete sets of linear functions $\{A,B,C\}$.

c) If $\mathrm{M}(A,B,C) \rightsquigarrow A$, then
$$\mathrm{M}(B,C,A+B+C) \rightsquigarrow A+B+C.$$
d) If $\mathrm{M}(A,B,C) \rightsquigarrow A+B+C$, then
$$\mathrm{M}(B,C,A+B+C) \rightsquigarrow A.$$
e) If $\mathrm{M}(A,B,C) \rightsquigarrow A$, then $\mathrm{M}(A,B,A+B+C) \rightsquigarrow B$.

f) If $\mathrm{M}(A,B,C) \rightsquigarrow A$, then
$$\mathrm{M}(A,B,D) \rightsquigarrow \{A,B\}$$
for all linear D such that $\{A,B,D\}$ is complete.

g) If $\mathrm{M}(A,B,C) \rightsquigarrow \{C,A+B+C\}$, then
$$\mathrm{M}(A,B,D) \rightsquigarrow \{D,A+B+D\}$$
for all linear D such that $\{A,B,D\}$ is complete.

h) If $\mathrm{M}(A,B,C) \rightsquigarrow \{A,A+B+C\}$, then
$$\mathrm{M}(A,D,B+C+D) \rightsquigarrow \{A,A+B+C\}$$
for all linear D such that $\{A,D,B+C+D\}$ is complete.

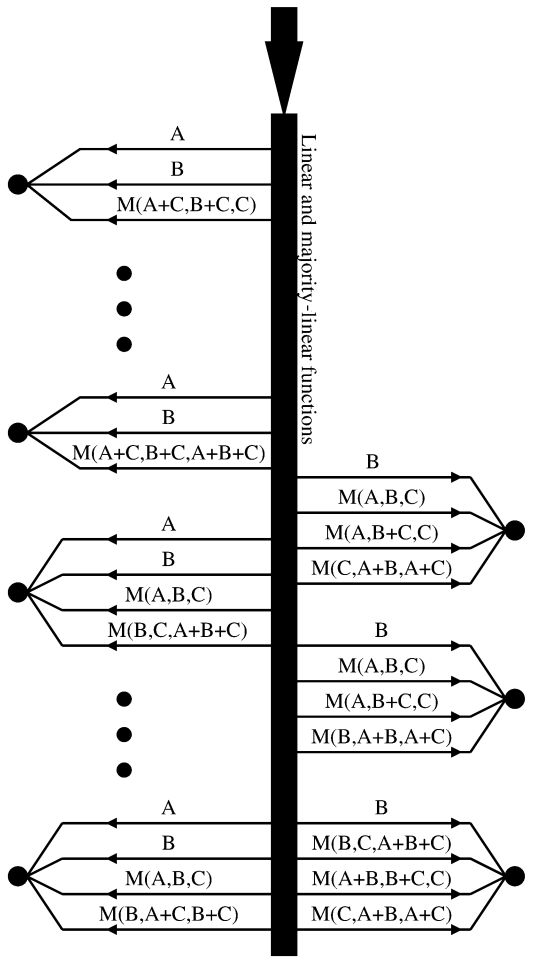


Fig. 5. This circuit includes various three-input and four-input combinations of linear and majority-linear functions, for each complete set $\{A, B, C\}$ of linear functions.

i) If $\mathrm{M}(A, B, C) \rightsquigarrow \{B, C\}$, then

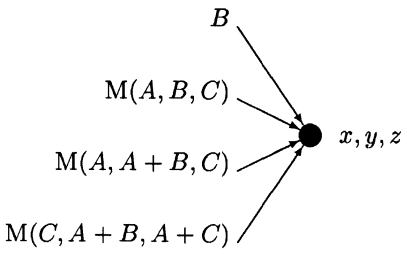
$$\mathrm{M}(A, D, B+C+D) \rightsquigarrow \{D, B+C+D\}$$

for all linear D such that $\{A, D, B+C+D\}$ is complete.

By Lemma IV.4b), either there exists a complete set of linear functions $\{A, B, C\}$ such that $\mathrm{M}(A, B, C) \rightsquigarrow A$, or else $\mathrm{M}(A, B, C) \rightsquigarrow A+B+C$ for every complete set of linear functions $\{A, B, C\}$.

First let us assume (in Cases 1 and 2 below) that $\{A, B, C\}$ is a complete set of linear functions such that $\mathrm{M}(A, B, C) \rightsquigarrow A$. Then $\mathrm{M}(A, A+B, C) \rightsquigarrow \{A, C\}$ by Lemma IV.4f). We show that this leads to a contradiction. Then we handle the remaining possibility in Case 3, below.

Case 1: $\mathrm{M}(A, B, C) \rightsquigarrow A$ and $\mathrm{M}(A, A+B, C) \rightsquigarrow A$: Lemma IV.2c) (taking $a = A$, $b = B$, $c = C$) implies that $n^{(4)}(A, B, C)$ is a destination node of $\mathcal{N}_1$, as shown below.



Then $\mathrm{M}(A, A+B, C) \rightsquigarrow A$ and Lemma IV.4g) (taking $D = A+C$) give

$$\mathrm{M}(C, A+C, A+B) \rightsquigarrow \{A+C, B\}.$$

Lemma IV.4f) and $\mathrm{M}(A, B, C) \rightsquigarrow A$ imply

$$\mathrm{M}(A, B, A+C) \rightsquigarrow \{A, B\}.$$

Lemma IV.4h) and $\mathrm{M}(A, A+B, C) \rightsquigarrow A$ imply

$$\mathrm{M}(A, B, A+C) \rightsquigarrow \{A, B+C\}.$$

Thus, $\mathrm{M}(A, B, A+C) \rightsquigarrow A$.

Lemma IV.4f) and $\mathrm{M}(A, A+B, C) \rightsquigarrow A$ imply

$$\mathrm{M}(A, A+B, A+C) \rightsquigarrow \{A, A+B\}.$$

Lemma IV.4f) and $\mathrm{M}(A, B, A+C) \rightsquigarrow A$ imply

$$\mathrm{M}(A, A+B, A+C) \rightsquigarrow \{A, A+C\}.$$

Thus,

$$\mathrm{M}(A, A+B, A+C) \rightsquigarrow \{A, A+B\} \cap \{A, A+C\} = \{A\}.$$

Lemma IV.4g) and $\mathrm{M}(A, A+B, A+C) \rightsquigarrow A$ imply

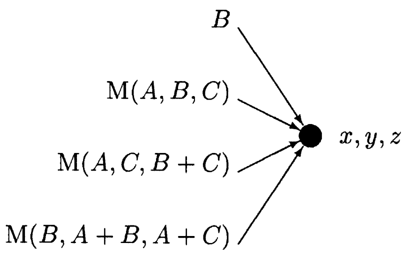
$$\mathrm{M}(C, A+C, A+B) \rightsquigarrow \{B, C\}.$$

Thus, we conclude that

$$\mathrm{M}(A+C, A+B, C) \rightsquigarrow \{B, C\} \cap \{B, A+C\} = \{B\}.$$

Together, these imply that the valuations of the four inputs to the destination node $n^{(4)}(A, B, C)$ above lie in the set $\{A, B\}$, which is not complete, contradicting the solvability of $\mathcal{N}_1$. So Case 1 is impossible.

Case 2: $\mathrm{M}(A, B, C) \rightsquigarrow A$ and $\mathrm{M}(A, A+B, C) \rightsquigarrow C$: Lemma IV.2d) (taking $a = A$, $b = B$, $c = C$) implies that $n^{(5)}(A, B, C)$ is a destination node of $\mathcal{N}_1$, as shown below.



Lemma IV.4c) and $\mathrm{M}(A, A+B, C) \rightsquigarrow C$ imply

$$\mathrm{M}(A, A+B, B+C) \rightsquigarrow B+C$$

which with Lemma IV.4i) implies

$$\mathrm{M}(B, A+B, A+C) \rightsquigarrow \{B, A+C\}.$$

Lemma IV.4e) and $\mathrm{M}(A, B, C) \rightsquigarrow A$ imply

$$\mathrm{M}(A, B, A+B+C) \rightsquigarrow B$$

which with Lemma IV.4h) implies

$$\mathrm{M}(B, A+B, A+C) \rightsquigarrow \{B, C\}.$$

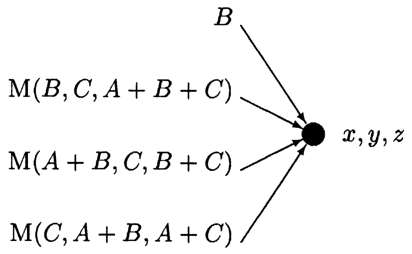
Thus, $\mathrm{M}(B, A+B, A+C) \rightsquigarrow B$. Also, Lemma IV.4e) and $\mathrm{M}(A, A+B, C) \rightsquigarrow C$ imply

$$\mathrm{M}(A, B+C, C) \rightsquigarrow A.$$

Together, these imply that the valuations of the four inputs to the destination node $n^{(5)}(A, B, C)$ above lie in the set $\{A, B\}$, which is not complete, contradicting the solvability of $\mathcal{N}_1$. So Case 2 is impossible.

Case 3: $\mathrm{M}(A, B, C) \rightsquigarrow A + B + C$ for every complete set of linear functions $\{A, B, C\}$:

Lemma IV.2e) (taking $a = A$, $b = B$, $c = C$) implies that $n^{(6)}(A, B, C)$ is a destination node of $\mathcal{N}_1$, as shown below.


B
$\mathrm{M}(B, C, A+B+C)$
$\mathrm{M}(A+B, C, B+C)$
$\mathrm{M}(C, A+B, A+C)$
x, y, z

Since $\mathrm{M}(A, B, C) \rightsquigarrow A + B + C$ holds for all complete sets of linear functions $\{A, B, C\}$, we can apply it to the three majority vote functions in the node above to get

$$\mathrm{M}(B, C, A+B+C) \rightsquigarrow A$$
$$\mathrm{M}(A+B, C, B+C) \rightsquigarrow A$$

and

$$\mathrm{M}(C, A+B, A+C) \rightsquigarrow B.$$

Together, these imply that the valuations of the four inputs to the destination node $n^{(6)}(A, B, C)$ above lie in the set $\{A, B\}$, which is not complete, contradicting the solvability of $\mathcal{N}_1$. So Case 3 is impossible.

Since all three cases are impossible, there is no linear solution to the given network, even though it is solvable. □

V. Solvability for Different Alphabet Sizes

In this section, we examine the role of alphabet size in the solvability of multicast networks.

If a network is solvable for a particular alphabet, then it is clearly solvable, using Cartesian products, for any alphabet of cardinality $|\mathcal{A}|^i$ and any integer $i \geq 1$. So, in this sense, solvability becomes somewhat "easier" as the alphabet size grows. In fact, the Li–Yeung–Cai linearity result in [11] guarantees not only a linear solution to a solvable network for large enough cardinality, but also a linear solution for any finite field larger than some specific size. This tends to add support to the notion that larger alphabets make solvability easier. One might be tempted to conjecture that if a network is solvable for a certain alphabet, it must be solvable for every larger alphabet.

However, the main result in this section shows that it is possible for a multicast network to be solvable for a certain alphabet but not solvable for some larger alphabet.

Theorem V.2 given later in this section considers solutions to a network with arbitrary alphabet sizes. No specific algebraic structure (e.g., a ring or field) is imposed on the alphabet, and hence the result does not depend on the notion of linearity.

First, we need to present a lemma about latin squares. A *latin square* [13] of order n is an $n \times n$ square matrix, each row and column of which is a permutation of the integers $\{1, \ldots, n\}$. Two latin squares $\{a_{ij}\}$ and $\{b_{ij}\}$ are *orthogonal* if each ordered pair (a_{ij}, b_{ij}) is distinct, for all i and j.

In 1779, Euler conjectured that no pairs of orthogonal latin squares exist if the order of the matrices equals 2 modulo 4. It is easy to show that pairs of orthogonal latin squares of order two do not exist. In 1900, Tarry [18] proved that there do not exist pairs of orthogonal latin squares of order six. Tarry's proof involved an enormous exhaustive calculation by hand. A short, self-contained proof was given by Stinson [17] in 1984. However, Euler's conjecture was disproved in 1960, when Bose, Shrikhande, and Parker proved in a series of long and complicated papers [2]–[4] that pairs of orthogonal latin squares *always* exist if the order of the matrices is neither two nor six. Lie [12] gave a short and elegant proof of this result in 1982. These results are summarized in the following lemma.

Lemma V.1: Pairs of orthogonal latin squares exist if and only if the order of the matrices is neither 2 nor 6.

The following theorem follows immediately from Lemma V.3, which makes use of the multicast network $\mathcal{N}_2$ shown in Fig. 6.

Theorem V.2: A multicast network that has a solution for a given alphabet might not have a solution for all larger alphabets.

Lemma V.3: The multicast network $\mathcal{N}_2$ is solvable if and only if the alphabet size is neither 2 nor 6.

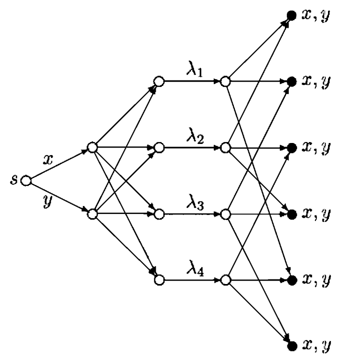


Fig. 6. Example of a multicast network $\mathcal{N}_2$ with messages x, y, source node s, and six destination nodes. The edge valuations λ_1, λ_2, λ_3, λ_4 are functions of x, y, and each of the six possible pairs of them constitutes the in-edges of a unique destination node.

Proof: The network $\mathcal{N}_2$ is solvable for a specific alphabet $\mathcal{A} = \{a_1, \dots a_{|\mathcal{A}|}\}$ if and only if each of the six pairs $\{\lambda_i, \lambda_j\}$, for $1 \leq i, j \leq 4$ and $i \neq j$, is complete (so that the pair $\{x, y\}$ can be recovered for each destination node). Let $\mathcal{S}_k$ be the $|\mathcal{A}| \times |\mathcal{A}|$ square matrix whose entry in the ith row and jth column is n if $\lambda_k(a_i, a_j) = a_n$. Let us now temporarily assume $\lambda_1 = x$ and $\lambda_2 = y$, i.e., the projection functions.

Clearly, $\{\lambda_1, \lambda_2\}$ is complete. The pair $\{\lambda_1, \lambda_3\}$ is complete if and only if for each $a_i \in \mathcal{A}$, no two elements of the ith row of $\mathcal{S}_3$ are the same (for otherwise y could not be uniquely recovered from x and λ_3), i.e., every row of $\mathcal{S}_3$ must contain each of the integers in $\{1, \dots, |\mathcal{A}|\}$ exactly once. Similarly, the pair $\{\lambda_2, \lambda_3\}$ is complete if and only if every column of $\mathcal{S}_3$ contains each of the integers in $\{1, \dots, |\mathcal{A}|\}$ exactly once. Thus, the pairs $\{\lambda_1, \lambda_3\}$ and $\{\lambda_2, \lambda_3\}$ are both complete if and only if $\mathcal{S}_3$ is a latin square of order $|\mathcal{A}|$. A similar argument shows that the pairs $\{\lambda_1, \lambda_4\}$ and $\{\lambda_2, \lambda_4\}$ are both complete if and only if $\mathcal{S}_4$ is a latin square of order $|\mathcal{A}|$. Now, the pair $\{\lambda_3, \lambda_4\}$ is complete if and only if for all $i, j \in \{1, \dots, |\mathcal{A}|\}$, the ordered pair of (i, j)th entries in $\mathcal{S}_3$ and $\mathcal{S}_4$ does not repeat at any other location in the two matrices (i.e., allowing unique recovery of i and j given the (i, j)th entries), if and only if each of the $|\mathcal{A}|^2$ pairs of integers from $\{1, \dots, |\mathcal{A}|\}$ appears exactly once in the matrices $\mathcal{S}_3$ and $\mathcal{S}_4$ at the same positions. Hence, all six pairs $\{\lambda_i, \lambda_j\}$ are complete if and only if $\mathcal{S}_3$ and $\mathcal{S}_4$ are orthogonal latin squares. So, by Lemma V.1, we conclude that $\mathcal{N}_2$ is solvable for all alphabets $\mathcal{A}$ such that $|\mathcal{A}| \notin \{2, 6\}$, and is not solvable for $|\mathcal{A}| \in \{2, 6\}$ under the assumptions $\lambda_1 = x$ and $\lambda_2 = y$.

Now let us relax the assumptions that $\lambda_1 = x$ and $\lambda_2 = y$ and suppose that $\mathcal{N}_2$ has a solution. Let $w = \lambda_1$ and $z = \lambda_2$. Then the pair $\{w, z\}$ is complete, so that x and y can each be recovered from w and z. Thus, since λ_3 and λ_4 are each functions of x and y, they are also functions of w and z. Formally, since the mapping $(\lambda_1, \lambda_2) : \mathcal{A}^2 \to \mathcal{A}^2$ is a bijection, it has an inverse h, and then λ_3 and λ_4 can be identified with $\hat{\lambda}_3 = \lambda_3 \circ h$ and $\hat{\lambda}_4 = \lambda_4 \circ h$, respectively. Matrices $\hat{\mathcal{S}}_3$ and $\hat{\mathcal{S}}_4$ can be defined for $\hat{\lambda}_3$ and $\hat{\lambda}_4$ analogously as before. Now the same argument as earlier implies that the six pairs

$$\{w, z\}, \{w, \hat{\lambda}_3\}, \{w, \hat{\lambda}_4\}, \{z, \hat{\lambda}_3\}, \{z, \hat{\lambda}_4\}, \{\hat{\lambda}_3, \hat{\lambda}_4\}$$

are each complete if and only if the matrices $\hat{\mathcal{S}}_3$ and $\hat{\mathcal{S}}_4$ are orthogonal latin squares. Thus, if $\mathcal{N}_2$ were solvable for $|\mathcal{A}| \in \{2, 6\}$ then there would exist a pair of orthogonal latin squares of order 2 or 6, contradicting their known nonexistence. $\square$

Note that, if $|\mathcal{A}|$ is odd and we view $\mathcal{A}$ as the ring of integers modulo $|\mathcal{A}|$, then $\mathcal{N}_2$ has a linear solution by choosing

$$\begin{aligned} \lambda_1 &= x \\ \lambda_2 &= y \\ \lambda_3 &= x + y \\ \lambda_4 &= x - y \end{aligned}$$

and then recovering x and y using

$$\begin{aligned} x &= \lambda_2 + \lambda_4 = \lambda_3 - \lambda_2 = (\lambda_3 + \lambda_4) \cdot 2^{-1} \\ x &= \lambda_2 + \lambda_4 = \lambda_3 - \lambda_2 = (\lambda_3 + \lambda_4) \cdot 2^{-1} \\ y &= \lambda_3 - \lambda_1 = \lambda_1 - \lambda_4 = (\lambda_3 - \lambda_4) \cdot 2^{-1}. \end{aligned}$$

This demonstrates linear solutions for more alphabets than those whose cardinalities are powers of primes.

Also, note that, if we allow coding over two units of time (i.e., two uses of the network), then a vector-linear solution exists, by choosing encoding operation vectors

$$\begin{aligned} \lambda_1 &= \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \\ \lambda_2 &= \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \\ \lambda_3 &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \\ \lambda_4 &= \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \end{aligned}$$

where the message vectors are

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}.$$

This vector-linear solution is valid for any alphabet size by viewing $\mathcal{A}$ as the ring of integers modulo $|\mathcal{A}|$ (i.e., $x_1, x_2, y_1, y_2 \in \mathcal{A}$). This is true even though $\mathcal{N}_2$ is not instantaneously solvable for $|\mathcal{A}| \in \{2, 6\}$. This multicast network vector-linear solution is also an example of what was conjectured in [14] to be true for more general networks.

The multicast network in Fig. 6 was one member in a family of networks used in [15] to show that the minimum alphabet size of multicast network solutions might have to be at least about the square root of the number of destination nodes. It was also used in independent work in [16] in examining codes over multiple time units, where a binary vector-linear solution was given. Our characterization of which alphabet sizes admit solutions to $\mathcal{N}_2$ gives some more insight about the role of alphabet sizes and solvability.

VI. Proofs of Lemmas

A. *Proof of Lemma II.1*

Consider the valuations induced by a nonhomogeneous solution. Let $e = (a, b)$ be any edge whose valuation is nonhomogeneous. Define $\hat{\phi}_e : \mathcal{A}^{|\mathcal{E}_i(a)|} \to \mathcal{A}$ by

$$\hat{\phi}_e(z) = \phi_e(z) - \phi_e(0)$$

and for each $w \in \mathcal{E}_o(b)$, define $\hat{\phi}_w : \mathcal{A}^{|\mathcal{E}_i(b)|} \to \mathcal{A}$ by

$$\hat{\phi}_w(z) = \phi_w(\hat{z})$$

where

$$z = (z_1, \dots, z_{|\mathcal{E}_i(b)|})$$
$$\hat{z} = (\hat{z}_1, \dots, \hat{z}_{|\mathcal{E}_i(b)|})$$

and

$$\hat{z}_i = \begin{cases} z_i + \phi_e(0), & \text{if } z_i = e \\ z_i, & \text{if } z_i \neq e. \end{cases}$$

In the network solution, replace the encoding operation ϕ_e by $\hat{\phi}_e$ and the encoding operations ϕ_w by the functions $\hat{\phi}_w$, for all $w \in \mathcal{E}_o(b)$. (If e is the in-edge to a destination node n, then perform the same procedure as above, but replace the decoding operations $\phi_{n,x}$ in an analogous manner.)

The replacement encoding operations (or decoding operations) at each iteration preserve the valuations except at the edge e. Thus, the new code is still a solution to the network. Repeat this procedure using the new codes until all edge valuations are homogeneous. The number of homogeneous edge valuations in the network increases by exactly one after each iteration, and therefore the procedure must terminate in a finite number of iterations.

Since each iteration either adds a constant to an existing encoding operation or subtracts a constant from the input of an existing encoding operation (and does the same for decoding operations), if the original nonhomogeneous solution is affine, then the resulting solution when the procedure terminates will be linear. $\square$

B. *Proof of Lemma IV.1*

Since the network has three source messages, each destination node must be able to recover all three edge valuations a, b, c in order to recover the messages. Suppose d is not the constant 0. Then d must depend on both a and b, for otherwise n_1 and n_2 would not be able to recover a and b. Thus, if d is a linear function of a and b then it must equal $a + b$. If d is not a linear function of a and b, since the network is homogeneous by assumption, we must have

$$d \in \{ab, ab + a, ab + b, ab + a + b\}.$$

The sets $\{a, ab\}$ and $\{a, ab+a\}$ are not complete, since b cannot be determined when $a = 0$. Hence,

$$d \notin \{ab, ab + a\}.$$

The same reasoning shows $\{b, ab + b\}$ is not complete, so $d \neq ab + b$. The set $\{a, a + b + ab\}$ is not complete, since b cannot be determined when $a = 1$. Therefore, $d \neq ab + a + b$. Thus, d cannot be a nonlinear function of a and b. $\square$

C. *Proof of Lemma IV.2*

a) The variable c can be recovered as $c = (a + c)(b + c) + \mathrm{M}(a, b, c)$, from which a and b can be recovered from $a+c$ and $b+c$. Thus, the set is complete.

b) If $a = b$ then $d = \mathrm{M}(a, d, a+b+d)$. If $a \neq b$ then $c = \mathrm{M}(a, b, c)$. Thus, the set is complete, since both $\{a, b, c\}$ and $\{a, b, d\}$ are complete.

c) If $b = 0$ then a is recoverable as $a = \mathrm{M}(a, a+b, c)$. So if $b = 0$ and $a = 0$ then $c = \mathrm{M}(c, a+b, a+c)$, and if $b = 0$ and $a = 1$ then $c = \mathrm{M}(a, b, c)$. This implies that the complete set $\{a, b, c\}$ is recoverable when $b = 0$. If $b = 1$ then c is recoverable as $c = \mathrm{M}(a, a+b, c)$. So if $b = 1$ and $c = 0$ then a is recoverable as $a = \mathrm{M}(a, b, c)$, and if $b = 1$ and $c = 1$ then a is recoverable as $a = 1+\mathrm{M}(c, a+b, a+c)$. This implies that the complete set $\{a, b, c\}$ is recoverable when $b = 1$.

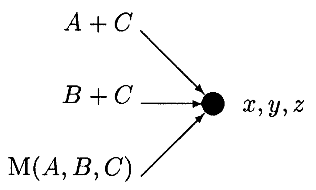
d) If $b = 0$ then c can be recovered as $c = \mathrm{M}(a, c, b+c)$. If $b = 0$ and $c = 0$ then a can be recovered as $a = \mathrm{M}(b, a+b, a+c)$. If $b = 0$ and $c = 1$ then a can be recovered as $a = \mathrm{M}(a, b, c)$. If $b = 1$ then a can be recovered as $a = \mathrm{M}(a, c, b+c)$. If $b = 1$ and $a = 0$ then c can be recovered as $c = \mathrm{M}(a, b, c)$. If $b = 1$ and $a = 1$ then c can be recovered as $c = 1+\mathrm{M}(b, a+b, a+c)$.

e) If $b = 0$ then c can be recovered as $c = \mathrm{M}(a+b, c, b+c)$. If $b = 0$ and $c = 0$ then a can be recovered as $a = \mathrm{M}(c, a+b, a+c)$. If $b = 0$ and $c = 1$ then a can be recovered as $a = 1+\mathrm{M}(b, c, a+b+c)$. If $b = 1$ then a can be recovered as $a = 1+\mathrm{M}(a+b, c, b+c)$. If $b = 1$ and $a = 0$ then c can be recovered as $c = \mathrm{M}(c, a+b, a+c)$. If $b = 1$ and $a = 1$ then c can be recovered as $c = \mathrm{M}(b, c, a+b+c)$. $\square$

D. *Proof of Lemma IV.4*

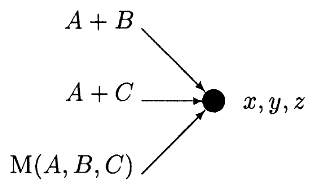
Lemma IV.4a) is used, without explicit mention, throughout the proofs of the subsequent parts of Lemma IV.4.

a) Since $C = (A + C)(B + C) + \mathrm{M}(A, B, C)$, the set $\{A + C, B + C, \mathrm{M}(A, B, C)\}$ is complete and, therefore, $n^{(2)}(A + C, B + C, \mathrm{M}(A, B, C))$ is a destination node in $\mathcal{N}_1$ as shown below.



The lemma then follows from that fact that $\{A+C, B+C\}$ is not a complete set.

b) Lemma IV.2a) (taking $a = A, b = B, c = C$) implies that $n^{(2)}(A+B, A+C, \mathrm{M}(A, B, C))$ is a destination node of $\mathcal{N}_1$, as shown below.



If $\mathrm{M}(A, B, C) \rightsquigarrow \{A + B, A + C\}$ then we do not have a solution to $\mathcal{N}_1$ since the set $\{A+B, A+C\}$ is not complete. Similarly, if $\mathrm{M}(A, B, C) \rightsquigarrow B + C$ then we do not have a solution to $\mathcal{N}_1$ since the set $\{A + B, B + C, A + C\}$ is not complete.

c) Lemma IV.2b) (taking $a = B$, $b = A$, $c = C$, $d = C$) implies that

$$n^{(3)}(A, B, \mathrm{M}(A, B, C), \mathrm{M}(B, C, A{+}B{+}C))$$

is a destination node of $\mathcal{N}_1$, as shown below.

$$\begin{tikzcd}
A \arrow[ddr] & \\
B \arrow[dr] & \\
& \bullet \ x, y, z \\
\mathrm{M}(A,B,C) \arrow[ur] & \\
\mathrm{M}(B,C,A+B+C) \arrow[uur] &
\end{tikzcd}$$

Lemma IV.4 b) implies

$$\mathrm{M}(B, C, A + B + C) \rightsquigarrow \{A, B, C, A + B + C\}.$$

The first two possibilities would result in the set of inputs $\{A, B\}$ which is not complete. Thus,

$$\mathrm{M}(B, C, A{+}B{+}C) \rightsquigarrow \{C, A{+}B{+}C\}.$$

Lemma IV.2b) (taking $a{=}C$, $b{=}A$, $c{=}B$, $d{=}A{+}B{+}C$) implies that

$$n^{(3)}(A, C, \mathrm{M}(A, B, C), \mathrm{M}(B, C, A + B + C))$$

is a destination node $\mathcal{N}_1$, as shown below.

$$\begin{tikzcd}
A \arrow[ddr] & \\
C \arrow[dr] & \\
& \bullet \ x, y, z \\
\mathrm{M}(A,B,C) \arrow[ur] & \\
\mathrm{M}(B,C,A+B+C) \arrow[uur] &
\end{tikzcd}$$

Also, $\mathrm{M}(B, C, A + B + C) \rightsquigarrow C$ is not possible since it would result in the set of inputs $\{A, C\}$ which is not complete. Thus, we must have $\mathrm{M}(B, C, A + B + C) \rightsquigarrow A + B + C$.

d) Lemma IV.2b) (taking $a{=}B$, $b{=}A{+}B{+}C$, $c{=}C$, $d{=}C$) implies that

$$n^{(3)}(B, A{+}B{+}C, \mathrm{M}(A, B, C), \mathrm{M}(B, C, A{+}B{+}C))$$

is a destination node of $\mathcal{N}_1$, as shown below.

$$\begin{tikzcd}
B \arrow[ddr] & \\
A+B+C \arrow[dr] & \\
& \bullet \ x, y, z \\
\mathrm{M}(A,B,C) \arrow[ur] & \\
\mathrm{M}(B,C,A+B+C) \arrow[uur] &
\end{tikzcd}$$

Lemma IV.4b) implies

$$\mathrm{M}(B, C, A + B + C) \rightsquigarrow \{A, B, C, A + B + C\}.$$

The second and fourth possibilities would result in the set of inputs $\{B, A + B + C\}$ which is not complete. Thus, $\mathrm{M}(B, C, A + B + C) \rightsquigarrow \{A, C\}$. Lemma IV.2b) (taking $a = C$, $b = A + B + C$, $c = B$, $d = B$) implies that

$$n^{(3)}(C, A + B + C, \mathrm{M}(A, B, C), \mathrm{M}(B, C, A + B + C))$$

is a destination node of $\mathcal{N}_1$, as shown below.

$$\begin{tikzcd}
C \arrow[ddr] & \\
A+B+C \arrow[dr] & \\
& \bullet \ x, y, z \\
\mathrm{M}(A,B,C) \arrow[ur] & \\
\mathrm{M}(B,C,A+B+C) \arrow[uur] &
\end{tikzcd}$$

Also, $\mathrm{M}(B, C, A{+}B{+}C) \rightsquigarrow C$ is not possible since it would result in the set of inputs $\{C, A{+}B{+}C\}$ which is not complete. Thus, we must have

$$\mathrm{M}(B, C, A{+}B{+}C) \rightsquigarrow A.$$

e) Lemma IV.2b) (taking $a = A$, $b = A{+}B{+}C$, $c = B$, $d = B$) implies that

$$n^{(3)}(A, A{+}B{+}C, \mathrm{M}(A, B, C), \mathrm{M}(A, B, A{+}B{+}C))$$

is a destination node of $\mathcal{N}_1$, as shown below.

$$\begin{tikzcd}
A \arrow[ddr] & \\
A+B+C \arrow[dr] & \\
& \bullet \ x, y, z \\
\mathrm{M}(A,B,C) \arrow[ur] & \\
\mathrm{M}(A,B,A+B+C) \arrow[uur] &
\end{tikzcd}$$

Lemma IV.4b) implies

$$\mathrm{M}(A, B, A{+}B{+}C) \rightsquigarrow \{A, B, C, A{+}B{+}C\}.$$

The first and fourth possibilities would result in the set of inputs $\{A, A{+}B{+}C\}$ which is not complete. Thus,

$$\mathrm{M}(A, B, A{+}B{+}C) \rightsquigarrow \{B, C\}.$$

Lemma IV.2b) (taking $a = C$, $b = A$, $c = B$, $d = B$) implies that

$$n^{(3)}(A, C, \mathrm{M}(A, B, C), \mathrm{M}(B, C, A + B + C))$$

is a destination node of $\mathcal{N}_1$, as shown below.

$$\begin{tikzcd}
A \arrow[dddr] & \\
C \arrow[ddr] & \\
\mathrm{M}(A,B,C) \arrow[dr] & \\
\mathrm{M}(B,C,A+B+C) \arrow[r] & \bullet\; x,y,z
\end{tikzcd}$$

$\mathrm{M}(A,B,A+B+C) \rightsquigarrow C$ is not possible since it would result in the set of inputs $\{A,C\}$ which is not complete. Thus, we must have $\mathrm{M}(A,B,A{+}B{+}C) \rightsquigarrow B$.

f) Lemma IV.4 b) implies

$$\mathrm{M}(A,B,D) \rightsquigarrow \{A,B,D,A{+}B{+}D\}.$$

We will rule out the third and fourth possibilities. Lemma IV.2b) (taking $a=A, b=B, c=C, d=D$) implies that

$$n^{(3)}(A,B,\mathrm{M}(A,B,C),\mathrm{M}(A,D,A{+}B{+}D))$$

is a destination node of $\mathcal{N}_1$, as shown below.

$$\begin{tikzcd}
A \arrow[dddr] & \\
B \arrow[ddr] & \\
\mathrm{M}(A,B,C) \arrow[dr] & \\
\mathrm{M}(A,D,A+B+D) \arrow[r] & \bullet\; x,y,z
\end{tikzcd}$$

It is not possible for $\mathrm{M}(A,B,D) \rightsquigarrow D$, for otherwise $\mathrm{M}(A,D,A+B+D) \rightsquigarrow A$, by Lemma IV.4e), and then the input set would be $\{A,B\}$, which is not complete. Lemma IV.2b) (taking $a=B, b=A, c=C, d=D$) implies that

$$n^{(3)}(A,B,\mathrm{M}(A,B,C),\mathrm{M}(B,D,A+B+D))$$

is a destination node of $\mathcal{N}_1$, as shown below.

$$\begin{tikzcd}
A \arrow[dddr] & \\
B \arrow[ddr] & \\
\mathrm{M}(A,B,C) \arrow[dr] & \\
\mathrm{M}(B,D,A+B+D) \arrow[r] & \bullet\; x,y,z
\end{tikzcd}$$

It is not possible for $\mathrm{M}(A,B,D) \rightsquigarrow A{+}B{+}D$, for otherwise $\mathrm{M}(B,D,A{+}B{+}D) \rightsquigarrow A$, by Lemma IV.4d), and then the input set would be $\{A,B\}$, which is not complete.

g) Lemma IV.4b) implies

$$\mathrm{M}(A,B,D) \rightsquigarrow \{A,B,D,A{+}B{+}D\}.$$

If $\mathrm{M}(A,B,D) \rightsquigarrow \{A,B\}$ then Lemma IV.4f) implies that $\mathrm{M}(A,B,C) \rightsquigarrow \{A,B\}$, contradicting the supposition that

$$\mathrm{M}(A,B,C) \rightsquigarrow \{C,A{+}B{+}C\}.$$

h) Lemma IV.4b) implies

$$\mathrm{M}(A,D,B{+}C{+}D) \rightsquigarrow \{A,D,A{+}B{+}C,B{+}C{+}D\}.$$

We will rule out the second and fourth possibilities. Lemma IV.2b) (taking $a=A$, $b=A+B+C$, $c=B{+}C{+}D$, $d=B$) implies that

$$n^{(3)}(A,A{+}B{+}C,\mathrm{M}(A,B,C),\\ \mathrm{M}(A,A{+}B{+}C,B{+}C{+}D))$$

is a destination node of $\mathcal{N}_1$, as shown below.

$$\begin{tikzcd}
A \arrow[dddr] & \\
A+B+C \arrow[ddr] & \\
\mathrm{M}(A,B,C) \arrow[dr] & \\
\mathrm{M}(A,A+B+C,B+C+D) \arrow[r] & \bullet\; x,y,z
\end{tikzcd}$$

Since $\mathrm{M}(A,B,C) \rightsquigarrow \{A,A{+}B{+}C\}$, it is not possible for

$$\mathrm{M}(A,D,B{+}C{+}D) \rightsquigarrow D$$

for, otherwise,

$$\mathrm{M}(A,A{+}B{+}C,B{+}C{+}D) \rightsquigarrow A{+}B{+}C$$

by Lemma IV.4c), and then the input set would be $\{A,A{+}B{+}C\}$, which is not complete. Nearly the same argument shows that

$$n^{(3)}(A,A{+}B{+}C,\mathrm{M}(A,B,C),\mathrm{M}(A,A{+}B{+}C,D))$$

is a destination node of $\mathcal{N}_1$, as shown below

$$\begin{tikzcd}
A \arrow[dddr] & \\
A+B+C \arrow[ddr] & \\
\mathrm{M}(A,B,C) \arrow[dr] & \\
\mathrm{M}(A,A+B+C,D) \arrow[r] & \bullet\; x,y,z
\end{tikzcd}$$

and that it is not possible for $\mathrm{M}(A,B+C+D,D) \rightsquigarrow B+C+D$.

i) By Lemma IV.4b)

$$\mathrm{M}(A, D, B+C+D) \rightsquigarrow \{A, D, A+B+C, B+C+D\}.$$

If $\mathrm{M}(A, D, B+C+D) \rightsquigarrow \{A, A+B+C\}$ then

$$\mathrm{M}(A, B, C) \rightsquigarrow \{A, A+B+C\}$$

by Lemma IV.4h), a contradiction. □

Acknowledgment

The authors would like to thank Søren Riis for providing a preprint of his manuscript [16], and Raymond Yeung for some helpful pointers.

References

[1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inform. Theory*, vol. 46, pp. 1204–1216, July 2000.

[2] R. C. Bose and S. S. Shrikhande, "On the falsity of Euler's conjecture about the nonexistence of two orthogonal latin squares of order $4t+2$," *Proc. Nat. Acad. Sci.*, vol. 45, pp. 734–737, 1959.

[3] ——, "On the construction of sets of mutually orthogonal latin squares and falsity of a conjecture of Euler," *Trans. Amer. Math. Soc.*, vol. 95, pp. 191–209, 1960.

[4] R. C. Bose, S. S. Shrikhande, and E. T. Parker, "Further results on the construction of mutually orthogonal latin squares and the falsity of Euler's conjecture," *Canad. J. Math.*, vol. 12, pp. 189–203, 1960.

[5] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Proc. IEEE Allerton Conf. Communication, Control, and Computing*, Monticello, IL, Oct. 2003.

[6] M. Feder, D. Ron, and A. Tavory, "Bounds on linear codes for network multicast," in *Electronic Colloquium on Computational Complexity (ECCC)*, 2003, Rep. 33, pp. 1–9.

[7] T. Ho, R. Koetter, M. Médard, D. Karger, and M. Effros, "The benefits of coding over routing in a randomized setting," in *Proc. IEEE Int. Symp. Information Theory*, Yokohama, Japan, June 2003, p. 442.

[8] T. Ho, D. Karger, M. Médard, and R. Koetter, "Network coding from a network flow perspective," in *Proc. IEEE Int. Symp. Information Theory*, Yokohama, Japan, June 2003, p. 441.

[9] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen, "Polynomial time algorithms for multicast network code construction," *IEEE Trans. Inform. Theory*, submitted for publication.

[10] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Trans. Networking*, vol. 11, pp. 782–795, Oct. 2003.

[11] S.-Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. Inform. Theory*, vol. 49, pp. 371–381, Feb. 2003.

[12] Z. Lie, "A short disproof of Euler's conjecture concerning orthogonal latin squares," *Ars Combin.*, vol. 14, pp. 47–55, 1982.

[13] J. H. van Lint and R. M. Wilson, *A Course in Combinatorics*. Cambridge, U.K.: Cambridge Univ. Press, 1992.

[14] M. Médard, M. Effros, T. Ho, and D. Karger, "On coding for non-multicast networks," in *Proc. 41st Annu. Allerton Conf. Communication Control and Computing*, Monticello, IL, Oct. 2003.

[15] A. Rasala Lehman and E. Lehman, "Complexity classification of network information flow problems," in *Proc. 41st Annu. Allerton Conf. Communication Control and Computing*, Monticello, IL, Oct. 2003.

[16] S. Riis, "Linear versus non-linear Boolean functions in network flow," in *Proc. 38th Annu. Conf. Information Sciences and Systems (CISS)*, Princeton Univ., Princeton, NJ, Mar. 2004, pp. 263–268. Paper 603.

[17] D. R. Stinson, "A short proof of the nonexistence of a pair of orthogonal latin squares of order six," *J. Combin. Theory*, ser. A, vol. 36, pp. 373–376, 1984.

[18] G. Tarry, "Le problème des 36 officiers," *C. R. Assoc. France Av. Sci.*, pt. 2, vol. 29, pp. 170–203, 1900.

[19] R. W. Yeung, *A First Course in Information Theory*. Norwell, MA: Kluwer, 2002.