Sum-networks from undirected graphs: construction and capacity analysis

Ardhendu Tripathy and Aditya Ramamoorthy

Department of Electrical and Computer Engineering, Iowa State University, Ames, Iowa 50011
Email: {ardhendu,adityar}@iastate.edu

Abstract—We consider a directed acyclic network with multiple sources and multiple terminals where each terminal is interested in decoding the sum of independent sources generated at the source nodes. We describe a procedure whereby a simple undirected graph can be used to construct such a *sum-network* and demonstrate an upper bound on its computation rate. Furthermore, we show sufficient conditions for the construction of a linear network code that achieves this upper bound. Our procedure allows us to construct sum-networks that have any arbitrary computation rate $\frac{p}{q}$ (where p, q are non-negative integers). Our work significantly generalizes a previous approach for constructing sum-networks with arbitrary capacities. Specifically, we answer an open question in prior work by demonstrating sum-networks with significantly fewer number of sources and terminals.

I. INTRODUCTION

Function computation using network coding is an area that has received significant attention in recent years (see for instance [1], [2], [3], [4]). Broadly speaking, one considers directed acyclic networks with error free links, a set of source nodes that generate independent information and terminal nodes that demand a certain function of the source values. The topology of the network is allowed to be arbitrary. The most general formulation is evidently quite complex to study as depending on the function the demands can be arbitrarily complex and contain for instance multiple unicast as a special case, a problem which is well recognized to be hard (see for instance the discussion in [5], [6]). Accordingly, several simplified settings have been considered in the literature. The work of [1], [2] considers general functions, but networks with only one terminal. A different line of work considers networks with multiple terminals that each need a simple function such as the sum [3], [4]. Significant prior work [7], [8], [9] considers information theoretic issues in function computation where the sources are dependent but the network structures are simple in the sense that there are direct links between sources and the terminals.

In this work, we consider the problem of network coding for sum-networks. Specifically, the problem is one of multicasting the finite field sum of source values that are available at a set of source nodes to a designated set of terminals over a directed acyclic network, i.e., a sum-network. The topology of the graph denoting the network can be completely arbitrary under the trivial restriction that there exists a path between any terminal and each source node. We assume that the sources are independent and that the network links are delay and error-free

but have finite capacity.

This problem was first considered in the work of [10], where the class of networks with unit-entropy sources, unit-capacity edges and either two sources or two terminals was considered. For this class of networks it was demonstrated that as long as each source is connected to each terminals, computation of the sum was possible. In contrast, it was shown [3] that there exist networks with three sources and three terminals where sum computation is impossible even though each source terminal pair is connected. Conditions on sum computation for networks with three sources and three terminals have also been investigated in prior work [3], [11]. More generally, one can define the notion of computation rate [1]. Informally, a network code for a sum-network is said to have rate r/l if in l time slots, one can multicast the sum r times to all the terminals. A network is called solvable if it has a (r, r) code and not solvable otherwise. The problem of multicasting the sum can be shown to be equivalent to the problem of multiple unicast. Specifically, [4] shows that there is a solvably equivalent sum-network for any multiple-unicast network. Thus characterizing the solvability of sum networks and identifying the network resources required in order to ensure solvability of a sum-network assumes importance.

There are several open problems for sum-networks where the number of sources and terminals is at least three. For instance, nontrivial sufficient conditions on the network resources that allow for sum computation are not known. However, recently certain impossibility results have been obtained. Reference [12] shows that for any integer $k \geq 2$, there exists a sum-network with three sources and four or more terminals (and also a sum-network with three terminals and four or more sources) with coding capacity $\frac{k}{k+1}$ for integers $k \geq 0$. Reference [13] is most closely related to our work. Given a ratio p/q it constructs a sum-network that has capacity equal to p/q. In this work, we propose a construction of sum networks that significantly generalizes the work of [13] and answer some of its open questions.

A. Main Contributions

• Asssuming coprime p and q, the work of [13], constructs a sum-network that has $2q - 1 + \binom{2q-1}{2}$ sources and $2q - 1 + \binom{2q-1}{2} + 1$ terminals, which can be significantly large if q is large. An open question posed in [13] is whether sum-networks of smaller size exist that have a capacity p/q. In this work, we answer this in the affirmative. Specifically, we construct a

large family of sum-networks that can be significantly smaller for several values of p/q and recover their result as a special case. We note that examples of small sum-networks with capacity strictly smaller than one are useful in investigating sufficiency conditions for general networks. For example, [3] demonstrates an instance of a network with three sources and three terminals where unit connectivity between each source terminal pair does not suffice, implying that a sufficiency condition for such 3-source, 3-terminal networks that only looks at minimum connectivity between source terminal pairs needs to consider networks where the minimum cut between each source terminal pair is at least two.

• Our proof that the constructed sum-network has the appropriate capacity value is simpler than the proof of [13].

This paper is organized as follows. The problem is formally posed in Section II, our construction is explained in Section III and a comparison with existing results appears in Section IV. We conclude the paper with a discussion about future work in Section V.

II. PROBLEM FORMULATION

We consider communication over a directed acyclic graph (DAG) $G = (V, E)$ where V is the set of nodes and $E \in V \times V \times \mathbf{Z}_+$ are the edges denoting the delay-free communication links between them. Subset $S \subset V$ denotes the source nodes and $T \subset V$ denotes the terminal nodes. The source nodes have no incoming edges and the terminal nodes have no outgoing edges. Each source node $s_i \in S$ generates an independent random process X_i, such that the sequence of random variables $X_{i1}, X_{i2}, \ldots$ indexed by time are i.i.d. and each X_{ij} takes values that are uniformly distributed over a finite alphabet $\mathcal{A}$ that is assumed to be a finite field such that $|\mathcal{A}| = q$. Each edge is of unit capacity and can transmit one symbol from $\mathcal{A}$ per unit time. Our model allows for multiple edges between nodes. In this case the edges are given an additional index. For instance if there are two edges between nodes u and v, these will be represented as $(u, v, 1)$ and $(u, v, 2)$. The capacity of the edge (u, v) is defined as the number of edges between u and v.

We use the notation $\text{In}(v)$ to represent the set of incoming edges at node $v \in V$. We will also work with undirected graphs in this paper. If v is a node in an undirected graph, then $\text{In}(v)$ will represent the edges incident on v.

An network code is an assignment of edge functions to each edge in E and a decoding function to each terminal in T. The edge function for an edge connected to a source, depends only the source values. Likewise an edge function for an edge that is not connected to a sources depends on the values received on its incoming edges and the decoding function for a terminal depends only on its incoming edges. We let the source messages be vectors of length r and the edge functions to be vectors of length l. The decoding functions should be such that each terminal recovers the sum of all the source message vectors. The domain and range of the encoding functions can be summarized as follows.

• Edge function for edge e.

$$\phi_e : \mathcal{A}^r \to \mathcal{A}^l \quad \text{if tail}(e) \in S,$$
$$\phi_e : \mathcal{A}^{l|\text{In(tail}(e))|} \to \mathcal{A}^l \quad \text{if tail}(e) \notin S.$$

• Decoding function for the terminal $t_i \in T$.

$$\psi_{t_i} : \mathcal{A}^{l|\text{In}(t_i)|} \to \mathcal{A}^r$$

A network code is a linear network code if all the edge and decoding functions are linear. For the sum-networks that we consider, a (r, l) fractional network code solution over $\mathcal{A}$ is such that the sum of r source symbols (over the finite field) can be communicated to all the terminals in l units of time. The rate of this network code is defined to be r/l. A network is said to be solvable if it has a (r, r) network coding solution for some $r \geq 1$. A network is said to have a scalar solution if it has a $(1, 1)$ solution. The supremum of all achievable rates is called the capacity of the network.

III. CONSTRUCTION AND CAPACITY OF A FAMILY OF SUM NETWORKS

In this section we construct a family of sum-networks which are not solvable even though each source terminal pair is connected via at least one path. In fact, we will demonstrate that there exist families of sum-networks that are not solvable even though each source terminal pair has a minimum cut that is strictly larger than a fixed constant. The construction of the sum-network starts with a parallel set of $b \geq 2$ edges that we refer to as bottleneck edges in the subsequent discussion. Each bottleneck edge has a capacity α. There are α-capacity edges connecting carefully chosen subsets of the source nodes S to the tail of each bottleneck edge. Similarly, α-capacity edges connect the head of each bottleneck edge to a subset of terminal nodes T. The choice of the various subsets of the source nodes is made via the help of a undirected simple connected graph[1] $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ where $|\widetilde{V}| = b$. We arbitrarily number the vertices in $\widetilde{G}$ as $1, 2, \ldots, |\widetilde{V}|$. Then there are potentially the following three sets of sources in the sum-network G.

1) $S_1 = \{s_i : i = 1, 2, \ldots, |\widetilde{V}|\}$,
2) $S_2 = \{s_e : e \in \widetilde{E}\}$, and
3) $s_\star$.

Similarly, there are three types of terminals in G.

1) $T_1 = \{t_i : i = 1, 2, \ldots, |\widetilde{V}|\}$,
2) $T_2 = \{t_e : e \in \widetilde{E}\}$, and
3) $t_\star$,

so that $T = T_1 \cup T_2 \cup \{t^\star\}$. We propose two different constructions depending on whether we include $s_\star$ in the sum-network or not. In the constructions, we assume that $\widetilde{G}$ is connected and that $|\widetilde{E}| \geq |\widetilde{V}|$, i.e., it is not a tree.

---

[1] A graph is said to be simple if does not have self loops and multiple edges between a pair of nodes.
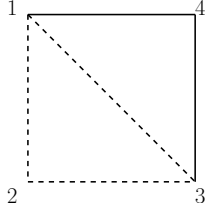
Fig. 1. $\widetilde{G}$ for Example 1. The dotted lines show the edge set $E_{\widetilde{Cyc}}$ for Construction 2.

A. Construction 1

For the first construction we do not include $s_\star$ in the network, thus $S = S_1 \cup S_2$. Let

$$X = \{X_1, X_2, \ldots, X_{|\widetilde{V}|}\} \cup \{X_e : e \in \widetilde{E}\} \quad (1)$$

be the set of independent random processes generated at the respective source nodes in G.

Let $\text{In}_{\widetilde{G}}(i)$ denote the edges $e \in \widetilde{E}$ that are incident to the vertex $i \in \widetilde{V}$ in the simple graph $\widetilde{G}$ and

$$A_i = \{X_i\} \cup \{X_e : e \in \text{In}_{\widetilde{G}}(i)\}. \quad (2)$$

We use A_i^c to denote $X \setminus A_i$.

Our sum-network G can be constructed as follows. We first include the source node set S, the terminal node set T and the bottleneck edges $B = \{e_1, e_2, \ldots, e_b\}$ where $b = |\widetilde{V}|$ and each edge is of capacity α. Next, we follow the construction algorithm below where edges (each of capacity α) are included.

Construction Algorithm 1

1) Edges from sources to bottlenecks.
   a) $(s_i, \text{tail}(e_j))$ if $X_i \in A_j$ for all $i, j \in \{1, 2, \ldots, b\}$,
   b) $(s_e, \text{tail}(e_j))$ if $X_e \in A_j$ for all $j \in \{1, 2, \ldots, b\}$.

2) Edges from bottlenecks to terminals.
   a) $(\text{head}(e_i), t_i)$ for all $i = \{1, 2, \ldots, b\}$,
   b) $(\text{head}(e_i), t_e)$ and $(\text{head}(e_j), t_e)$ for all $e = (i, j) \in \widetilde{E}$, and
   c) $(\text{head}(e_i), t_\star)$ for all $i \in \{1, 2, \ldots, b\}$.

3) Direct edges.
   a) (s_i, t_j) if $X_i \in A_j^c$ for all $i, j \in \{1, 2, \ldots, b\}$ and (s_e, t_j) if $X_e \in A_j^c$ for $j \in \{1, 2, \ldots, b\}$.
   b) For all $e = (i, j) \in \widetilde{E}$,
   (s_k, t_e) for $k \in \{1, 2, \ldots, b\}$ if $X_k \in A_i^c \cap A_j^c$ and
   $(s_{e'}, t_e)$ for $e' \in \widetilde{E}$ if $X_{e'} \in A_i^c \cap A_j^c$.

Next, we illustrate an example of the above construction.

Example 1: Let $\widetilde{G}$ be as shown in Figure 1. Then,

$$A_1 = \{X_1, X_{(1,2)}, X_{(1,3)}, X_{(1,4)}\}$$
$$A_2 = \{X_2, X_{(1,2)}, X_{(2,3)}\}$$
$$A_3 = \{X_3, X_{(1,3)}, X_{(2,3)}, X_{(3,4)}\}$$
$$A_4 = \{X_4, X_{(1,4)}, X_{(3,4)}\}$$

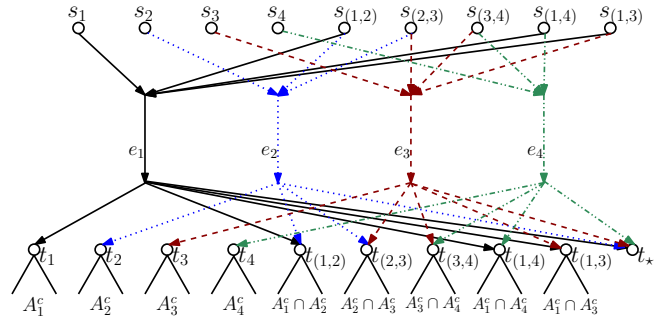The corresponding sum-network G is shown in Figure 2.

Fig. 2. Sum-network G constructed from $\widetilde{G}$ in Fig. 1 via Construction 1. Edges $e_i, i = 1, \ldots, 4$ represent the bottlenecks. The direct edges are specified by means of the set that appears below each terminal.

Fig. 3. Sum-network G constructed from $\widetilde{G}$ in Fig. 1 via Construction 2 (with source $s_\star$). Edges $e_i, i = 1, \ldots, 4$ represent the bottlenecks. The direct edges are specified by means of the set that appears below each terminal.
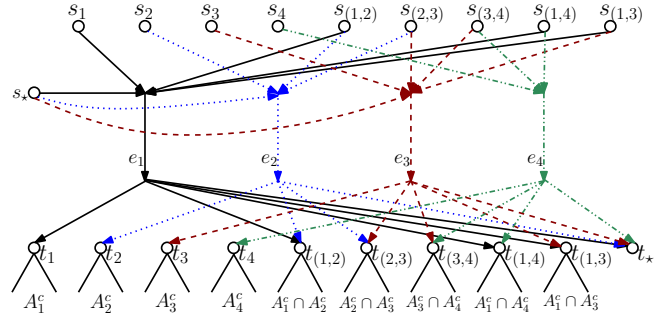
B. Construction 2

In the second construction, we include the source $s_\star$, i.e., $S = S_1 \cup S_2 \cup \{s_\star\}$, so that we have source $X_\star$ in addition to the sources listed in eq. (1). Recall that we assumed that $\widetilde{G}$ is not a tree. Let $\widetilde{Cyc} = (V_{\widetilde{Cyc}}, E_{\widetilde{Cyc}})$ be a subgraph of $\widetilde{G}$ corresponding to the _shortest cycle_ in $\widetilde{G}$; $\widetilde{Cyc}$ may not be unique. The following is a useful fact (proof appears in the Appendix).

Claim 1: Suppose that nodes $i, j \in V_{\widetilde{Cyc}}$. Then either $(i, j) \in E_{\widetilde{Cyc}}$ or $(i, j) \notin \widetilde{E}$.

The set of random processes A_i is defined as follows.

$$A_i = \begin{cases} \{X_i\} \cup \{X_e : e \in \text{In}_{\widetilde{G}}(i)\} \cup \{X_\star\} & \text{if } i \in V_{\widetilde{Cyc}}, \\ \{X_i\} \cup \{X_e : e \in \text{In}_{\widetilde{G}}(i)\} & \text{otherwise.} \end{cases}$$

Following the definition of the sets A_i, the steps outlined in Construction 1 can be followed to construct most of the sum-network G. Following this, the additional source $s_\star$ is connected to each e_i where $i \in V_{\widetilde{Cyc}}$. Each terminal in T that does not have a path from $s_\star$ to it, is provided a direct edge $s_\star$ to it. This concludes the construction of G. For an example, check Figure 3 for a sum-network constructed from Figure 1 with the choice of $\widetilde{Cyc}$ as indicated in caption.

C. Upper bound on the capacity of G

We derive an upper bound on the capacity of G assuming that we followed Construction 2 so that source $s_\star$ exists. The

corresponding upper bound for Construction 1 follows in a similar manner. Suppose that there exists a (r,l) fractional network code assignment $\phi_e, e \in E$ and decoding functions $\psi_t, t \in T$ so that all the terminals in T can recover the sum of sources denoted by $Z = \sum_{i\in\widetilde{V}} X_i + \sum_{e\in\widetilde{E}} X_e + X_\star$. As such we have three types of sources, corresponding to (i) the vertices of $\widetilde{G}$, (ii) the edges of $\widetilde{G}$ and (iii) the starred source. We use a generic index to refer to all these types of sources. For instance, $\sum_{\beta\in A_k} X_\beta$ would equal $\sum_{i:X_i\in A_k, i\in\widetilde{V}} X_i + \sum_{e:X_e\in A_k, e\in\widetilde{E}} X_e + \mathbb{1}_{\{k\in V_{\widetilde{Cyc}}\}} X_\star$, where $\mathbb{1}$ is the indicator function.

Lemma 1: For terminal $t \in T$, if edge (head$(e_k), t)$ exists, then terminal t can compute $\sum_{\beta\in A_k} X_\beta$ from the function value on e_k.

Proof: Consider terminal t_k for $k \in \widetilde{V}$. It is connected to head(e_k). By assumption, it is able to recover Z from the information transmitted on its incoming edges. In the discussion below we let $\phi_{e_k}(X)$ refer to the value that the network code assigns on edge e_k, where we emphasize that $\phi_{e_k}(X)$ only depends on sources in the set A_k. Suppose we cannot decode the sum $\sum_{\beta\in A_k} X_\beta$ from the value of $\phi_{e_k}(X)$. This implies that we can find two different instantiations of source symbols $\mathbf{x}$ and $\mathbf{x}'$ such that

- $\phi_{e_k}(\mathbf{x}) = \phi_{e_k}(\mathbf{x}')$ but $\sum_{\beta\in A_k} \mathbf{x}_\beta \neq \sum_{\beta\in A_k} \mathbf{x}'_\beta$, and
- $\mathbf{x}_\beta = \mathbf{x}'_\beta$ for $\beta \in A_k^c$.

Thus, even though $Z \neq Z'$, $\psi_{t_k}(\mathbf{x}) = \psi_{t_k}(\mathbf{x}')$ as $\phi_e(\mathbf{x}) = \phi_e(\mathbf{x}')$ for all $e \in \text{In}(t_k)$. Thus, the terminal t_k is unable to compute the sum which is a contradiction. ∎

In a similar manner the following lemma can be shown to hold.

Lemma 2: Any terminal t_e for $e = (i,j) \in \widetilde{E}$ is able to compute $\sum_{\beta\in A_i\cup A_j} X_\beta$ from the function values on edges e_i and e_j in G.

Theorem 1: The rate of a fractional network coding solution for the sum-network G constructed by the procedure above is upper bounded by $\frac{\alpha|\widetilde{V}|}{|\widetilde{E}|+|\widetilde{V}|+1}$.

Proof: We assume that there is a valid network code such that each terminal is able to compute the sum Z. Consider a terminal t_e where $e = (i,j) \in \widetilde{E}$. In this case t_e can compute (from Lemma 1 and 2), the following partial sums

- $\sum_{\beta\in A_i} X_\beta$.
- $\sum_{\beta\in A_j} X_\beta$.
- $\sum_{\beta\in A_i\cup A_j} X_\beta$.

Thus, it can compute $\sum_{\beta\in A_i} X_\beta + \sum_{\beta\in A_j} X_\beta - \sum_{\beta\in A_i\cup A_j} X_\beta$. Now, if e participates in the cycle $\widetilde{Cyc}$ we obtain $X_e + X_\star$, which can be observed by noting that $A_i \cap A_j = \{X_e, X_\star\}$. It can also be seen that if edge $(i,j) \notin E_{\widetilde{Cyc}}$ then at least one of i or j do not participate in $\widetilde{Cyc}$ (*cf.* Claim 1). In this case the previous operation would simply provide X_e. Terminal $t_\star$ is connected to all the bottleneck edges. Thus, it can obtain the following symbols.

$$Y_{1e} = \begin{cases} X_e + X_\star & \text{if } e \in E_{\widetilde{Cyc}} \\ X_e & \text{if } e \notin E_{\widetilde{Cyc}} \end{cases}$$

Following this step, $t_\star$ can compute $\sum_{\beta\in A_k} X_\beta - \sum_{e:X_e\in A_k} Y_{1e}$. Note that A_k contains only one source corresponding to the vertices, namely X_k. Now, if $k \notin V_{\widetilde{Cyc}}$, then it is evident that $Y_{1e} = X_e$ for all $e \in \text{In}_{\widetilde{G}}(k)$, so that $\sum_{\beta\in A_k} X_\beta - \sum_{e:X_e\in A_k, e\in\widetilde{E}} Y_{1e} = X_k$. Alternatively, if $k \in V_{\widetilde{Cyc}}$, then $\sum_{\beta\in A_k} X_\beta - \sum_{e:X_e\in A_k, e\in\widetilde{E}} Y_{1e} = X_k - X_\star$. Thus at the end of this operation, $t_\star$ has the following symbols.

$$Y_{2k} = \begin{cases} X_k - X_\star & \text{if } k \in V_{\widetilde{Cyc}} \\ X_k & \text{if } k \notin V_{\widetilde{Cyc}} \end{cases}$$

Note that a cycle has the same number of edges and nodes. Thus, $\sum_{k\in\widetilde{V}} Y_{2k} + \sum_{e\in\widetilde{E}} Y_{1e} = \sum_{i\in\widetilde{V}} X_i + \sum_{e\in\widetilde{E}} X_e$. However, terminal $t_\star$ already knows Z, thus it can compute $X_\star$ and consequently the value of all the sources. Therefore, under the assumption that all the terminals can compute the sum of sources, we can conclude that $t^\star$ is able to compute all the individual source symbols present in the sum-network.

It can be observed that the minimum cut between the set of all the sources and $t_\star$ is $\alpha|\widetilde{V}|$. Thus under a valid fractional (r,l)-network code we must have

$$\left(q^l\right)^{\alpha|\widetilde{V}|} \geq (q^r)^{|\widetilde{E}|+|\widetilde{V}|+1} \tag{3}$$

$$\implies \frac{r}{l} \leq \frac{\alpha|\widetilde{V}|}{|\widetilde{E}|+|\widetilde{V}|+1}. \tag{4}$$

This concludes the proof. ∎

Thus for the sum network constructed in Example 1 we must have, for a (r,l)-network coding solution

$$\frac{r}{l} \leq \frac{4\alpha}{4+5+1} = \frac{2\alpha}{5}. \tag{5}$$

Remark 1: It can be seen that following the line of proof above for the case of Construction 1 (without source $s_\star$), one arrives at a capacity upper bound of $\frac{\alpha|\widetilde{V}|}{|\widetilde{E}|+|\widetilde{V}|}$.

Remark 2: In Construction 2, we chose to connect the starred source only a carefully chosen subset of the bottleneck edges. If instead we had connected it to all the bottleneck edges (for instance), the starred terminal could only recover the source, under conditions on the characteristic of the field $\mathcal{A}$. Our choice of the subset of bottleneck edges avoids this dependence on the field characteristic.

D. Achievability scheme for G

For certain classes of undirected simple connected graphs, the corresponding sum-network is such that we can demonstrate a linear network coding scheme that achieves the upper bound in Theorem 1. Towards this end, we first assign non-negative integers to variables $m_{(i,j)}(i)$ and $m_{(i,j)}(j)$, for all $(i,j) \in \widetilde{E}$, that satisfy certain constraints. The constraints depend on whether we use Construction 1 or 2 for constructing G and are given below.

Feasible solution for Construction 1.

$$m_{(i,j)}(i) + m_{(i,j)}(j) = b, \quad \forall (i,j) \in \widetilde{E}. \quad \text{(CODE-FEAS-1)}$$

$$\sum_{j:(i,j)\in\text{In}_{\widetilde{G}}(i)} m_{(i,j)}(i) \leq |\widetilde{E}|, \quad \forall i \in \widetilde{V}.$$

Feasible solution for Construction 2.

$$m_{(i,j)}(i) + m_{(i,j)}(j) = b, \quad \forall (i,j) \in \widetilde{E}. \quad \text{(CODE-FEAS-2)}$$

$$\sum_{j:(i,j) \in \text{In}_{\widetilde{G}}(i)} m_{(i,j)}(i) \le |\widetilde{E}| + 1, \quad \forall i \in \widetilde{V}.$$

$$\sum_{i \in V_{\widetilde{Cyc}}} \left[|\widetilde{E}| + 1 - \sum_{j:(i,j) \in \text{In}_{\widetilde{G}}(i)} m_{(i,j)}(i) \right] \ge b.$$

As will be evident shortly, the existence of the variables $m_{(i,j)}(i)$ allow us to construct the achievability scheme that matches the upper bound in Theorem 1. Such feasible assignments do not exist for all graphs. However, for a large class of graphs, we can in fact arrive at an assignment. For instance, for a simple, regular graph where $|\widetilde{V}|$ is even, it can be seen that $m_{(i,j)}(i) = |\widetilde{V}|/2, \; \forall (i,j) \in \widetilde{E}$ results in a feasible assignment for CODE-FEAS-1. The following claim can be shown (the proof appears in the Appendix).

Claim 2: If $\widetilde{G}$ is a regular graph or a biregular bipartite graph, then CODE-FEAS-1 has a feasible solution.

1) Linear network code for Construction 1: We construct a (r, l) network code for the sum-network G constructed above, by choosing $r = |\widetilde{V}|$ and $l = |\widetilde{V}| + |\widetilde{E}|$. In the discussion below, we assume that $\alpha = 1$. However, it will be evident that the case of higher α is a simple extension of the scheme for $\alpha = 1$. Here we describe the encoding function ϕ_{e_i} for bottleneck edge e_i, and assign the other edges to simply forward the messages that they receive.

We number the edges in $\widetilde{E}$ in an arbitrary order, so that the edges can be indexed as $\tilde{e}_1, \ldots, \tilde{e}_{|\widetilde{E}|}$. Then the source processes can be represented as a vector $\bar{\mathbf{X}}$ of dimension $r(|\widetilde{V}| + |\widetilde{E}|)$ as shown below.

$$\bar{\mathbf{X}} = [X_1^T \; X_2^T \; \ldots \; X_{|\widetilde{V}|}^T \; X_{\tilde{e}_1}^T \; X_{\tilde{e}_2}^T \; \ldots \; X_{\tilde{e}_{|\widetilde{E}|}}^T]^T$$

where $X_i, i = 1, \ldots, |\widetilde{V}|$, $X_{\tilde{e}_i}, i = 1, \ldots, |\widetilde{E}|$ are each of dimension $r \times 1$. Recall that $\text{tail}(e_i)$ is connected to the set of sources in A_i, thus $\phi_{e_i}(\bar{\mathbf{X}})$ is only a function of the sources in A_i. Moreover, $\text{head}(e_i)$ is connected to terminals t_i and t_e, where $e \in \text{In}_{\widetilde{G}}(i)$ and also $t_\star$. We partition the vector $\phi_{e_i}(\bar{\mathbf{X}})$ as follows.

$$\phi_{e_i}(\bar{\mathbf{X}})^T = [\phi_{e_i}(\bar{\mathbf{X}})_1^T \; \phi_{e_i}(\bar{\mathbf{X}})_{\hat{e}_1}^T \; \ldots \; \phi_{e_i}(\bar{\mathbf{X}})_{\hat{e}_{|\text{In}(i)|}}^T]$$

where $\phi_{e_i}(\bar{\mathbf{X}})_1$ is of dimension $r \times 1$. Furthermore, $\hat{e}_1, \ldots, \hat{e}_{|\text{In}_{\widetilde{G}}(i)|} \in \text{In}_{\widetilde{G}}(i)$ and $\phi_{e_i}(\bar{\mathbf{X}})_{\hat{e}_j}$ is of dimension $m_{\hat{e}_j}(i) \times 1$ ($m_{\hat{e}_j}(i)$ is obtained from CODE-FEAS-1). We set

$$\phi_{e_i}(\bar{\mathbf{X}})_1 = \sum_{\beta \in A_i} X_\beta$$

Next, for $\hat{e}_j = (u, i) \in \text{In}_{\widetilde{G}}(i)$, we set

$$\phi_{e_i}(\bar{\mathbf{X}})_{\hat{e}_j} = \begin{cases} [I_{m_{\hat{e}_j}(i) \times m_{\hat{e}_j}(i)} \; \mathbf{0}]X_{(u,i)} & \text{if } i < u \\ [\mathbf{0} \; I_{m_{\hat{e}_j}(i) \times m_{\hat{e}_j}(i)}]X_{(u,i)} & \text{otherwise.} \end{cases}$$

It is evident that the assignment for the encoding function $\phi_{e_i}(\bar{\mathbf{X}})$ discussed above is feasible since

1) it is a function only of sources in A_i, and
2) the row dimension of the transformation is
$$r + \sum_{\hat{e}_j \in \text{In}(i)} m_{\hat{e}_j}(i) \le |\widetilde{V}| + |\widetilde{E}|.$$

Theorem 2: The sum of sources Z, can be computed at all the terminals with the encoding functions specified above.

Proof: The terminal $t_i, i = 1, \ldots, |\widetilde{V}|$ is connected to $\text{head}(e_i)$ in G for $i \in \{1, \ldots, |\widetilde{V}|\}$. Therefore, from $\phi_{e_i}(\bar{\mathbf{X}})_1$ it recovers $\sum_{\beta \in A_i} X_\beta$. Furthermore, it has access to the set of sources in A_i^c via direct edges. Thus, it can compute the sum.

Next, consider a terminal t_e where $e = (i, j) \in \widetilde{E}$ such that $i < j$. From the assignment above it can be seen that t_e can recover X_e since

$$\begin{bmatrix} \phi_{e_i}(\bar{\mathbf{X}})_e \\ \phi_{e_j}(\bar{\mathbf{X}})_e \end{bmatrix} = \begin{bmatrix} [I_{m_e(i) \times m_e(i)} \; \mathbf{0}]X_{(i,j)} \\ [\mathbf{0} \; I_{m_e(j) \times m_e(j)}]X_{(i,j)} \end{bmatrix} = X_e,$$

since $m_e(i) + m_e(j) = b = r$. Next, note that $\sum_{\beta \in A_i} X_\beta + \sum_{\beta' \in A_j} X_{\beta'} - X_e = \sum_{\beta \in A_i \cup A_j} X_\beta$, since $A_i \cap A_j = X_e$. Moreover, t_e, gets all the sources $(A_i \cup A_j)^c$ via direct edges. Thus, it can compute the sum.

Finally, for $t_\star$, note that by previous arguments, it can recover any $X_e, \; e \in |\widetilde{E}|$ from the bottleneck edges. Following this, it can recover $X_i = \phi_{e_i}(\bar{\mathbf{X}})_1 - \sum_{e \in \text{In}_{\widetilde{G}}(i)} X_e$ for all $i \in \widetilde{V}$ and consequently the sum. ∎

Example 2: We describe the edge functions and decoding procedure for the case when $\widetilde{G} = K_3$, i.e. the complete graph on three vertices. We note that the following assignment satisfies the constraints in CODE-FEAS-1.

$$m_{(1,2)}(1) = 1 \quad m_{(1,2)}(2) = 2,$$
$$m_{(1,3)}(1) = 2 \quad m_{(1,3)}(3) = 1, \text{ and}$$
$$m_{(2,3)}(2) = 1 \quad m_{(2,3)}(3) = 2.$$

The linear encoding functions are shown in Fig. 4.

2) Linear network code for Construction 2: For the second construction, we have one additional source $X_\star$ and we construct a (r, l) network code for the sum-network with $r = |\widetilde{V}|$ and $l = |\widetilde{V}| + |\widetilde{E}| + 1$. Similar to the previous case we index the edges so that the source processes can be stacked in a vector $\bar{\mathbf{X}}$ of dimension $r(|\widetilde{V}| + |\widetilde{E}| + 1)$ as follows

$$\bar{\mathbf{X}} = [X_1^T \; X_2^T \; \ldots \; X_{|\widetilde{V}|}^T \; X_{\tilde{e}_1}^T \; X_{\tilde{e}_2}^T \; \ldots \; X_{\tilde{e}_{|\widetilde{E}|}}^T \; X_\star^T]^T$$

where $X_i, i = 1, \ldots, |\widetilde{V}|$, $X_{\tilde{e}_i}, i = 1, \ldots, |\widetilde{E}|$ and $X_\star$ are each of dimension $r \times 1$. The constraints satisfied by variables $m_{(i,j)}(i)$ is as outlined in CODE-FEAS-2. Now, for $i \notin V_{\widetilde{Cyc}}$, the construction of $\phi_{e_i}(\bar{\mathbf{X}})$ is exactly the same as in the previous one.

For $i \in V_{\widetilde{Cyc}}$, the vector $\phi_{e_i}(\bar{\mathbf{X}})$ also contains information from source $s_\star$ as follows

$$\phi_{e_i}(\bar{\mathbf{X}})^T = [\phi_{e_i}(\bar{\mathbf{X}})_1^T \; \phi_{e_i}(\bar{\mathbf{X}})_{\hat{e}_1}^T \; \ldots \; \phi_{e_i}(\bar{\mathbf{X}})_{\hat{e}_{|\text{In}(i)|}}^T \; \phi_{e_i}(\bar{\mathbf{X}})_\star^T]$$

where $\phi_{e_i}(\bar{\mathbf{X}})_1$ is of dimension $r \times 1$. Also, $\hat{e}_1, \ldots, \hat{e}_{|\text{In}_{\widetilde{G}}(i)|} \in \text{In}_{\widetilde{G}}(i)$ and $\phi_{e_i}(\bar{\mathbf{X}})_{\hat{e}_j}$ is of dimension $m_{\hat{e}_j}(i) \times 1$. $\phi_{e_i}(\bar{\mathbf{X}})_\star$ is of dimension $w_i \times 1$ where

$$w_i = |\widetilde{E}| + 1 - \sum_{j:(i,j) \in \text{In}_{\widetilde{G}}(i)} m_{(i,j)}(i).$$

$$\bar{\mathbf{X}} = \begin{bmatrix} X_1^T & X_2^T & X_3^T & X_{(1,2)}^T & X_{(1,3)}^T & X_{(2,3)}^T \end{bmatrix}^T \tag{6}$$

$$\phi_{e_1}(\bar{\mathbf{X}}) = \begin{bmatrix} I_3 & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & I_3 & I_3 & \mathbf{0}_{3\times3} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & [1\ \mathbf{0}] & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & [I_2\ \mathbf{0}] & \mathbf{0} \end{bmatrix} \bar{\mathbf{X}}$$

$$\phi_{e_2}(\bar{\mathbf{X}}) = \begin{bmatrix} \mathbf{0}_{3\times3} & I_3 & \mathbf{0}_{3\times3} & I_3 & \mathbf{0}_{3\times3} & I_3 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & [\mathbf{0}\ I_2] & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & [1\ \mathbf{0}] \end{bmatrix} \bar{\mathbf{X}}$$

$$\phi_{e_3}(\bar{\mathbf{X}}) = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & I_3 & \mathbf{0}_{3\times3} & I_3 & I_3 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & [\mathbf{0}\ 1] & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & [\mathbf{0}\ I_2] \end{bmatrix} \bar{\mathbf{X}}$$

Fig. 4. Encoding functions for the sum-network constructed from K_3 (see discussion in Example 2). It can observed that $t_{1,2}$ can be satisfied since can recover the first component of $X_{(1,2)}$ from $\phi_{e_1}(\bar{\mathbf{X}})$ and the remaining components from $\phi_{e_2}(\bar{\mathbf{X}})$, following which it can recover $X_1 + X_2 + X_{(1,2)} + X_{(2,3)} + X_{(1,3)}$. It has X_3 available via a direct edge. In a similar fashion, it can be verified that all terminals can be satisfied.

Note that $w_i \geq 0$ for $i \in V_{\widetilde{Cyc}}$ owing to the second constraint of CODE-FEAS-2.

$\phi_{e_i}(\bar{\mathbf{X}})_1$ and $\phi_{e_i}(\bar{\mathbf{X}})_{\hat{e}_j}$ for $\hat{e}_j \notin E_{\widetilde{Cyc}}$ are defined exactly as in previous construction. For $\hat{e}_j = (u,i) \in E_{\widetilde{Cyc}}$ there is a slight modification as follows.

$$\phi_{e_i}(\bar{\mathbf{X}})_{\hat{e}_j} = \begin{cases} [I_{m_{\hat{e}_j}(i) \times m_{\hat{e}_j}(i)} \ \mathbf{0}](X_{(u,i)} + X_\star) & \text{if } i < u, \\ [\mathbf{0} \ I_{m_{\hat{e}_j}(i) \times m_{\hat{e}_j}(i)}](X_{(u,i)} + X_\star) & \text{otherwise.} \end{cases}$$

We let

$$\gamma_{w_i} = \sum_{\{j : j \in V_{\widetilde{Cyc}}, j < i\}} w_j$$
$$\gamma'_{w_i} = \sum_{\{j : j \in V_{\widetilde{Cyc}}, j > i\}} w_j.$$

Then, $\phi_{e_i}(\bar{\mathbf{X}})_\star$ is defined as

$$\phi_{e_i}(\bar{\mathbf{X}})_\star = \begin{bmatrix} \mathbf{0}_{w_i \times r(|\widetilde{E}| + |\widetilde{V}|)} & \mathbf{0}_{w_i \times \gamma_{w_i}} & I_{w_i} & \mathbf{0}_{w_i \times \gamma'_{w_i}} \end{bmatrix} \bar{\mathbf{X}} \tag{7}$$

Such an assignment for the encoding function $\phi_{e_i}(\bar{\mathbf{X}})$ for $i \in V_{\widetilde{Cyc}}$ is feasible as
1) $\text{tail}(e_i)$ is connected to the starred source $s_\star$, and
2) the row dimension of the transformation is

$$r + \sum_{\hat{e}_j \in \text{In}_{\bar{G}}(i)} m_{\hat{e}_j}(i) + w_i$$
$$= r + \sum_{\hat{e}_j \in \text{In}(i)} m_{\hat{e}_j}(i) + \left(|\widetilde{E}| + 1 - \sum_{j : (i,j) \in \text{In}_{\bar{G}}(i)} m_{(i,j)}(i)\right)$$
$$= |\widetilde{E}| + |\widetilde{V}| + 1$$

Theorem 3: The sum of sources Z, can be computed at all the terminals with the encoding functions specified above.

Proof: For terminals other than $t_\star$ and t_e where $e \in E_{\widetilde{Cyc}}$, the sum can be computed in the same fashion as described in Theorem 2.

Consider terminal t_e where $e = (i,j) \in E_{\widetilde{Cyc}}$ such that $i < j$. t_e is connected to $\text{head}(e_i)$ and $\text{head}(e_j)$. t_e can recover the sum $X_e + X_\star$ by

$$X_e + X_\star = \begin{bmatrix} \phi_{e_i}(\bar{\mathbf{X}})_e \\ \phi_{e_j}(\bar{\mathbf{X}})_e \end{bmatrix}.$$

Note that $\sum_{\beta \in A_i} X_\beta + \sum_{\beta' \in A_j} X_{\beta'} - (X_e + X_\star) = \sum_{\beta \in A_i \cup A_j} X_\beta$, since $A_i \cap A_j = \{X_e, X_\star\}$. Moreover, t_e, gets all the sources $(A_i \cup A_j)^c$ via direct edges. Thus, it can compute the sum.

Terminal $t_\star$ can obtain $X_\star$ as follows.

$$X_\star^T = \begin{bmatrix} \phi_{e_{i_1}}(\bar{\mathbf{X}})_\star^T & \phi_{e_{i_2}}(\bar{\mathbf{X}})_\star^T & \cdots & \phi_{e_{i_{|V_{\widetilde{Cyc}}|}}}(\bar{\mathbf{X}})_\star^T \end{bmatrix},$$

where $i_1 < i_2 < \ldots < i_{|V_{\widetilde{Cyc}}|}$ all belong to $V_{\widetilde{Cyc}}$. This is because

$$\sum_{i \in V_{\widetilde{Cyc}}} w_i = \sum_{i \in V_{\widetilde{Cyc}}} \left[|\widetilde{E}| + 1 - \sum_{j : (i,j) \in \text{In}_{\bar{G}}(i)} m_{(i,j)}(i)\right] \geq b$$

owing to the constraints in CODE-FEAS-2. Moreover, eq. (7) shows that the different bottleneck edges recover all the disjoint subsets of $X_\star$.

For $e \notin E_{\widetilde{Cyc}}$, $t_\star$ can recover X_e in the same way as t_e. For $e = (i,j) \in E_{\widetilde{Cyc}}$ $i < j$, $t_\star$ can recover X_e as

$$\begin{bmatrix} \phi_{e_i}(\bar{\mathbf{X}})_e \\ \phi_{e_j}(\bar{\mathbf{X}})_e \end{bmatrix} - X_\star = X_e + X_\star - X_\star$$

Thus, $t_\star$ can recover any $X_e, e \in \widetilde{E}$. Following this it can be seen that it can also recover X_i for all $i \in V$ and then the sum. ∎

The linear network code described here assumed edges of unit capacity. The same scheme can be used in sum-networks where all the edges have integer edge capacity $\alpha > 1$. In order to see this, we think of every α capacity edge as a union of α unit capacity edges between the same two vertices.

Then we can think of this modified network as a union of α sub-networks, each of which is topologically equivalent to the original network but consists of only unit capacity edges. We can use the linear network code described above to achieve the coding capacity $\frac{r}{l}$ on each of these unit edge capacity networks. Thus, at the very least, we will be able to transmit a sum of sources $\in \mathcal{A}^{\alpha r}$ to all the terminals by a repeated application of the same network code on all α sub-networks. Thus, this will be a rate $\frac{\alpha r}{l}$ solution to the original sum-network. However, as evinced by Theorem 1 this is also the upper bound on the rate and hence this repeated application of our linear network code achieves capacity.

IV. Comparison with existing results

The work most closely related to ours is by Rai & Das [13]. They showed that there exists a sum-network that has coding capacity $\frac{p}{q}$ for any p, q and constructed a linear code that achieves capacity on an instance of such a sum-network. They also pointed out that finding sum-networks that have the same coding capacity but with fewer sources and terminals was an open problem. We now demonstrate that our approach is a strict generalization of the work of [13].

Their construction for a network with capacity p/q starts by constructing a base network that has a capacity of $1/q$. The base network only has unit capacity edges. The edges are replicated p times to obtain a network with capacity p/q. We now show that our approach gives their result as a special case when $\widetilde{G}$ is chosen to be the complete graph on $2q-1$ vertices.

Example 3: Consider $\widetilde{G}$ to be a complete graph on $2q-1$ vertices. This graph has a feasible assignment to constraints in CODE-FEAS-1 by Claim 2. Construct the sum-network G from $\widetilde{G}$ without using the extra source $s_\star$. Then, by using Theorems 1 and 2 the coding capacity of G is $\frac{2q-1}{2q-1+\binom{2q-1}{2}} = \frac{1}{q}$. By replicating each edge p times, we obtain a coding rate of p/q.

However, it is important to note that our framework allows us more parameters, which can be chosen to obtain stronger results in the sense that specific coding rates can be achieved by using fewer sources and terminals.

Example 4: Suppose that the approach of [13] was used to construct a sum-network with capacity $2/5$. It can be verified that this requires a sum-network with 45 sources and 46 terminals. In contrast, in our approach we can choose $\widetilde{G}$ to be the graph in Fig. 1 (a feasible assignment for CODE-FEAS-2 can be easily derived). In this case, we obtain a sum-network G (with source $s_\star$) with capacity $\frac{4}{4+5+1} = \frac{2}{5}$. However, our sum-network only has 9 sources and 10 terminals.

As another example, we can construct a network with capacity $5/13$ that is significantly smaller.

Example 5: Let $\widetilde{G}$ be the Petersen graph [14], which is a 3-regular graph on 10 vertices (see Figure 5). Construct sum-network G (with source $s_\star$). For this, choose the shortest cycle as $V_{\widetilde{Cyc}} = \{1, 2, 3, 4, 5\}$. The assignment for the variables in
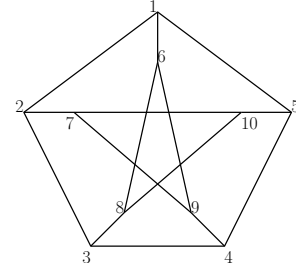

Fig. 5. Petersen graph.

CODE-FEAS-2 can be done as follows

$$
m_{(i,j)}(i) = \begin{cases} 4 & \text{if } i \in V_{\widetilde{Cyc}} \text{ and } j \notin V_{\widetilde{Cyc}}, \\ 6 & \text{if } i \notin V_{\widetilde{Cyc}} \text{ and } j \in V_{\widetilde{Cyc}}, \text{ and} \\ 5 & \text{otherwise.} \end{cases}
$$
$$
w_i = 2 \quad \text{for all } i \in V_{\widetilde{Cyc}}.
$$

The coding capacity of G can be verified to be $\frac{10}{10+\frac{3\times 10}{2}+1} = \frac{5}{13}$ and it has 26 sources and 26 terminals. However, the approach of [13] would need $2 \times 13 - 1 + \binom{2\times 13 - 1}{2} = 325$ sources and 326 terminals.

To a certain extent we can analyze the parameters that can be obtained by considering regular graphs. Let $\widetilde{G}$ be a d-regular graph on b vertices, hence $d \leq b - 1$. Then the upper bounds that we have on the coding rate for Constructions 1 and 2 respectively are

$$
\frac{r}{l} \leq \frac{\alpha b}{b + \frac{db}{2}} = \frac{2\alpha}{2+d}, \tag{8}
$$

and

$$
\frac{r}{l} \leq \frac{\alpha b}{b + \frac{db}{2} + 1}. \tag{9}
$$

Without loss of generality consider p/q where p and q are coprime. It can be seen that (8) can achieve any such p/q by setting $\alpha = p$ and $d = 2q - 2$. The resulting G is a $2q - 2$-regular graph on atleast $2q - 1$ vertices, in which case it is just the complete graph on $2q - 1$ vertices. This recovers the result of [13].

We can also choose $\widetilde{G}$ to be a biregular bipartite graph with n_l vertices of degree d_l each in one part and n_r vertices of degree d_r each in the other part. Without the source $s_\star$, the coding capacity for a network constructed from such a graph is

$$
\frac{n_l + n_r}{n_l + n_r + \frac{n_l d_l + n_r d_r}{2}} = \frac{2}{2 + \frac{n_l d_l + n_r d_r}{n_l + n_r}} = \frac{2}{2 + d_{\text{av}}} \tag{10}
$$

where d_{av} is the average degree of the graph. Here d_{av} is not necessarily an integer and thus opens up more possibilities of coding capacities that can be achieved from this graph construction as opposed to (8) where d has to be an integer.

Example 6: Consider $\widetilde{G}$ to be the complete bipartite graph $K_{3,5}$. The sum-network G constructed from this graph has coding capacity $\frac{2}{2+\frac{3\times 5 + 5 \times 3}{8}} = \frac{8}{23}$ and has 23 sources and 24

terminals. The approach of [13] would require $45 + \binom{45}{2} = 1035$ sources and 1036 terminals.

It can also be seen that for any specified minimum cut between source terminal pairs, we can always choose an appropriate regular graph $\widetilde{G}$ so that the corresponding sum network has a capacity strictly smaller than one. Thus, it can be inferred that any min-cut ($= \alpha \times b$) between each source-terminal pair can never guarantee solvability.

V. Conclusions and future work

In this paper, we have constructed a large class of sum-networks for which we can determine the capacity. These sum-networks are in general, smaller (with fewer sources and terminals) than sum-networks known to achieve the said capacity and answer a question raised in prior work. The construction of these sum-networks with the help of undirected graphs allows us to identify certain graph-theoretic properties that aid in constructing a capacity-achieving linear network code. Future work will involve analyzing these properties in greater detail and also examining whether there are other combinatorial structures that can be used to construct sum-networks.

Appendix

Proof of Claim 1.
We only need to rule out the possibility that $(i,j) \notin E_{\widetilde{Cyc}}$, but $(i,j) \in \widetilde{E}$. But this is impossible, since if $(i,j) \in \widetilde{E}$, we could find a shorter cycle in $\widetilde{G}$ than $\widetilde{Cyc}$.

Proof of Claim 2.
Suppose $\widetilde{G}$ is a k-regular graph on b vertices. If b is even, then assigning

$$m_{(i,j)}(i) = \frac{b}{2} \text{ and } m_{(i,j)}(j) = \frac{b}{2} \quad \forall (i,j) \in \widetilde{E} \qquad (11)$$

satisfies the first two constraints. Also, it satisfies the third constraint as

$$\sum_{j:(i,j) \in \text{In}_{\widetilde{G}}(i)} m_{(i,j)}(i) = \frac{kb}{2} = |\widetilde{E}| \quad \forall i \in \widetilde{V}. \qquad (12)$$

If b is odd, then k is even and we can construct an *Euler tour* [14] of $\widetilde{G}$, i.e., a cycle that traverses every edge in $\widetilde{E}$ exactly once, though it may visit a vertex any number of times. Suppose we start at vertex 1 and the Euler tour is of the form

$$[(1,j_1),(j_1,j_2),\ldots,(j_f,1),(1,j_{f+1}),\ldots,(j_F,u)] \qquad (13)$$

where $f, f+1$ are such that the Euler tour passes through the sequence of vertices $j_f \rightarrow 1 \rightarrow j_{f+1}$ and F is such that the edge (j_F, u) is the last edge traversed on the Euler tour.

Then we can arrange the variables $m_{(i,j)}(i) \; \forall (i,j) \in \widetilde{E}$ as a one-to-one correspondence with (13) in the following way

$$[m_{(1,j_1)}(1), m_{(1,j_1)}(j_1), m_{(j_1,j_2)}(j_1), \ldots, \qquad (14)$$
$$m_{(j_f,1)}(1), m_{(1,j_{f+1})}(1), \ldots, m_{(j_F,1)}(1)]$$

Assigning consecutive terms in (14) alternately as $\lfloor \frac{b}{2} \rfloor$ and $\lceil \frac{b}{2} \rceil$, we see that for every vertex i, the value assigned to $m_{(i_{\text{in}},i)}(i)$ is different from $m_{(i,i_{\text{out}})}(i)$, where the Euler tour

progresses in the order $i_{\text{in}} \rightarrow i \rightarrow i_{\text{out}}$. We can see that such an assignment satisfies the first constraint.

Also since there are only two possible values, we conclude that there are equal number of variables $m_{(i,j)}(i)$ with values equal to $\lfloor \frac{b}{2} \rfloor$ and $\lceil \frac{b}{2} \rceil$ for all $i \in \widetilde{V}$. Hence, we have $\forall i \in \widetilde{V}$

$$\sum_{j:(i,j) \in \text{In}_{\widetilde{G}}(i)} m_{(i,j)}(i) =$$
$$\frac{k}{2}\left\lfloor \frac{b}{2} \right\rfloor + \frac{k}{2}\left\lceil \frac{b}{2} \right\rceil = \frac{kb}{2} = |\widetilde{E}|$$

Next, consider a biregular bipartite simple graph $\widetilde{G}$ with n_l vertices in one part and n_r vertices in the other part. Each vertex in the first part has degree d_l and each vertex in the other part has degree d_r. Since $\widetilde{G}$ is simple, we must have that $d_l \le n_r$ and $d_r \le n_l$. Every edge e in the graph is of the form (i_l, i_r) where i_l is a vertex in one part while i_r is a vertex in the other part. Then it is easy to see that the following assignment satisfies the constraints in CODE-FEAS-1.

$$m_e(i_l) = n_l$$
$$m_e(i_r) = n_r$$

as $\forall l, r \in \widetilde{V}$ and $\forall e \in \widetilde{E}$

$$m_e(l) + m_e(r) = n_l + n_r = b$$
$$\sum_{e_i:e_i \in \text{In}(l)} m_{e_i}(l) = d_l \times n_l = |\widetilde{E}|$$

References

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

[2] ——, "Linear codes, target function classes, and network computing capacity," *Information Theory, IEEE Transactions on*, vol. 59, no. 9, pp. 5741–5753, Sept 2013.

[3] A. Ramamoorthy and M. Langberg, "Communicating the sum of sources over a network," *IEEE Journal on Selected Areas in Communication: Special Issue on In-network Computation: Exploring the Fundamental Limits*, vol. 31(4), pp. 655–665, 2013.

[4] B. K. Rai and B. K. Dey, "On network coding for sum-networks," *IEEE Trans. on Info. Th.*, vol. 58, no. 1, pp. 50 –63, 2012.

[5] S. Huang and A. Ramamoorthy, "On the multiple unicast capacity of 3-source, 3-terminal directed acyclic networks," *IEEE/ACM Trans. on Networking*, vol. 22(1), pp. 285–299, 2014.

[6] ——, "An achievable region for the double unicast problem based on a minimum cut analysis," *IEEE Trans. on Comm.*, vol. 61(7), pp. 2890–2899, 2013.

[7] J. Korner and K. Marton, "How to encode the modulo-2 sum of binary sources," *IEEE Trans. on Info. Th.*, vol. 25, no. 2, pp. 219–221, 1979.

[8] A. Orlitsky and J. R. Roche, "Coding for computing," *IEEE Trans. on Info. Th.*, vol. 47, no. 3, pp. 903–917, 2001.

[9] V. Doshi, D. Shah, M. Médard, and S. Jaggi, "Distributed Functional Compression through Graph Coloring," in *Data Compression Conference*, 2007, pp. 93–102.

[10] A. Ramamoorthy, "Communicating the sum of sources over a network," in *IEEE Intl. Symposium on Info. Th.*, 2008, pp. 1646–1650.

[11] S. Shenvi and B. K. Dey, "A Necessary and Sufficient Condition for Solvability of a 3s/3t sum-network," in *IEEE Intl. Symposium on Info. Th.*, 2010, pp. 1858–1862.

[12] B. K. Rai and N. Das, "On the capacity of ms/3t and 3s/nt sum-networks," in *IEEE Information Theory Workshop (ITW)*, 2013, pp. 1–5.

[13] ——, "On the capacity of sum-networks," in *2013 51st Annual Allerton Conference on Communication, Control, and Computing*, 2013, pp. 1545–1552.

[14] R. Diestel, *Graph Theory, 4th Edition*. Springer, 2012.