Network Coding: A Computational Perspective

Michael Langberg[1] Alexander Sprintson[2] Jehoshua Bruck[1]

***Abstract*— In this work, we study the computational perspective of network coding, focusing on two issues. First, we address the computational complexity of finding a network code for acyclic multicast networks. Second, we address the issue of reducing the amount of computation performed by network nodes. In particular, we consider the problem of finding a network code with the minimum possible number of *encoding nodes*, i.e., nodes that generate new packets by combining the packets received over incoming links.**

We present a deterministic algorithm that finds a feasible network code for a multicast network over an underlying graph $G(V, E)$ in time $O(|E|kh + |V|k^2h^2 + h^4k^3(k + h))$, where k is the number of destinations and h is the number of packets. Our result improves the best known running time of $O(|E|kh + |V|k^2h^2(k + h))$ of the algorithm due to Jaggi et al. [1] in the typical case of large communication graphs. In addition, our algorithm guarantees that the number of encoding nodes in the obtained network code is bounded by $O(h^3k^2)$.

Next, we address the problem of finding a network code with the minimum number of encoding nodes in both integer and fractional coding networks. We prove that in the majority of settings this problem is $\mathcal{NP}$-hard. However, we show that if $h = O(1)$, $k = O(1)$, and the underlying communication graph is acyclic, then there exists an algorithm that solves this problem in polynomial time.

I. Introduction

The new paradigm of network coding promises to benefit many areas of communication and networking [2]. The network coding approach generalizes traditional routing by allowing intermediate network nodes to generate new packets by combining incoming data packets.

Establishing efficient multicast connections is a central problem in network coding. In the *multicast network coding problem* a source s needs to deliver h packets to a set T of k terminals over the underlying communication graph G. It was shown in [2] and [3] that the capacity of the network, i.e., the maximum number of packets that can be sent between s and T per time unit, is equal to the minimum size of a cut that separates the source s from a terminal $t \in T$. Specifically, a source s can send h packets to all terminals T if and only if the total capacity of all links in any cut that separates s and $t \in T$ is at least h. Li et. al. [3] proved that *linear network codes* are sufficient for achieving the capacity of the network. In a subsequent work, Koetter and Médard [4] developed an algebraic framework for network coding and investigated linear network codes for directed graphs with cycles. This framework was used by Ho et al. [5] to show that linear network codes can be efficiently constructed through a randomized algorithm. Jaggi et al. [1] proposed a deterministic polynomial-time algorithm for finding feasible network codes for multicast networks.

In this paper we study the computational perspective of multicast network coding. Our goal to minimize (i) The time required for finding an feasible network code; (ii) The total amount of computation performed by network nodes. In particular, we consider the problem of finding a network code that uses a bounded number of *encoding nodes*. Encoding nodes generate new packets by combining the packets received over incoming links, in contrast to *forwarding* nodes that can only forward and duplicate incoming packets.

We study both *fractional* and *integer* coding networks. In fractional coding networks, each packet can be split into a number of smaller packets, each of which is sent over different paths. In integer coding networks packets cannot be split and have to be sent through the network in one piece.

A. Our results

Our study makes the following contributions. First, we present an efficient algorithm for integer coding networks. Given an acyclic multicast network with h packets and k terminals, our algorithm finds a network code that includes at most $O(h^3k^2)$ encoding nodes. The computational complexity of our algorithm is $O(|E| + |V|k^2 + k^4)$ for $h = 2$ and $O(|E|kh + |V|k^2h^2 + h^4k^3(k + h))$ for general h. Our algorithm improves the previously best known running time of $O(|E|kh + |V|k^2h^2(k + h))$ of the algorithm due to Jaggi et al. [1]. The improvement is most significant in the case of sparse graphs with a large number of nodes, which is typically the case in communication networks.

Second, we study the problem of finding a network code with the minimum possible number of encoding nodes, considering both integer and fractional coding networks. We prove that in the majority of settings this problem is $\mathcal{NP}$-hard. However, we show that if $h = O(1)$, $k = O(1)$, and the underlying communication graph is acyclic, then the problem can be solved in polynomial time. Our results are summarized in Figure 1.

B. Related work

In a previous work of ours [6] we established a lower and an upper bounds of $\Omega(h^2k)$ and h^3k^2, respectively. In addition, we proved that finding the minimum number of encoding

[1]Michael Langberg and Jehoshua Bruck are with the Department of Electrical Engineering, California Institute of Technology, Pasadena, California, USA. Email: {mikel,bruck}@caltech.edu. Research supported in part by by the Caltech Lee Center for Advanced Networking and by NSF grants ANI-0322475 and CCF-0346991.

[2]Alexander Sprintson is with the Department of Electrical and Computer Engineering, Texas A&M University, College Station, Texas, USA. Email: spalex@ece.tamu.edu.

Type of Coding Network	Restrictions	Acyclic	General(cyclic)
Integer	k and h are constant	DTIME$(n^{O(h^3k^2)})$ (Theorem 14)	$\mathcal{NP}$-hard (Theorem 12)
	no restrictions	$\mathcal{NP}$-hard (Theorem 13)	
Fractional	k and h are constant	DTIME$(n^{O(h^3k^2)})$ (Theorem 15)	Not resolved in this work
	no restrictions	$\mathcal{NP}$-hard (Theorem 13)	$\mathcal{NP}$-hard (Theorem 13)

Fig. 1. Our results for the problem of finding a network code with the minimum possible number of encoding nodes.

nodes in integer coding networks with cycles is an $\mathcal{NP}$-hard problem. In this paper we use the results of [6] in order to devise an efficient algorithm for finding network codes with bounded number of encoding nodes.

The fastest deterministic algorithm for finding feasible network code for multicast networks prior to our work was due to Jaggi et al. [1]. Randomized algorithms for this problem have been presented in [1] and [5]. For acyclic graphs, the currently best known expected running time for randomized algorithms is $O(|E|kh + kh^{2.376})$ when the packet size may depend on the size of the network G and $O(|E|kh + |V|k^2h^3)$ when the packet size is independent of the size of G; both results appear in [1]. Recently, [7] proposed an algorithm for finding multicast network codes based on matrix completion. Their algorithm addresses a more general class of problems and has a running time of $\min(O(k|E|^3 \log|E|), O(|E|kh + |V|^3k^3h^3 \log(|V|h)))$. However, none of the previous work provide a non-trivial bound on the number of encoding nodes in the network.

A recent work by Bhattad et al. [8] considered several several optimization problems that arise in fractional multicast coding networks. This work models the flow of information in a coding network by a linear program with $O(|G|2^k)$ variables, where $|G|$ is the size of the underlying communication graph and k is the number of terminals. For small values of k this program enables to optimize several objective functions that are strongly related to the number of encoding nodes in coding networks. We use the framework of [8] in parts of this work.

The rest of the paper is organized as follows. In Section II, we formulate the network model and present the definition of integer coding networks. In Section III, we present our efficient algorithm for finding feasible network codes in integer networks. In Section IV, we define fractional coding networks and analyze the computational complexity of minimizing the number of encoding nodes in both fractional and integer coding networks.

II. Model

The communication network is represented by a directed graph $G = (V, E)$, where V is the set of nodes and E the set of links in G. The capacity c_e of link $e \in E$ is defined to be the number of packets that can be sent over e in one time unit. We assume that link capacities c_e are integer numbers. An instance $\mathbb{N}(G, s, T, h)$ of the multicast network coding problem is a 4-tuple that includes the graph $G(V, E)$, a source node $s \in V$, a set $T \subset V$ of terminals, and the number of packets h that must be transmitted from the source node s to every terminal $t \in T$. We assume, without loss of generality, that the source s has no incoming links and that the terminals T have no outgoing links. We also assume that each packet is an element of a finite field Σ. We denote the size $|T|$ of the terminal set by k. Each node $v \in G$, $v \neq s$, $v \notin T$ is referred to as an *internal* node.

In this section, we define network codes for acyclic integer coding networks. A more general settings of cyclic and fractional coding networks is discussed in Section IV.

Definition 1 (Integer network code $\mathbb{F}(\mathbb{N})$): A network code for $\mathbb{N}(G, s, T, h)$ is defined by the set of encoding functions $\mathbb{F}(\mathbb{N}) = \{f_e \mid e \in E\}$. If $e(v, u)$ is an outgoing link of the source node s, then f_e is a mapping from Σ^h to Σ^{c_e}. Otherwise, f_e is a mapping from $\Sigma^{c_{in}(v)}$ to Σ^{c_e}, where $c_{in}(v) = \sum_{(w,v)\in E} c_{(w,v)}$ is the total capacity of the incoming links of v.

The encoding function f_e of $e(v, u)$ determines the packets transmitted on link e for any possible combination of the packets available at the source (if $v = s$) or received over the incoming links of v (if $v \neq s$).

We focus on linear network codes $\mathbb{F}(\mathbb{N})$, i.e., for each $e \in E$ the encoding function f_e is a linear function over Σ. With linear network coding, each packet transmitted over link $e \in E$ is a linear combination of the h packets available at source s. Accordingly, we define a function $F_e : \Sigma^h \mapsto \Sigma^{c_e}$ that determines the packets transmitted on link e as a function of the packets available at s. If e is an outgoing link of the source node, then F_e is identical to f_e. For any other link $e(v, u) \in E$, F_e is defined as $F_e \equiv f_e(F_{e_1^v}, \ldots, F_{e^v_{d_{in}(v)}})$, where $d_{in}(v)$ is the in-degree of v and $\{e_1^v, \ldots, e^v_{d_{in}(v)}\}$ is the set of the incoming links of v.

A network code $\mathbb{F}(\mathbb{N})$ for $\mathbb{N}(G, s, T, h)$ is said to be feasible if for each destination node $t \in T$, there exists a decoding function $g_t : \Sigma^{c_{in}(t)} \mapsto \Sigma^h$ such that $g_t(F_{e_1^t}, \ldots, F_{e^t_{d_{in}(t)}})$ is the identify function, where $d_{in}(t)$ is the in-degree of t and $\{e_1^t, \ldots, e^t_{d_{in}(t)}\}$ is the set of the incoming links of t.

An instance $\mathbb{N}(G, s, T, h)$ of the multicast network coding problem is said to be feasible if there exists a feasible network code for $\mathbb{N}$. If $\mathbb{N}(G, s, T, h)$ is feasible we refer to h as the *rate* of the multicast coding network. The multicast capacity of the communication network G with respect to source s and set T of terminals is defined to be the maximum value of h such that the coding network $\mathbb{N}(G, s, T, h)$ is feasible. The *capacity* of the network is determined by the minimum capacity of a cut that separates the source s and any terminal $t \in T$ [2], [3], where the capacity of a cut is the sum of the capacities of the links that belong to the cut.

A coding network $\mathbb{N}(G, s, T, h)$ is said to be *minimal* with respect to link removal if (i) $\mathbb{N}(G, s, T, h)$ is feasible (ii) Removal of any link from G would violate the feasibility of $\mathbb{N}(G, s, T, h)$.

Let $\mathbb{N}(G, s, T, h)$ be a feasible coding network. We say that link $e \in G$ is *vital* if after removing e from G the resulting network is no longer feasible. Note that every link of the minimal network is vital.

Definition 2 (Encoding and forwarding links and nodes): Let $\mathbb{F}(\mathbb{N})$ be a network code. A link e is referred to as a *forwarding link* if it is an outgoing link of the source node s or if f_e can be decomposed to c_e functions $f_e^1, \ldots, f_e^{c_e}$ that map $\Sigma^{c_{in}(v)}$ to Σ such that each function f_e^i depends only on one variable, where $c_{in}(v) = \sum_{(w,v) \in E} c_{(w,v)}$. Otherwise, link e is referred to as an *encoding link*. We say that a node v, $v \neq s$, is an *encoding node* if at least one of its outgoing links (v, u) is encoding. If all outgoing links of a node v are forwarding, the node is referred to as a *forwarding node*.

Encoding links generate new packets by combining the packets received over the incoming links; forwarding links can only forward incoming packets.

We will use the following lemma implied by [6]:

Lemma 3 ([6]): Let $\mathbb{N}(G, s, T, h)$ be an acyclic coding network which is minimal with respect to link removal. Then, the number of internal nodes in G of degree 3 or more is bounded by $O(h^3k^2)$.

Let $\mathbb{N}(G(V, E), s, T, h)$ and $\hat{\mathbb{N}}(\hat{G}(\hat{V}, \hat{E}), \hat{s}, \hat{T}, \hat{h})$ be multicast coding networks. We say that $\hat{\mathbb{N}}$ is equivalent to $\mathbb{N}$ if the following three conditions hold (i) $\mathbb{N}$ is feasible if and only if $\hat{\mathbb{N}}$ is feasible; (ii) For any feasible network code $\hat{\mathbb{F}}(\hat{\mathbb{N}})$ for $\hat{\mathbb{N}}$, there exists a corresponding network code $\mathbb{F}(\mathbb{N})$ for $\mathbb{N}$ that includes the same or lower number of encoding nodes (iii) Such code can be found through an efficient procedure whose running time is bounded by $O(|E| + |\hat{E}|)$.

In some parts of our paper we use a notion of network flows [9].

Definition 4 (Flow): An *integer* (s, t)*-flow* θ is a function $\theta : E \mapsto \mathbb{R}$ that satisfies the following two properties:

1) For all $e = (u, v) \in E$, it holds that θ_e is an integer number that satisfies $0 \leq \theta_e \leq c_e$;
2) For each internal node $v \in V$, $v \neq s$, $v \notin T$ it holds that
$$\sum_{w:(w,v)\in E} \theta_{(w,v)} = \sum_{w:(v,w)\in E} \theta_{(v,w)}.$$

The *value* $|\theta|$ of a flow θ is defined as $|\theta| = \sum\limits_{v:(s,v)\in E} \theta_{(s,v)}$. If each link $e \in E$ is associated with a cost $\omega(e)$ then the cost $\omega(\theta)$ of a flow θ is defined as follows:

$$\omega(\theta) = \sum_{(u,v)\in E} \omega_{(u,v)} \cdot \theta_{(u,v)} \quad (1)$$

A minimum cost (s, t)-flow ω can be decomposed into a set of $|\omega|$ paths between s and t [9].

III. Algorithm for Computing a Feasible Network Code

In this section we present an algorithm that receives as input an acyclic coding network $\mathbb{N}(G, s, T, h)$ and computes a feasible integer network code for $\mathbb{N}$ over a field of size $k = |T|$. The computational complexity of our algorithm is $O(|E|kh + |V|k^2h^2 + h^4k^3(h + k))$.

A. Algorithm overview

Our algorithm uses three auxiliary coding networks $\mathbb{N}'(G', s, T, h)$, $\mathbb{N}^*(G^*, s, T, h)$, and $\hat{\mathbb{N}}(\hat{G}, s, T, h)$, all of them are equivalent to $\mathbb{N}(G, s, T, h)$.

The coding network $\mathbb{N}'(G', s, T, h)$ is constructed by Procedure Expand, described in Section III-B. This network has the following properties: (i) All links in G' are of capacity 1; (ii) The total number of links in G' is bounded by $|V|hk$; (iii) For each $t_i \in T$ there exist h node-disjoint paths in G' between s and t_i.

Next, we apply Algorithm Min-Global, described in Section III-D below. The algorithm constructs the auxiliary network $\mathbb{N}^*(G^*, s, T, h)$ by deleting redundant links from G' such that the number of nodes of in-degree more than 2 in G^* is bounded by $O(h^3k^2)$. Finally, we invoke Procedure Shrink, described in Section III-E below. This procedure constructs coding network $\hat{\mathbb{N}}(\hat{G}, s, T, h)$ by contracting all nodes in $\mathbb{N}^*(G^*, s, T, h)$ of degree 2. The number of links in $\hat{\mathbb{N}}(\hat{G}, s, T, h)$ is bounded by $O(h^3k^2)$.

This property enables us to find a network code for $\mathbb{N}$ by performing the following steps:

1) Construct an auxiliary coding network $\hat{\mathbb{N}}(\hat{G}, s, T, h)$;
2) Find a feasible network code $\hat{\mathbb{F}}(\hat{\mathbb{N}})$ for $\hat{\mathbb{N}}$, i.e., by applying the algorithm due to Jaggi et. al. [1];
3) Find a network code $\mathbb{F}(\mathbb{N})$ for $\mathbb{N}$ that corresponds to $\hat{\mathbb{F}}(\hat{\mathbb{N}})$.

which has the following properties: (i) $\hat{\mathbb{N}}$ is equivalent to $\mathbb{N}$; (ii) The number of links in $\hat{G}$ is bounded by $O(h^3k^2)$.

B. Procedure Expand

Procedure Expand begins by assigning a unit cost for each link $e \in E$. Then, the procedure finds, for each terminal $t_i \in T$, a minimum-cost (s, t_i)-flow θ_i of value h. In order to find a minimum cost flow we employ the Successive Shortest Path algorithm [9, Chapter 9]. Next, each link $e(v, u) \in E$ is substituted by $\max_{t_i \in T} \theta_i(e)$ parallel links of unit capacity that connect v and u. All links for which $\max_{t_i \in T} \theta_i(e) = 0$ are removed from the graph. Note that the resulting graph contains h link-disjoint paths between source s and any terminal $t_i \in T$. We denote a set of h link-disjoint paths between s and t_i by $\mathbb{P}_i$. The sets $\{\mathbb{P}_i \mid t_i \in T\}$ can be found by invoking the flow decomposition algorithm [9].

Finally, the procedure substitutes each internal node v in the resulting graph of degree larger than 3 by a gadget Γ_v, constructed as follows: Let E_v^{in} and E_v^{out} be the incoming and outgoing links of v, respectively. For every link $(x, v) \in E_v^{in}$ we add a node x' to Γ_v and a link (x, x'). Similarly, for every link $(v, y) \in E_v^{out}$ we add a node y' to Γ_v and a

```
Procedure EXPAND ($\mathbb{N}(G, s, T, h)$):
    Input:
        $\mathbb{N}$ - a feasible coding network;

    1  Assign a unit cost for every link $e \in E$.
    2  for each terminal $t_i \in T$ do
    3      Find a minimum cost $(s, t_i)$-flow $\theta_i$.
    4  for each link $e(v, u) \in E$ do
    5      if $\max_{t_i \in T} \theta_i(e) > 0$ then
    6          Replace $e$ by $\max_{t_i \in T} \theta_i(e)$ parallel links of
               capacity one between $v$ and $u$
    7      else
    8          Remove $e$
    9  for each terminal $t_i \in T$ do
    10     Find a set $\mathbb{P}_i$ of $h$ link-disjoint paths between $s$ and $t_i$.
    11 For each node $v \in G$, $v \neq s$, $v \notin T$ whose degree is more
       than 3, replace $v$ by a gadget $\Gamma_v$.
    12 Denote by $G'$ the resulting graph.
    13 Return $\mathbb{N}'(G', s, T, h)$.
```

Fig. 2. Procedure EXPAND

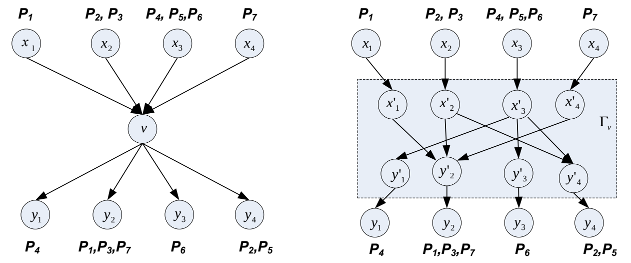

Fig. 3. Substituting a node v by a gadget Γ_v.

link (y', y). For each path $P \in \{\mathbb{P}_i \mid t_i \in T\}$ let x' be a node in Γ_v that corresponds to link $(x, v) \in P_i$ and y' be a node in Γ_v that corresponds to link $(v, y) \in P_i$. If Γ_v does not include link (x', y'), we add (x', y') to Γ_v. Fig. 3 demonstrates the construction of the subgraph Γ_v. The resulting graph is denoted by $G'(V', E')$. Note that in G' the paths corresponding to $\mathbb{P}_i$ are node disjoint.

The formal description of Procedure EXPAND appears in Fig. 2. We proceed to analyze the computational complexity of the procedure. Finding flows $\{\theta_i \mid t_i \in T\}$ and decomposing them into paths $\{\mathbb{P}_i \mid t_i \in T\}$ can be done in $O(|E|kh)$ time [9]. The total number of links that belong to paths in $\bigcup_{t_i \in T} \mathbb{P}_i$ is bounded by $|V|hk$. Since $G'(V', E')$ includes at most two links for every link that belongs to a path in $\bigcup_{t_i \in T} \mathbb{P}_i$, the number of links in G' is bounded by $O(|V|kh)$. Thus, the computational complexity of Procedure EXPAND is bounded by $O(|E|kh)$. In Theorem 7 below we show that the coding network $\mathbb{N}'(G', s, T, h)$ returned by Procedure EXPAND is equivalent to the original coding network $\mathbb{N}(G, s, T, h)$.

C. *Algorithm* MIN-LOCAL

We proceed to describe Algorithm MIN-LOCAL, which is an important building block of Algorithm MIN-GLOBAL, presented in the next section.

Algorithm MIN-LOCAL receives as input a coding network $\mathbb{N}'(G', s, T, h)$ and two sets of h link-disjoint paths $\mathbb{P}_i$, $\mathbb{P}_j$

```
Algorithm MIN-LOCAL ($\mathbb{N}'(G', s, T, h)$, $\mathbb{P}_i, \mathbb{P}_j$):
    Input:
        $\mathbb{N}'$ - a feasible coding network,
        $\mathbb{P}_i$ - a set of $h$ node-disjoint paths between $s$ and $t_i$,
        $\mathbb{P}_j$ - a set of $h$ node-disjoint paths between $s$ and $t_j$;

    1  $G_{ij} \leftarrow$ the subgraph of $G'$ induced by links that belong to
       paths in $\mathbb{P}_i$ and $\mathbb{P}_j$.
    2  Assign zero cost to every link that belongs to a path in $\mathbb{P}_i$ and
       assign a unit cost to every other link in $G_{ij}$.
    3  Find $h$ link-disjoint paths $\mathbb{P}^*_j$ in $G_{ij}$ between $s$ and $t_j$
       of minimum total cost.
    4  $G'_{ij} \leftarrow$ the subgraph of $G_{ij}$ induced by links that belong to
       paths in $\mathbb{P}_i$ and $\mathbb{P}^*_j$.
    5  Assign zero cost to every link that belongs to a path in $\mathbb{P}^*_j$ and
       assign a unit cost to every other link in $G'_{ij}$.
    6  Find $h$ link-disjoint paths $\mathbb{P}^*_i$ in $G'_{ij}$ between $s$ and $t_i$ of
       minimum total cost.
    7  Return $\mathbb{P}^*_i$ and $\mathbb{P}^*_j$.
```

Fig. 4. Algorithm MIN-LOCAL

that connect source s to terminals t_i and t_j, respectively. The algorithm finds two sets of h node-disjoint paths $\mathbb{P}^*_i$, $\mathbb{P}^*_j$ connecting s to t_i and s to t_j such that the subgraph G^*_{ij} of G' induced by paths in $\mathbb{P}^*_i \cup \mathbb{P}^*_j$ is minimal with respect to link removal. That is, removal of any link from G^*_{ij} results in a reduction of the value of the minimum cut between s and t_i or s and t_j. The formal description of Algorithm MIN-LOCAL appears in Fig. 4.

Lemma 5: Let G^*_{ij} be a subgraph of G induced by path sets $\mathbb{P}^*_i$ and $\mathbb{P}^*_j$ returned by Algorithm MIN-LOCAL and let $\mathbb{N}^* = \mathbb{N}^*(G^*_{ij}, s, \{t_i, t_j\}, h)$ be a coding network formed by G^*_{ij} and two terminals, t_i and t_j. Then, all links in $\mathbb{N}^* = \mathbb{N}^*(G^*_{ij}, s, \{t_i, t_j\}, h)$ are vital.

Proof: We start by showing that the links in $\mathbb{P}^*_j$ are vital. Consider graph G'_{ij} with link costs assigned at Step 2. We assume, by way of contradiction, that there exists a link $e \in \mathbb{P}^*_j$ which is not vital in $\mathbb{N}^*(G^*_{ij}, s, \{t_i . t_j\}, 2)$. Then, there exist h link-disjoint paths $\hat{\mathbb{P}}_j$ between s and t_j in G^*_{ij} that do not use link e. We denote by $\hat{f}$ and f^* the flows defined by sets of disjoint paths $\hat{\mathbb{P}}_j$ and $\mathbb{P}^*_j$, respectively. We denote by $G'_{ij}(f^*)$ the residual graph of G'_{ij} with respect to flow f^*. That is, $G'_{ij}(f^*)$ is formed from G'_{ij} by reversing links that belong to paths $\mathbb{P}^*_j$ and negating their cost. Note that the cost of every link $e \in G'_{ij}(f^*)$ is either 0 or -1. By the Augmenting Cycle Theorem [9, Chapter 3], flow $\hat{f}$ is equal to flow f^* plus flow along a set of directed cycles $\mathbb{C}$ in $G'_{ij}(f^*)$. Moreover, the cost of $\hat{f}$ equals to the cost of f^* plus the cost of flow on cycles in $\mathbb{C}$, where the cost of flow is defined with respect to costs assigned at Step 2. Since $\mathbb{P}^*_j$ is a minimum cost set of disjoint paths, the cost of flow $\hat{f}$ is greater or equal to that of f^*. This implies that all cycles in $\mathbb{C}$ contain only zero cost links, i.e., links whose originals belong to paths in $\mathbb{P}_i$.

Let C be a cycle in $\mathbb{C}$. Note that $\mathbb{C}$ must contain at least one cycle because $\hat{f}$ and f^* differ by at least one link. We classify the links in C into two categories: (1) the links that were reversed with respect to G'_{ij} (that is, whose origins in G'_{ij} belong to a path in $\mathbb{P}^*_j$) (2) the links that were not reversed. We observe that the degree of any node in G_{ij} (and thus in G'_{ij}) is at most 3 and consider the following cases.

1) The cycle C contains only links of type (1) or only links of type (2). Then, there is a cycle $C' \in G'_{ij}$ that corresponds to C. Such a cycle is either equal to C or obtained from C by reversing all its links. Since the total degree of each node in G'_{ij} is at most 3, C' belongs to a single path in $\mathbb{P}_i$. This contradicts the minimality of $\mathbb{P}_i$ (recall that these paths are chosen to be of minimum cost when all links in G were of unit cost).
2) The cycle C contains links of both types (1) and (2). Then, there exists a node $v \in C$ whose incoming link is of type (2) and whose outgoing link is of type (1). Thus, in G'_{ij} node v has two incoming links that belong to two different paths in $\mathbb{P}_i$. Such a node will also have two outgoing links, which contradicts the fact that the degree of any node in G'_{ij} is at most 3.

We have proven so far that every link in $\mathbb{P}^*_j$ is vital. To show that every link in $\mathbb{P}^*_i$ is also vital, notice that every link of cost 1 with respect to costs assigned at Step 5 is vital. Indeed, a non-vital link of cost one would contradict the minimality of $\mathbb{P}^*_i$. Hence the lemma follows. ■

It is not hard to verify that Algorithm MIN-LOCAL runs in time $O(|V|h^2)$ (again we use the augmenting path approach to find h link-disjoint paths).

D. *Algorithm* MIN-GLOBAL

Algorithm MIN-GLOBAL receives, as input, a feasible coding network $\mathbb{N}'(G', s, T, h)$. First, the algorithm iteratively constructs, for each $t_i \in T$, a set $\mathbb{P}_i$ of h link-disjoint paths between s and t_i. We denote by $E(\mathbb{P}_i)$ the set of links that belong to paths in $\mathbb{P}_i$, by $E_i = \cup_{j=1}^{i} E(\mathbb{P}_j)$, and by G^* the subgraph of G' induced by links in E_k. Our goal is to ensure that the total number of links in G^* which are incoming links of nodes of in-degree 2 or more is bounded by $O(h^3k^2)$. To that end, we first minimize the number of links in $E_i \setminus E_{i-1}$. In addition, we apply Algorithm MIN-LOCAL for $E(\mathbb{P}_j)$ and $E(\mathbb{P}_i)$, $1 \le j < i$, in order to further delete non-vital edges from E_i. The algorithm returns a coding network $\mathbb{N}^*(G^*, s, T, h)$. The formal description of Algorithm MIN-GLOBAL appears in Fig. 5.

Theorem 6: Let $\mathbb{N}^*(G^*(V^*, E^*), s, T, h)$ be the coding network returned by Algorithm MIN-GLOBAL($\mathbb{N}$). Let $\bar{V}^*$ be the subset of $V^* \setminus T$ that includes nodes of in-degree two or more and let $\bar{E}^*$ be the set of incoming links of nodes in $\bar{V}^*$. Then, it holds that $|\bar{E}^*| = O(h^3k^2)$.

Proof: We denote by $G_i(V_i, E_i)$ the subgraph of G' induced by links in E_i. We also denote by $\bar{V}_i$ the subset of $V_i \setminus T$ that includes nodes of in-degree two or more and by $\bar{E}_i$ the set of incoming links of nodes in $\bar{V}_i$. We prove, by induction on i, that $|\bar{E}_i|$ is bounded by $2h^3ki$.

For the base step, we note that $|\bar{E}_2|$ is bounded by $2h^3$. Indeed, Lemma 3, stated in Section II, implies that the subgraph $G_{1,2}$ induced by links in $E(\mathbb{P}_1) \cup E(\mathbb{P}_2)$ includes at most h^3 nodes of in-degree 2, each such node has at most two incoming links.

```
Algorithm MIN-GLOBAL ($\mathbb{N}'(G', s, T, h)$):
  Input:
      $\mathbb{N}'(G', s, T, h)$ - a feasible coding network;

  1   $\mathbb{P} \leftarrow \emptyset$.
  2   for $i \leftarrow 1$ to $k$ do
  3       Assign zero cost to all links in $E(\mathbb{P})$. Assign unit costs
          to all other links in $G'$.
  4       Find a set of $h$ link-disjoint paths $\mathbb{P}_i$ in $G'$ between $s$
          and $t_i$ of minimal total cost.
  5       if $i > 1$ do
  6           for $j \leftarrow 1$ to $i-1$ do
  7               $\mathbb{P}_i, \mathbb{P}_j \leftarrow$ MIN-LOCAL($\mathbb{N}'(G', s, T, h), \mathbb{P}_i, \mathbb{P}_j$)
  8       $\mathbb{P} \leftarrow \bigcup_{j=1}^{i} \mathbb{P}_j$ and $E_i \leftarrow \bigcup_{j=1}^{i} E(\mathbb{P}_j)$.
  9   $G^* \leftarrow$ a subgraph of $G'$ induced by links in $E_k$.
  10  Return $\mathbb{N}^*(G^*, s, T, h)$.
```

Fig. 5. Algorithm MIN-GLOBAL

For the induction step, we prove that for $i = 2, \ldots, k$ it holds that $|\bar{E}_i| \le 2h^3ki$. We divide the set $\bar{E}_i$ into two subsets $\bar{E}^1_i = \bar{E}_i \cap \bar{E}_{i-1}$ and $\bar{E}^2_i = \bar{E}_i \setminus \bar{E}^1_i$. By the inductive argument, the number of links in $\bar{E}^1_i$ is bounded by $2h^3k(i-1)$. Thus, in order to complete the proof we need to bound the number of links that belong to $\bar{E}^2_i$.

We denote by E_{ij} the set of incoming links of nodes of in-degree two in the subnetwork of G induced by links in $E(\mathbb{P}_i) \cup E(\mathbb{P}_j)$ in step 7. By Lemma 3, stated in Section II, each of the sets E_{ij}, $j \in \{1, \ldots, i-1\}$ contains at most $2h^3$ links each. We show that each link in $\bar{E}^2_i$ belongs to E_{ij} for some $j \in \{1, \ldots, i-1\}$, which, in turn, implies that $|\bar{E}^2_i| \le 2h^3k$ which concludes our assertion.

Let $e = (u, v)$ be a link in $\bar{E}^2_i$. We consider two cases.

1) Link $e = (u, v)$ belongs to $E_i \setminus E_{i-1}$. This implies that e belongs to $E(\mathbb{P}_i)$. In fact, e belongs to $E(\mathbb{P}_i)$ at any time during iteration i of the main loop (the loop that begins on line 2). Indeed, otherwise, there would exists a set of disjoint paths between s and t_i that has a smaller cost than that selected in line 4, resulting in a contradiction. Since the in-degree of v is at least two, v has an additional incoming link $e' = (w, v)$. Note that $e' \notin \mathbb{P}_i$ because $\mathbb{P}_i$ only contains node-disjoint paths. We conclude that e belongs to E_{ij} for some $j \in \{1, \ldots, i-1\}$.
2) Link $e = (u, v)$ belongs to E_{i-1}. This implies that in G_{i-1} node v has in-degree one. Since the in-degree of v in E_i is at least two, v has an additional incoming link $e' = (w, v)$. Such link must belong to $E_i \setminus E_{i-1}$, and, in turn to $E(\mathbb{P}_i)$ (due to the same argument as in case 1). This implies that e belongs to E_{ij} for some $j \in \{1, \ldots, i-1\}$.

■

Note that $\mathbb{N}^*(G^*, s, T, h)$ is a feasible network obtained from $\mathbb{N}'(G', s, T, h)$ by deleting redundant links. Thus,

$\mathbb{N}^*(G^*, s, T, h)$ is equivalent to $\mathbb{N}'(G', s, T, h)$, and, in turn, to $\mathbb{N}(G, s, T, h)$.

Algorithm MIN-GLOBAL invokes Algorithm MIN-LOCAL k^2 times, hence its running time is $O(|V|k^2h^2)$.

E. Procedure SHRINK

Procedure SHRINK receives as input the coding network $\mathbb{N}^*(G^*, s, T, h)$. The procedure forms an auxiliary network $\hat{\mathbb{N}}(\hat{G}, s, T, h)$ by repeatedly contracting nodes of total degree 2. Specifically, we remove every node $v \in G^*$ that has one incoming link (u, v) and one outgoing link (v, w) and substitute links (u, v) and (v, w) by a single link (u, w). By Theorem 6, the total number of links in $\hat{\mathbb{N}}(\hat{G}, s, T, h)$ is bounded by $O(h^3k^2)$. The computational complexity of Procedure SHRINK is $O(|V|)$.

F. Algorithm Analysis

We are ready to formally prove the correctness of our algorithm for finding a feasible network code and analyze its performance.

Theorem 7: Let $\mathbb{N}(G, s, T, h)$ be an acyclic coding network with unit capacity links. If $\mathbb{N}(G, s, T, h)$ is feasible, then there exists a deterministic algorithm that computes a network code $\mathbb{F}(\mathbb{N})$ for $\mathbb{N}$ in time $O(|E|kh + |V|k^2h^2 + h^4k^3(k + h))$. Moreover, the number of encoding nodes in $\mathbb{F}(\mathbb{N})$ is bounded by $O(h^3k^2)$.

Proof: We begin by observing that $\hat{\mathbb{N}}(\hat{G}, s, T, h)$ is a feasible coding network. Let $\hat{\mathbb{F}}(\hat{\mathbb{N}})$ be a feasible network code for $\hat{\mathbb{N}}$. The number of encoding nodes in $\hat{\mathbb{F}}$ is bounded by the number of nodes in $\hat{G}$ of in-degree two or more. Theorem 6 implies that the number of such nodes in G^* and, in turn, in $\hat{G}$ is at most $O(h^3k^2)$.

Let $\mathbb{N}^*(G^*, s, T, h)$ be the coding network formed by graph G^*. We construct a feasible network code $\mathbb{F}^*(\mathbb{N}^*)$ for $\mathbb{N}^*$ as follows. All nodes of G^* that belong to $\hat{G}$ have the same encoding function as in $\hat{\mathbb{F}}$. All other nodes just forward their incoming packets. Since all nodes in G^* that do not appear in $\hat{G}$ have one incoming link and one outgoing link, $\mathbb{F}^*(\mathbb{N}^*)$ is a feasible network code. Since G^* is a subgraph of G', $\mathbb{F}^*(\mathbb{N}^*)$ can be immediately extended into a feasible network code $\mathbb{F}'(\mathbb{N}')$ for $\mathbb{N}'(G', s, T, h)$. The number of encoding nodes in $\mathbb{F}'(\mathbb{N}')$ is at most $O(h^3k^2)$.

Next, we show how to construct a feasible network code for the original network $\mathbb{N}(G, s, T, h)$. Let $e = (v, u)$ be a link in G. Let $\{e'_1, \ldots, e'_{c_e}\}$ be the set of links in G' that correspond to e and let $f'(e'_i)$ be the encoding function of link e'_i in G'. If $e'_i \notin G'$, we $f'(e'_i)$ is equal to the zero element of Σ. Then the encoding function f_e of e is a composition of the encoding functions $\{f(e'_1), \ldots, f(e'_{c_e})\}$. The number of encoding nodes in $\mathbb{F}(\mathbb{N})$ is now bounded by the number of encoding nodes in $\mathbb{F}'(\mathbb{N}')$.

We proceed to determine the computational complexity of the algorithm. Recall that the running time of Procedure EXPAND is bounded by $O(|E|kh)$. The running time of Algorithm MIN-GLOBAL is $O(|V|k^2h^2)$. Since the graph $\hat{G}$ contains $O(h^3k^2)$ links, finding a feasible network code $\hat{\mathbb{F}}(\hat{\mathbb{N}})$ for $\hat{\mathbb{N}}(\hat{G}, s, T, h)$ requires $O(h^4k^3(k+h))$ (using the algorithm of [1]). We conclude that the total running time of the algorithm is bounded by $O(|E|kh + |V|k^2h^2 + h^4k^3(k + h))$. ■

For the special case of $h = 2$, the computational complexity of the algorithm can be improved by using the algorithm due to [10].

Corollary 8: Let $\mathbb{N}(G, s, T, h)$ be an acyclic coding network with links of integer capacity in which $h = 2$. If $\mathbb{N}(G, s, T, h)$ is feasible, then there exists a deterministic algorithm that computes a network code $\mathbb{F}(\mathbb{N})$ for $\mathbb{N}$ in time $O(|E| + |V|k^2 + k^4)$. Moreover, the number of encoding nodes in $\mathbb{F}(\mathbb{N})$ is bounded by $O(k^2)$.

Proof: In Procedure EXPAND, when finding $h = 2$ link disjoint paths between s and every terminal t_i we use the algorithm of [10] which preforms this task in time $O(|E|)$. ■

IV. MINIMIZING THE NUMBER OF ENCODING NODES

In this section we consider the problem of finding a network code with the minimum possible number of encoding nodes for both integer and fractional coding networks. We begin by defining information flow in a fractional coding network. Then, we show a relation between information flow and network codes in fractional coding networks. Finally, we present our results for both integer and fractional networks. We follow the definitions that appear in [8].

A. Fractional information flows

So far we considered integer coding networks $\mathbb{N}(G, s, T, h)$ that use h link-disjoint paths to deliver information between source s and each terminal $t_i \in T$. Fractional coding networks can use a set of paths between s and $t_i \in T$ which are not necessarily link-disjoint, each path delivers packets of a fractional size. For each $t_i \in T$ we denote by $\mathcal{P}_i$ the set of paths used to deliver information between s and t_i. Each path $P \in \mathcal{P}_i$ is associated with a weight $w(P)$ that specifies the size of a packet that can be sent over P. The set $\mathcal{P}_i$ is said to be *valid* if for every link $e \in G$ it holds that $\sum_{e \in P; P \in \mathcal{P}_i} w(P) \leq 1$. Menger's theorem [11] implies that if $\mathbb{N}(G, s, T, h)$ is feasible then there exists a valid path set $\mathcal{P}_i$ between s and t_i for every $t_i \in T$.

With the network coding approach paths that belong to different path sets in $\{\mathcal{P}_i\}_{i=1}^k$ can share a link or a portion of link capacity. In general, the capacity of a link $e \in G$ is divided between a number of subsets $\alpha_1, \alpha_2, \ldots, \alpha_x$ of T, such that the paths in path sets $\{\mathcal{P}_i \mid t_i \in \alpha_j\}$ share the portion of e's capacity allocated for α_j. Accordingly, for each link $e \in G$ we associate an *aggregation function* $x_e : \mathcal{T} \to R^+$, where $\mathcal{T}$ is the power set of T and R^+ is the set of non-negative real numbers. The function $x_e(\alpha)$ specifies the capacity allocated to the subset α of T. We refer to sets α_i for which $x_e(\alpha_i) > 0$ as *path aggregates*. We say that the set $X = \{x_e\}_{e \in G}$ of aggregation functions is consistent with path sets $\mathcal{P}_1, \ldots, \mathcal{P}_k$ if for all $e \in G$ it holds that $\sum_{\alpha \in \mathcal{T}} x_e(\alpha) \leq 1$ and

$$\forall e \in G,\ \forall t_i \in T \sum_{e\in P;\ P\in\mathcal{P}_i} w(P) = \sum_{t_i\in\alpha} x_e(\alpha).$$

Note that for given path sets $\mathcal{P}_1,\ldots,\mathcal{P}_k$ there may exist many sets of consistent aggregation functions.

Let $\mathcal{P}_1,\ldots,\mathcal{P}_k$ be valid paths sets and let X be a set of consistent aggregation functions. We divide the nodes of G into *path routing* nodes and *path mixing* nodes. A path routing node either preserves or splits incoming path aggregates. We represent the splitting of path aggregates by the function r defined below. Let $\mathcal{Q} \subseteq \mathcal{T} \times \mathcal{T}$ be the set of all pairs of disjoint sets in $\mathcal{T}$. A node v is said to be a path routing node with respect to X if there exists a function $r_v : \mathcal{Q} \rightarrow R^+$ such that for each $\alpha \in \mathcal{T}$ it holds that $\sum_{e\in d_v^{out}} x_e(\alpha) = \sum_{e\in d_v^{in}} x_e(\alpha) + \sum_{\{\alpha,\beta\}\in\mathcal{Q}} r_v(\{\alpha,\beta\}) - \sum_{\beta\cup\gamma=\alpha} r_v(\{\beta,\gamma\})$. Intuitively, if $r_v(\alpha,\beta) = x$ then path aggregate $\gamma = \alpha\cup\beta$ of value x is split into two path aggregates α and β.

Any node $v \in G$ which is not path routing is referred to as a path mixing node. Path mixing nodes can preserve, split, or combine path aggregates. The following theorem appears in a slightly modified form in [8]:

Theorem 9: Let $\mathbb{N}(G(V,E),s,T,h)$ be a coding network. Let V_1 and V_2 be a partition of V. Then, there exists a linear program with $O(|E|2^k)$ variables and coefficients in $\{-1,0,1\}$ which is feasible if and only if there exist feasible path sets $\mathcal{P}_1,\ldots,\mathcal{P}_{|T|}$ and a corresponding set of aggregation functions $X = \{x_e\}_{e\in G}$ in which only nodes in V_2 are path mixing nodes. Such $\{\mathcal{P}_i\}_{t_i\in T}$ and X are obtained as a solution to the linear program.

B. Fractional network codes

We begin with the definition of an m-fractional network code. Such a code partitions each of the h packets present at the source into m parts.

Definition 10 (m-fractional network code $\mathbb{F}(\mathbb{N}_m)$): For an integer m, an m-fractional code for $\mathbb{N}(G,s,T,h)$ is defined by an integral network code for $\mathbb{N}_m$= $\mathbb{N}(G_m,s,T,mh)$. Here G_m is the graph G in which each link e is replaced by m parallel links $\{e_1,\ldots,e_m\}$).

Note that a 1-fractional network code for $\mathbb{N}$ is an integral network code. The notions of encoding nodes and of the feasibility of $\mathbb{N}(G_m,s,T,mh)$ and $\mathbb{F}(\mathbb{N}_m)$ are defined similarly to that of integer coding networks (see Section II).

For a given instance $\mathbb{N}$=$\mathbb{N}(G,s,T,h)$ and an integer m, we denote by $Opt_m(\mathbb{N})$ the minimum number of encoding nodes in any feasible network code for $\mathbb{N}_{m'}$, where $m' \leq m$. We then define $Opt(\mathbb{N})$ to be $\min_m Opt_m(\mathbb{N})$ (the minimum exists as $Opt_m(\mathbb{N})$ is monotone in m and integral). The following theorem connects fractional information flows with fractional network coding and is sketched in [8].

Theorem 11: Let $\mathbb{N}(G,s,T,h)$ be a given network. Given an m-fractional feasible network code for $\mathbb{N}$ with Γ encoding nodes one can construct valid path sets $\mathcal{P}_1,\ldots,\mathcal{P}_{|T|}$ and a consistent set of aggregation functions $X = \{x_e\}_{e\in G}$ such that for each $t_i \in T$ the total weight of the paths $P \in \mathcal{P}_i$ is h and the number of path mixing nodes is bounded by Γ. Let $\mathcal{P}_1,\ldots,\mathcal{P}_{|T|}$ be valid path sets such that for each $t_i \in T$ the total weight of the paths $P \in \mathcal{P}_i$ is h, and such that each path in $\cup\mathcal{P}_i$ has weight which is a multiple of $\frac{1}{m}$. Let $X = \{x_e\}_{e\in G}$ be a corresponding consistent family of functions that have Γ path mixing nodes. Then, one can construct an m-fractional feasible network code for $\mathbb{N}$ with at most Γ encoding nodes. The reduction in both directions can be done in time which is polynomial in $|G|$, m, and 2^k.

C. Our results

We are now ready to state and prove our results.

Theorem 12 ([6], [12]): Computing $Opt_1(\mathbb{N})$ is $\mathcal{NP}$-hard for general networks $\mathbb{N}(G,s,T,h)$ in which $k = h = 2$.

Theorem 13: Computing $Opt(\mathbb{N})$ and $Opt_1(\mathbb{N})$ is $\mathcal{NP}$-hard even for acyclic networks $\mathbb{N}(G,s,T,h)$ in which either k or h is equal to 2.

Proof: We use a variant of the well know reduction from the minimum Set Cover (SC) problem. The input to the SC problem is a universe $U = (x_1,\ldots,x_n)$ of n elements and a set system $S = \{S_1,\ldots,S_m\}$; the objective is to find a minimum sized subset S' of S that *covers* all elements in U (namely each element $x \in U$ is in at least one set $S_i \in S'$). Consider the following *base graph* $G = (V,E)$ with a source node s, m intermediate nodes $\{S_1,\ldots,S_m\}$, and n leafs $\{x_1,\ldots,x_n\}$. We use the same notation for nodes and corresponding sets/elements throughout this proof. To avoid confusion, we will specify our exact meaning when needed. We add the links (s,S_j) for all sets S_j, and the links (S_j,x_i) iff $x_i \in S_j$.

In our reductions we use this base graph as a starting point, and enhance it with various nodes/links. We start with the case of $k = 2$. We add some nodes to G: t_1, t_2 (which will be our terminal nodes), and a new node s^*. We add the link (s,s^*). We partition each link (s,S_j) into a path of length four $(s,\alpha_j,\beta_j,\gamma_j,S_j)$. For t_1 we add the links (x_i,t_1) (for all elements x_i); the link (s^*,t_1); and the link (s,t_1). For t_2 we add the links (s^*,β_j), (γ_j,t_2), and (α_j,t_2) (for all j). The capacity of the links in our enhanced graph are either 1, n, $n(m-1)$ or nm. The links of capacity 1 are the links (S_j,x_i) (for all i, j), and the links (x_i,t_1). The links of capacity nm are the links (s,s^*) and (s^*,t_1). The link (s,t_1) is of capacity $n(m-1)$. The rest of the links are of capacity n. We now consider the network $\mathbb{N} = (G,s^*,\{t_1,t_2\},2nm)$. It is not hard to verify that $\mathbb{N}$ is feasible. We now prove that $Opt(\mathbb{N})$= k iff the minimum SC is of size k

First we note that any feasible set of paths $\mathbb{P}_2$ (for t_2) must consist of the paths (s,α_j,t_2) and $(s,s^*,\beta_j,\gamma_j,t_2)$. In addition, any feasible set of paths $\mathbb{P}_1$ for t_1 must include the path (s,s^*,t_1) of weight nm, the link (s,t_1) of weight $nm-n$, and a set of valid paths of total weight n that enter t_1 through the nodes x_i. Consider any path P that enters t_1 through node x_i. It must be of the form $(s,\alpha_j,\beta_j,\gamma_j,S_j,x_i,t_1)$ for some $x_i \in S_j$. Notice any such path P (of any weight) implies an encoding node (or more specifically a path mixing node) at β_j. Hence $Opt(\mathbb{N})$ is obtained when we design these paths to pass through as few as possible nodes β_j. If there is a set cover of size k (say by the sets $1, \ldots k$) then there is a set of valid paths of weight n from s^* to t_1 that only pass through β_j for $j \leq k$ and we have that $Opt(\mathbb{N}) \leq k$. In the other direction, if all the paths to t_1 of weight n (through x_i) pass through k nodes of β_j then there is a set cover of size at most k.

For the case $h = 2$ and arbitrary k we consider another variant of the base graph. We add some nodes to G. The terminal nodes will be $\hat{x}_1, \ldots, \hat{x}_n$, and $t_1, \ldots, t_m$. We also add a new node s^*. We add the link (s, s^*), and links $(x_i, \hat{x}_i)$. We partition each link (s, S_j) into a path of length four $(s, \alpha_j, \beta_j, \gamma_j, S_j)$. For terminals $\hat{x}_i$ we add the links $(s^*, \hat{x}_i)$. For terminals t_j we add the links (α_j, t_j) and the links (s^*, β_j), (γ_j, t_j). All capacities of the links in our enhanced graph are unit capacities. We now consider the network $\mathbb{N} = (G, s^*, \{\hat{x}_1, \ldots, \hat{x}_n; t_1, \ldots, t_m\}, 2)$. It is not hard to verify that $\mathbb{N}$ is feasible. We now prove that $Opt(\mathbb{N}) = k$ iff the minimum SC is of size k

First we note that any feasible set of paths $\mathbb{P}_{t_j}$ (for terminal t_j) must consist of the paths (s, α_j, t_j) and $(s, s^*, \beta_j, \gamma_j, t_j)$. In addition, any feasible set of paths $\mathbb{P}_{\hat{x}_i}$ for $\hat{x}_i$ must include the path $(s, s^*, \hat{x}_i)$ and a set of valid paths of weight 1 that enter $\hat{x}_i$ through the nodes S_j. Consider any path P that enters $\hat{x}_i$ through node S_j. It must be of the form $(s, \alpha_j, \beta_j, \gamma_j, S_j, x_i, \hat{x}_i)$. Notice any such path P (of any weight) implies a path mixing node at β_j. Hence as before $Opt(\mathbb{N})$ is obtained when we design these paths to pass through as few as possible nodes β_j. If there is a set cover of size k (say by the sets $1, \ldots k$) then there is a set of valid paths for each x_i that only pass through β_j for $j \leq k$ and we have that $Opt(\mathbb{N}) \leq k$. In the other direction, if all the set of paths corresponding to all x_i pass through k nodes β_j then there is a set cover of size at most k. ■

Theorem 14: For a given feasible acyclic network $\mathbb{N}(G, s, T, h)$, an integral network code with $Opt_1(\mathbb{N})$ encoding nodes can be found in time $n^{O(h^3k^2)}$.

Proof: In [6] it was shown that $\mathbb{N}$ has a network code with at most $O(h^3k^2)$ encoding nodes (alternatively one can use the results of Section III). More specifically, it was shown that for each terminal $t_i \in T$ there is a set of h link disjoint paths $\mathbb{P}_i$, such that for the subgraph G' of G consisting only of links in $\{\mathbb{P}_i\}$ it holds that (a) G' has as most $O(h^3k^2)$ nodes of in-degree larger than 1, (b) G' has as most $O(h^3k^2)$ nodes of out-degree larger than 1 and (c) the network $\mathbb{N}' = (G', s, T, h)$ is feasible.

Consider the graph G'. Let $\Gamma = \Gamma_{in} \cup \Gamma_{out}$ be the set of nodes in G' with in-degree (Γ_{in}) or out-degree (Γ_{out}) larger than 1 . The links of G' can be decomposed into a set of paths with endpoints in Γ. Denote this set of paths by $\mathbb{P} = \{P(u_j, v_j)\}_{j=1}^r$ where $r = O(h^3k^2)$, $u_j \in \Gamma_{out}$, $v_j \in \Gamma_{in}$, and $P(u_j, v_j)$ is a path between u_j and v_j that does not pass through any other node in Γ (the bound on r follows from the analysis in [6] or the analysis in the Section III). Consider the integer multicommodity flow problem Π on the original graph G in which we wish to route a unit of flow between each pair (u_j, v_j) above. Clearly Π is feasible (using $\mathbb{P} = \{P(u_j, v_j)\}_{j=1}^r$), and its solution implies a set of h link disjoint paths between s and each $t_i \in T$: $\{\mathbb{P}_i^*\}_{i=1}^k$. We now claim that one can construct the functions $X = \{x_e\}$ (as defined above) corresponding to $\{\mathbb{P}_i^*\}_{i=1}^k$, such that in X there are at most $|\Gamma_{in}|$ path mixing nodes. Indeed, for all pairs (u_j, v_j) and all $\alpha \in \mathcal{T}$ define $x_e(\alpha)$ to be constant along the links of the path connecting (u_j, v_j). More specifically, for each link $e \in G'$ there exists a subset α_e of $\mathcal{T}$ such that $x_e(\alpha_e) = 1$ and $x_e(\alpha) = 0$ for all $\alpha \neq \alpha_e$. The subset α_e is the set of indices i such that $\mathbb{P}_i^*$ passes through e. The existence of X in turn implies an integral network code with at most $|\Gamma_{in}|$ encoding nodes (Theorem 11).

The discussion above implies the following algorithm for computing $Opt(\mathbb{N}_1)$. For all subsets $\Gamma = \Gamma_{in} \cup \Gamma_{out}$ of V and all subsets A of $\Gamma_{in} \times \Gamma_{out}$ as defined above; define the integral multicommodity flow problem Π in which a unit of flow is to be routed between each pair in A. As G is acyclic, Π is solvable in time $n^{O(h^3k^2)}$ [13], and the solution to Π implies the functions $X = \{x_e\}$. If X implies a set of h link disjoint paths between s and each $t_i \in T$, then X implies a network code with at most $|\Gamma_{in}|$ encoding nodes. We now take $Opt(\mathbb{N}_1)$ to be the minimum value of $|\Gamma_{in}|$ over all choices of $\Gamma = \Gamma_{in} \cup \Gamma_{out}$ and A as defined above in which the solution to the corresponding Π implies h link disjoint paths between s and each $t_i \in T$. The total running time of our algorithm is $n^{O(h^3k^2)}$ as asserted. ■

Theorem 15: For a given feasible acyclic network $\mathbb{N}(G, s, T, h)$, an m-fractional network code with $Opt(\mathbb{N})$ encoding nodes can be constructed in time $n^{O(h^3k^2)}$.

Proof: In [6] it was shown that $\mathbb{N}$ has a network code with at most h^3k^2 encoding nodes. This implies the following procedure for constructing the asserted network code. For all subsets of nodes V' in G of size at most h^3k^2 construct and solve a linear program as in Theorem 9 in which $V_1 = V \setminus V'$ and $V_2 = V'$. If the linear program is feasible, one may construct a network code corresponding to its solution (Theorem 11). We return the feasible network code corresponding to the smallest set V'. The running time follows from Theorem 11. ■

Our results for various settings are summarized in Figure 1.

References

[1] 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 Transactions on Information Theory*, 51(6):1973–1982, June 2005.

[2] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung. Network Information Flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, 2000.

[3] S.-Y. R. Li, R. W. Yeung, and N. Cai. Linear Network Coding. *IEEE Transactions on Information Theory*, 49(2):371 – 381, 2003.

[4] R. Koetter and M. Medard. An Algebraic Approach to Network Coding. *IEEE/ACM Transactions on Networking*, 11(5):782 – 795, 2003.

[5] T. Ho, R. Koetter, M. Medard, D. Karger, and M. Effros. The Benefits of Coding over Routing in a Randomized Setting. In *Proceedings of the IEEE International Symposium on Information Theory*, 2003.

[6] M. Langberg, A. Sprintson, and J. Bruck. The Encoding Complexity of Network Coding. *To appear in the joint special issue of the IEEE Transactions on Information Theory and the IEEE/ACM Transactions on Networking on Networking and Information Theory*, 2006.

[7] N. J. A. Harvey, D. R. Karger, and K. Murota. Deterministic network coding by matrix completion. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, 2005.

[8] K. Bhattad, N. Ratnakar, R. Koetter, and K.R. Narayanan. Minimal network coding for multicast. In *Proceedings of the IEEE International Symposium on Information Theory*, 2005.

[9] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Networks Flows*. Prentice-Hall, NJ, USA, 1993.

[10] J. Suurballe and R. Tarjan. A Quick Method for Finding Shortest Pairs of Disjoint Paths. *Networks*, 14:325–336, 1984.

[11] K. Menger. Zur allgemeinen Kurventheorie. *Fund. Math*, 10:95–115, 1927.

[12] M. Mahdian and M. R. Salavatipour. Hardness and Approximation Results for Packing Steiner Trees Problems. *Lecture Notes in Computer Science*, 3221:181–191, 2004.

[13] S. Fortune, J. Hopcroft, and J. Wyllie. The Directed Subgraph Homeomorphism Problem. *Theoretical Computer Science*, 10(2):111–121, 1980.