The Encoding Complexity of Network Coding

Michael Langberg* Alexander Sprintson Jehoshua Bruck
California Institute of Technology
Email: {mikel,spalex,bruck}@caltech.edu

Abstract

In the multicast network coding problem, a source s needs to deliver h packets to a set of k terminals over an underlying network G. The nodes of the coding network can be broadly categorized into two groups. The first group incudes *encoding* nodes, i.e., nodes that generate new packets by combining data received from two or more incoming links. The second group includes *forwarding* nodes that can only duplicate and forward the incoming packets. Encoding nodes are, in general, more expensive due to the need to equip them with encoding capabilities. In addition, encoding nodes incur delay and increase the overall complexity of the network.

Accordingly, in this paper we study the design of multicast coding networks with a limited number of encoding nodes. We prove that in an acyclic coding network, the number of encoding nodes required to achieve the capacity of the network is bounded by h^3k^2. Namely, we present (efficiently constructible) network codes that achieve capacity in which the total number of encoding nodes is independent of the size of the network and is bounded by h^3k^2. We show that the number of encoding nodes may depend both on h and k as we present acyclic instances of the multicast network coding problem in which $\Omega(h^2k)$ encoding nodes are needed.

In the general case of coding networks with cycles, we show that the number of encoding nodes is limited by the size of *the feedback link set*, i.e., the minimum number of links that must be removed from the network in order to eliminate cycles. Specifically, we prove that the number of encoding nodes is bounded by $(2B+1)h^3k^2$, where B is the minimum size of the feedback link set. Finally, we observe that determining or even crudely approximating the minimum number of encoding nodes needed to achieve the capacity for a given instance of the network coding problem is $\mathcal{NP}$-hard.

1 Introduction

The goal of communication networks is to transfer information between source and destination nodes. Accordingly, the fundamental question that arises in network design is how to increase the amount of information transferred by the network. Recently, it has been shown that the ability of the network to transfer information can be significantly improved by employing the novel technique of *network coding* [1–3]. The idea is to allow the intermediate network nodes to combine data received over different incoming links. Nodes with coding capabilities are referred to as *encoding* nodes, in contrast to *forwarding* nodes that can only forward and duplicate incoming packets. The network coding approach extends traditional routing schemes, which include only forwarding nodes.

The concept of network coding was introduced in a seminal paper by Ahlswede et. al. [1] and immediately attracted a significant amount of attention from the research community. A large body of research focused on the multicast network coding problem where a source s needs to deliver h packets to a set of k terminals T over an underlying network G. It was shown in [1] that the capacity of the network, i.e., the maximum number of packets that can be sent between s and T, is bounded by the size of the minimum *cut*[1] that separates the source s and a

*Research supported in part by NSF grant CCF-0346991

[1]A cut (V_1, V_2) in graph $G(V, E)$ is a partition of G into two subsets V_1 and $V_2 = V \setminus V_1$. The size of the cut is determined by the number of links that leave a node in V_1 and enter a node in V_2. We say that a cut (V_1, V_2) separates node s and t if $s \in V_1$ and $t \in V_2$.

terminal $t \in T$. Namely, a source s can transmit at capacity h to a set of terminals only if the size of the minimum cut separating s and any one of the terminals is at least h. This combinatorial condition was shown to be sufficient by Li, Yeung and Cai [2], and achievable by using a linear network code, i.e., a code in which each packet sent over the network is a linear combination of the original packets. In a subsequent work, Koetter and Médard [3] 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. [4] to show that linear network codes can be efficiently constructed by employing a randomized algorithm. Jaggi et al. [5] proposed a deterministic polynomial-time algorithm for finding a feasible network code for a given multicast network.

Previous work on network coding establishes a tight upper bound on the capacity of multicast networks and provides tools that enable to achieve this capacity. However, optimization issues in network coding have received little attention from the research community. In general, the goal of network optimization is to minimize the amount of resources consumed by network connections. In this study we focus on minimizing the total number of encoding nodes in multicast coding networks. More specifically, given a communication network G, a source node s, a set of terminals T, and a required number of packets, our goal is to find a feasible network code with as few encoding nodes as possible.

The problem of minimizing the amount of encoding nodes is important for both theoretical and practical reasons. First, encoding nodes in a network are, in general, more expensive than forwarding nodes, mostly because of the need to equip these nodes with coding capabilities. In addition, encoding nodes may incur delay and increase the overall complexity of the network.

Contribution

The contribution of our paper can be summarized as follows. We prove that to enable transmission at rate h from a source s to k terminals over an acyclic graph, one can efficiently construct network codes in which the number of encoding nodes is independent of the size of the underlying graph G and depends only on h and k. Our construction procedure is very simple and involves three steps: (1) Transform the original network into one which is minimal with respect to link removal and in which the degree of each internal node is at most three; (2) Find any feasible network code for the reduced network; (3) Reconstruct a network code for the original network. We show that such a procedure yields codes with only h^3k^2 encoding nodes. We show that in the worst case the number of encoding nodes depends on both h and k. To that end, we present, for any values of h and k, a coding network that requires $\Omega(h^2k)$ encoding nodes.

We also consider the general case of coding networks with cycles. We show that in such networks, the number of encoding nodes needed to enable transmission at capacity h from a source s to k terminals depends on the size of *the feedback link set* of the network, i.e., the minimum number of links that must be removed from the network in order to eliminate cycles. Specifically, we prove that the number of encoding nodes needed is bounded by $(2B + 1)h^3k^2$, where B is the minimum size of the feedback link set. We also present a lower bound of $\Omega(hB)$ on the number of encoding nodes in a network with cycles.

Finally, we consider the problem of finding a network code that enables transmission at capacity h from a source s to k terminals with a *minimum* number of encoding nodes. We observe that determining, or even crudely approximating, the minimum number of encoding nodes needed to achieve capacity is $\mathcal{NP}$-hard.

Encoding links

A more accurate estimation of the total amount of computation performed by a coding network can be obtained by counting *encoding links*, rather than encoding nodes. A link (v, u), $v \notin s$, is referred to as an encoding link if each packet sent on this link is a combination of two or more packets received through the incoming links of v. Indeed, as the output degrees of nodes in G may vary, each encoding node might have different computation load. In addition,

only some of the outgoing links of a node v can be encoding, while other outgoing links of v merely forward packets that arrive on v. Accordingly, we can consider the problem of finding a feasible network code that minimizes the total number of encoding links. It turns out that all upper and lower bounds on the minimum number of encoding nodes we present, as well as our inapproximability results, carry over to the case in which we want to minimize the number of encoding links. This follows from the fact that all our results are derived by studying networks in which internal nodes are of total degree three. In such networks, the number of encoding links is equal to the number of encoding nodes. For the remainder of this paper we state our results in terms of encoding nodes.

Related work

The problem of minimizing the number of encoding nodes in a network code is partially addressed in the works of Fragouli et al. [6, 7] and Tavory et al. [8]. The works of Fragouli et al. study the special case in which the given network is acyclic and one is required to transmit two packets from the source to a set of terminals of size k. For this specific case (i.e., $h = 2$) they show that the required number of encoding nodes is bounded by k. The proof techniques used in [6, 7] rely on a certain combinatorial decomposition of the underlying network and seem difficult to generalize for the case in which the number of packets h is larger than two.

The problem of minimizing the number of encoding nodes in a network code is also studied by Tavory et al. [8]. They obtain partial results of nature similar to those of [6] and [7], mentioned above. Namely, they are able to prove, for the case $h = 2$, that the number of required encoding nodes is independent of the size of the underlying graph G. For general values of h, [8] presents several observations which lead to the conjecture that the number of encoding nodes needed to enable transmission at capacity h from a source s to k terminals over an acyclic graph, is independent of the size of the underlying graph. In our study we prove this conjecture.

Finally, encoding vs. forwarding nodes in the solution of network coding problems was also studied by Wu et al. [9]. Wu et al. do not consider the amount of encoding nodes in a given network code. Rather, they show the existence (and efficient construction) of network codes in which only nodes which are not directly connected to a terminal perform encoding. The results in [9] do not imply bounds on the number of encoding nodes needed in communicating over a network.

Organization

The rest of the paper is organized as follows. In Section 2, we define the model of communication and state our results in detail. In Section 3 we define the notion of a *simple* network, and show that it suffices to analyze such networks to obtain our results. This section also includes the description of our algorithm for finding network codes with *a bounded number of* encoding nodes. In Section 4, we establish our upper bound for acyclic networks. In Section 5, we establish an upper bound on the minimum number of encoding nodes in general (cyclic) networks. In Section 6, we present our *negative* results. Namely, we present lower bounds for both acyclic and cyclic networks, and we show that determining (or even approximating) the minimum number of encoding nodes needed to achieve capacity is $\mathcal{NP}$-hard. Finally, in Section 7, we conclude with a few remarks and open problems.

2 Model

The communication network is represented by a directed graph $G = (V, E)$ where V is the set of nodes in G) and E the set of links. We assume that each link $e \in E$ can transmit one packet per time unit. In order to model links whose capacity is higher than one unit, G may include multiple parallel links. An instance $\mathbb{N}(G, s, T, h)$ of the network coding problem is a 4-tuple that includes the graph G, a source node $s \in V$, a set of terminals T, and the number of

packets h that must be transmitted from the source node s to every terminal $t \in T$. We assume that each packet is a symbol of some alphabet Σ.

Definition 1 (Network code $\mathbb{F}(\mathbb{N})$) *A network code for $\mathbb{N}(G, s, T, h)$ is defined by functions $\mathbb{F}(\mathbb{N}) = \{f_e \mid e \in E\}$. For links $e = (s, u)$ leaving the source, $f_e : \Sigma^h \to \Sigma$. For other links $e = (v, u)$, $f_e : \Sigma^{d_{in}(v)} \to \Sigma$. Here, $d_{in}(v)$ is the in-degree of node v.*

The function $f_{(v,u)}$ specifies the packet transmitted on link (v, u) for any possible combination of packets transmitted on the incoming links of v. For links leaving the source, f_e takes as input the h packets available at a source.

Definition 2 (Encoding and forwarding links and nodes) *For a network code $\mathbb{F}(\mathbb{N})$, e is referred to as an* encoding link*, if it has a corresponding function f_e that depends on two variables or more. If f_e depends on a single variable, we refer to e as a* forwarding 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*.*

Note, that there may be links e for which the function f_e depends on a single variable, but $f_e(x) \neq x$. We refer to such links as forwarding nevertheless, and do not count them as encoding links. It is not hard to verify that in the case that $\mathbb{F}(\mathbb{N})$ includes links with corresponding functions f_e that depend on a single variable but are not the identity function, one can construct a new network code $\mathbb{F}'(\mathbb{N})$ without such functions such that the number of encoding links in $\mathbb{F}'(\mathbb{N})$ and $\mathbb{F}(\mathbb{N})$ are equal.

The capacity of a multicast coding network is determined by the minimum size of a cut that separates the source s and any terminal $t \in T$ [2]. An instance $\mathbb{N}(G, s, T, h)$ of the network coding problem is said to be *feasible* if and only if the size of each such cut is at least h. Let $\mathbb{N}(G, s, T, h)$ be a feasible network. A network code $\{f_{(v,u)} \mid (v, u) \in E\}$ for $\mathbb{N}$ is said to be feasible if it *allows communication at rate h* between s and each terminal $t \in T$. For acyclic networks, a network code is said to allow communication at rate h if each terminal $t \in T$ can compute the original h packets available at the source from the packets received via its incoming links. To define the notion of rate for cyclic networks, one must take into consideration multiple rounds of transmission (in which h packets are sent from the source s over the network in each round), and require that over time each terminal $t \in T$ can compute the original packets available at the source from the packets received via its incoming links. In both the cyclic and acyclic case, if $\mathbb{N}(G, s, T, h)$ is feasible then there exists a feasible network code for $\mathbb{N}$ [2].

2.1 Statement of results

As mentioned in the introduction, our goal is to find feasible network codes with a minimum number of encoding nodes. For a given instance $\mathbb{N}(G, s, T, h)$ of the network coding problem, we denote by $Opt(\mathbb{N})$ the minimum number of encoding nodes in any feasible network code for $\mathbb{N}(G, s, T, h)$.

We show that computing $Opt(\mathbb{N})$ for a given instance $\mathbb{N}(G, s, T, h)$ of a network coding problem is an $\mathcal{NP}$-hard problem. Furthermore, it is $\mathcal{NP}$-hard to approximate $Opt(\mathbb{N})$ within any multiplicative factor or within an additive[2] factor significantly less than $|V|$. This result follows from the fact that it is $\mathcal{NP}$-hard to distinguish between instances $\mathbb{N}$ in which $Opt(\mathbb{N}) = 0$ and instances in which $Opt(\mathbb{N}) > 0$.

Theorem 3 *Let $\varepsilon > 0$ be any constant. Let $\mathbb{N}(G, s, T, h)$ be an instance of the multicast network coding problem in which the underlying graph has $|V|$ nodes. Approximating the value of $Opt(\mathbb{N})$ within any multiplicative factor or within an additive factor of $|V|^{1-\varepsilon}$ is $\mathcal{NP}$-hard.*

[2] An estimate to $Opt(\mathbb{N})$ which is within an α-multiplicative factor of $Opt(\mathbb{N})$ is referred to as an α-multiplicative approximation of $Opt(\mathbb{N})$. An estimate to $Opt(\mathbb{N})$ which is within an α-additive value of $Opt(\mathbb{N})$ is referred to as an α-additive approximation of $Opt(\mathbb{N})$.

Although the problem of finding the exact or approximate value of $Opt(\mathbb{N})$ is $\mathcal{NP}$-hard, we upper bounds on $Opt(\mathbb{N})$ that hold for any instance $\mathbb{N}(G, s, T, h)$ of the multicast network coding problem. The main contribution of our paper is an upper bound on $Opt(\mathbb{N})$ which is independent of the size of the network and depends on h and $|T| = k$ only. Specifically, we show that $Opt(\mathbb{N}) \leq h^3k^2$ for any acyclic coding network $\mathbb{N}$ that delivers h packets to k terminals. Our bound is constructive, i.e., for any feasible instance $\mathbb{N}(G, s, T, h)$ we present an *efficient* procedure that constructs a network code with at most h^3k^2 encoding nodes. In what follows, an algorithm is said to be efficient if its running time is polynomial in the size of the underlying graph G.

Theorem 4 (Upper bound, acyclic networks) *Let G be an acyclic graph and let $\mathbb{N}(G, s, T, h)$ be a feasible instance of the multicast network coding problem. Then, one can efficiently find a feasible network code for $\mathbb{N}$ with at most h^3k^2 encoding nodes, i.e., $Opt(\mathbb{N}) \leq h^3k^2$, where $k = |T|$.*

Theorem 5 (Lower bound, acyclic networks) *Let r_1 and r_2 be arbitrary integers. Then, there exist instances $\mathbb{N}(G, s, T, h)$ of the network coding problem such that $h \geq r_1$, $|T| = k \geq r_2$, the underlying graph G is acyclic, and $Opt(\mathbb{N}) \geq \Omega(h^2k)$.*

Finally, we establish upper and lower bounds on the number of encoding nodes in the general setting of communication networks with cycles. We show that the value of $Opt(\mathbb{N})$ in a cyclic network $\mathbb{N}$ depends on the size of the *minimum feedback link set*.

Definition 6 (Minimum feedback link set [10]) *Let $G(V, E)$ be a directed graph. A subset $\hat{E} \subseteq E$ is referred to as a* feedback link set *if the graph G' formed from G by removing all links in $\hat{E}$ is acyclic. A feedback link set of minimum size is referred to as the* minimum feedback link set. *Given a network $\mathbb{N}(G, s, T, h)$, we denote by $B(\mathbb{N})$ the minimum feedback link set of its underlying graph G.*

Theorem 7 (Upper bound, cyclic networks) *Let $\mathbb{N}(G, s, T, h)$ be an instance of the multicast network coding problem. Then, one can efficiently find a feasible network code for $\mathbb{N}$ with at most $(2B(\mathbb{N}) + 1)h^3k^2$ encoding nodes, i.e., $Opt(\mathbb{N}) \leq (2B(\mathbb{N}) + 1)h^3k^2$, where $k = |T|$.*

Theorem 8 (Lower bound, cyclic networks) *Let r_1 and r_2 be arbitrary integers. Then (a) there exists instances $\mathbb{N}(G, s, T, h)$ of the multicast network coding problem such that $B(\mathbb{N}) \geq r_1$, $h \geq r_2$ and $Opt(\mathbb{N}) \geq \Omega(B(\mathbb{N})h)$; (b) there exist instances $\mathbb{N}(G, s, T, h)$ of the multicast network coding problem such that $|V| \geq r_1$, $h = |T| = 2$, and $Opt(\mathbb{N}) = \frac{|V|-5}{2}$ (here V is the set of nodes in G).*

A couple of remarks are in place. First, note that Theorem 7 generalizes Theorem 4, as for acyclic networks the minimum feedback link set is of size 0. Second, note that Theorem 8 establishes two lower bounds. The first complements the upper bound of Theorem 7, while the second shows that in the case of cyclic networks the value of $Opt(\mathbb{N})$ is not necessarily independent of the size of the network and may depend linearly on the number of nodes in G.

3 "Simple" coding networks

Let $\mathbb{N}(G, s, T, h)$ be a feasible instance to the network coding problem. In order to establish a constructive upper bound on the minimal number of encoding nodes $Opt(\mathbb{N})$ of $\mathbb{N}$ we consider a special family of feasible networks, referred to as *simple networks*. In what follows we define simple coding networks, and show that finding network codes with few encoding nodes for this family of restricted networks suffices to prove Theorems 4 and 7. We start by defining feasible instances which are minimal with respect to link removal.

Definition 9 (Minimal Instance) *A feasible instance of the network coding problem $\mathbb{N}(G, s, T, h)$ is said to be* minimal with respect to link removal *if any instance $\hat{\mathbb{N}}(\hat{G}, s, T, h)$ formed from G by deleting a link e from G is no longer feasible.*

Definition 10 (Simple instance) *Let $\mathbb{N}(G, s, T, h)$ be an instance of the network coding problem. $\mathbb{N}(G, s, T, h)$ is said to be simple iff (a) $\mathbb{N}$ is feasible; (b) $\mathbb{N}$ is minimal with respect to link removal; (c) the total degree of each node in G is at most 3 (excluding the source and terminal nodes); and (d) the terminal nodes T have no outgoing links.*

We now present our reduction between general and simple networks.

3.1 Reduction to simple networks

Let $\mathbb{N}(G, s, T, h)$ be a feasible instance of the network coding problem. We construct a simple instance $\hat{\mathbb{N}}(\hat{G}, s, \hat{T}, h)$ corresponding to $\mathbb{N}$. The simple instance $\hat{\mathbb{N}}$ we construct corresponds to $\mathbb{N}$ in the sense that any feasible network code for $\hat{\mathbb{N}}$ yields a network code for $\mathbb{N}$ which includes the same or a smaller amount of encoding nodes. Our construction is computationally efficient and includes the three following steps.

Step 1: **Replacing terminals.** Let $T_1 \subseteq T$ be the set of terminals whose out-degree is not zero. For each terminal $t \in T_1$ we replace t by adding a new node t' to G and connecting t and t' by h parallel links. We denote the resulting set of terminals by $\hat{T}$, the resulting graph by G_1, and the resulting coding network by $\mathbb{N}_1(G_1, s, \hat{T}, h)$.

Step 2: **Reducing degrees.** Let G_2 be the graph formed from G_1 by replacing each node $v \in G_1, v \neq s, v \notin T_1$ whose degree is more than 3 by a subgraph Γ_v, constructed as follows. Let $\{x_i \mid i = 1, \ldots, d_{in}(v)\}$ and $\{y_i \mid i = 1, \ldots, d_{out}(v)\}$ be the incoming and outgoing links of v, respectively, where $d_{in}(v)$ and $d_{out}(v)$ are the in- and out- degrees of v. For each incoming link x_i, we construct a binary tree X_i that has a single incoming link x_i and $d_{out}(v)$ outgoing links $e_{i1}, \ldots, e_{id_{out}(v)}$ (with one or two links leaving each leaf). Similarly, for each outgoing link y_j we construct an inverted binary tree Y_j that has a single outgoing link y_j and $d_{in}(v)$ incoming links $e_{1j}, \ldots, e_{d_{in}(v)j}$ (again, with one or two links entering each leaf). Fig. 1 demonstrates the construction of the subgraph Γ_v for a node v with $d_{in}(v) = d_{out}(v) = 4$. Note that for any two links x_i and y_j there is a path in Γ_v that connects x_i and y_j. The resulting coding network is denoted by $\mathbb{N}_2(G_2, s, \hat{T}, h)$.

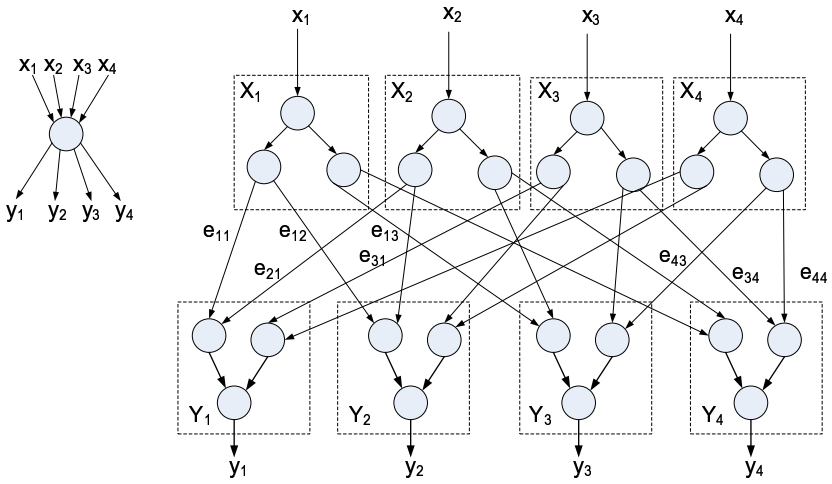

Figure 1: Substituting a node v by a gadget Γ_v.

Step 3: **Removing links.** Let $\hat{G}$ be *any* subgraph of G_2 such that $\hat{\mathbb{N}}(\hat{G}, s, \hat{T}, h)$ is minimal with respect to link removal. The graph G_2 can be efficiently computed by employing the following greedy approach. For each link $e \in G_2$, in an arbitrary order, we check whether removal of e from G_2 would result in a violation of the min-cut condition. The min-cut condition can be easily checked by finding h link-disjoint paths between s and each terminal $t \in \hat{T}$ (via max-flow techniques, e.g., [11]). All links whose removal does not result in a violation of the min-cut condition are removed from G_2. The resulting coding network, denoted by $\hat{\mathbb{N}}(\hat{G}, s, \hat{T}, h)$, is the final outcome of our reduction.

We proceed to analyze the properties of $\hat{\mathbb{N}}$. First, we show that $\hat{\mathbb{N}}$ is a simple network. Note that each step of our construction maintains the min-cut condition. Hence, the size of the minimum cut in $\hat{\mathbb{N}}$ between s and any terminal $t \in \hat{T}$ is at least h, which, in turn, implies, that $\hat{\mathbb{N}}$ is a feasible instance. Due to Step 1, the terminals $\hat{T}$ have no outgoing links. Finally, steps 2 and 3 ensure that the total degree of any node in $\hat{G}$ excluding the source and terminals is at most 3 and that $\hat{\mathbb{N}}$ is minimal with respect to link removal.

Second, we observe that the minimum size of a feedback link set in G is smaller or equal to that of $\hat{G}(\hat{V}, \hat{E})$. Indeed, it is easy to verify that if $\mathbb{B}$ is a feedback link set of G, then $\hat{\mathbb{B}} = \mathbb{B} \cap \hat{E}$ is a feedback link set of $\hat{G}$.

Finally, we show that $Opt(\mathbb{N}) \leq Opt(\hat{\mathbb{N}})$. That is, any feasible network code for $\hat{\mathbb{N}}$ with ℓ encoding nodes can be used to efficiently construct a network code for $\mathbb{N}$ with at most ℓ encoding nodes.

Reconstruction of feasible network codes for $\mathbb{N}$: Let $\hat{\mathbb{F}}(\hat{\mathbb{N}})$ be a feasible network code for $\hat{\mathbb{N}}(\hat{G}, s, \hat{T}, h)$ with ℓ encoding nodes. A feasible network code $\mathbb{F}(\mathbb{N})$ for $\mathbb{N}(G, s, T, h)$ is constructed as follows. Let $e = (v, u)$ be a link in G. Let e' be the corresponding link between Γ_v and Γ_u in $\mathbb{N}_2$ (recall that $\mathbb{N}_2$ was constructed at Step 2 above). If e' does not appear in $\hat{\mathbb{N}}$ then no information is sent over the link e in $\mathbb{F}(\mathbb{N})$. If e' appears in $\hat{\mathbb{N}}$, the code f_e for $e = (v, u)$ is determined by the codes $f_{\hat{e}} \in \hat{\mathbb{F}}(\hat{\mathbb{N}})$ of links $\hat{e}$ that belong to Γ_v. Specifically, let $X = \{x_1, \ldots, x_{d_{in}(v)}\}$ be the incoming links of Γ_v where $d_{in}(v)$ is the in-degree of v in $\hat{G}$. The construction of $\mathbb{N}_2$ implies that the packet transmitted on the link e' is a function f_e of the packets transmitted on links X. We use this function as an encoding function for link e in code $\mathbb{F}(\mathbb{N})$. The fact that the incoming links of v in G correspond to the links in X implies the feasibility of the resulting code $\mathbb{F}(\mathbb{N})$.

We note that a node $v \in \mathbb{F}(\mathbb{N})$ is an encoding node if and only if at least one of the nodes in Γ_v performs encoding. On the other hand, each encoding node in $\hat{\mathbb{F}}(\hat{\mathbb{N}})$ corresponds to at most one encoding node in $\mathbb{F}(\mathbb{N})$. This implies that the number of encoding nodes in $\mathbb{F}(\mathbb{N})$ is at most ℓ and, in turn, $Opt(\mathbb{N}) \leq Opt(\hat{\mathbb{N}})$.

We summarize the above discussion by the following lemma:

Lemma 11 *Let $\mathbb{N}(G, s, T, h)$ be a feasible instance of the network coding problem. Then, one can efficiently construct a simple instance $\hat{\mathbb{N}}(\hat{G}, \hat{s}, \hat{T}, h)$ for which (a) $|\hat{T}| = |T|$, (b) the size of the feedback link set of $\hat{\mathbb{N}}$ is less than or equal to that of $\mathbb{N}$, and (c) any feasible network code for $\hat{\mathbb{N}}$ with ℓ encoding nodes can be used to efficiently construct a feasible network code for $\mathbb{N}$ with at most ℓ encoding nodes.*

3.2 The value of $Opt(\mathbb{N})$ in simple instances

In what follows we show that for simple instances $\mathbb{N}(G, s, T, h)$ the value of $Opt(\mathbb{N})$ is equal to the number of nodes in G (excluding the terminals) with in-degree 2.

Definition 12 *Let $\mathbb{N}(G, s, T, h)$ be a simple instance of the network coding problem. We define $\alpha(\mathbb{N})$ to be the number of nodes in $G \setminus T$ of in-degree 2.*

Lemma 13 *Let $\mathbb{N}(G, s, T, h)$ be a simple instance of the network coding problem. Let $\mathbb{F}(\mathbb{N})$ be any feasible network code for $\mathbb{N}$. Then, a node $v \in G$, $v \neq s$, $v \notin T$, is an encoding node in $\mathbb{F}(\mathbb{N})$ if and only if the in-degree of v is 2. Thus, $Opt(\mathbb{N}) = \alpha(\mathbb{N})$.*

Proof: Let v be an encoding node in $\mathbb{F}(\mathbb{N})$. Then, the in-degree of v must be larger than one, otherwise all coding functions f_e of the outgoing links of v depend on a single variable, which contradicts the fact that v is an encoding node. Furthermore, since the total degree of v is at most three, and since it has at least one outgoing link, it follows that the in-degree of v is 2.

Now let v, $v \neq s$, $v \notin T$, be a node with in-degree 2, and let $e(v,u)$ be an outgoing link of v (node v must have an outgoing link, otherwise $\mathbb{N}$ is not minimal). If e is not an encoding link, then f_e is a function of a single variable, i.e., f_e depends on packets that arriving on one of the incoming links of v. This implies that the other incoming link can be omitted from G (while preserving the feasibility of $\mathbb{N}(G,s,T,h)$), which contradicts the minimality of $\mathbb{N}$. ■

3.3 The algorithm

Lemma 13 implies that for any given simple network $\mathbb{N}(G,s,T,h)$, any upper bound on $\alpha(\mathbb{N})$ is also an (efficiently constructible) upper bound on $Opt(\mathbb{N})$. Accordingly, in Sections 4 and 5 we prove that $\alpha(\mathbb{N})$ is bounded by h^3k^2 for acyclic instances and $h^3k^2(2B+1)$ for cyclic instances, where B is the size of the minimum feedback link set of G and $k=|T|$.

This leads to the following efficient procedure for finding a feasible network code with a bounded number of encoding nodes (implied by Lemma 11). The procedure works for a general (not necessarily simple) instance of the network coding problem $\mathbb{N}(G,s,T,h)$. We begin by transforming $\mathbb{N}$ into a simple network $\hat{\mathbb{N}}$. Then, we find any feasible network code for $\hat{\mathbb{N}}$. Finally, we reconstruct the corresponding network code for the original network $\mathbb{N}$. The description of our procedure appears in Figure 2.

Algorithm($\mathbb{N}$)

Input: Network $\mathbb{N}$.

1. Transform $\mathbb{N}$ into a simple network $\hat{\mathbb{N}}$ as described in Section 3.1.
2. Find **any** feasible network code for $\hat{\mathbb{N}}$, e.g., by using algorithms appearing in [4, 5].
3. Reconstruct the corresponding network code for $\mathbb{N}$ as described in Section 3.1.

Figure 2: Algorithm for finding a network code with a bounded number of encoding nodes.

4 Upper Bound for acyclic networks

In this section we present our proof of Theorem 4. The proof includes two steps. First, in Section 4.1, we analyze the special case in which the number of terminals in the network coding instance is 2 (i.e., $k=2$). Then, in Section 4.2, we address the general case in which the number of terminals is arbitrary. In both cases we establish an upper bound on $\alpha(\mathbb{N})$ for simple acyclic networks $\mathbb{N}$. By Lemmas 11 and 13, the upper bound for $\alpha(\mathbb{N})$ suffices to prove Theorem 4.

4.1 Networks with two terminals

Throughout this section we will use the following theorem implied by the results of [1] and [2] combined with Menger's Theorem [12]:

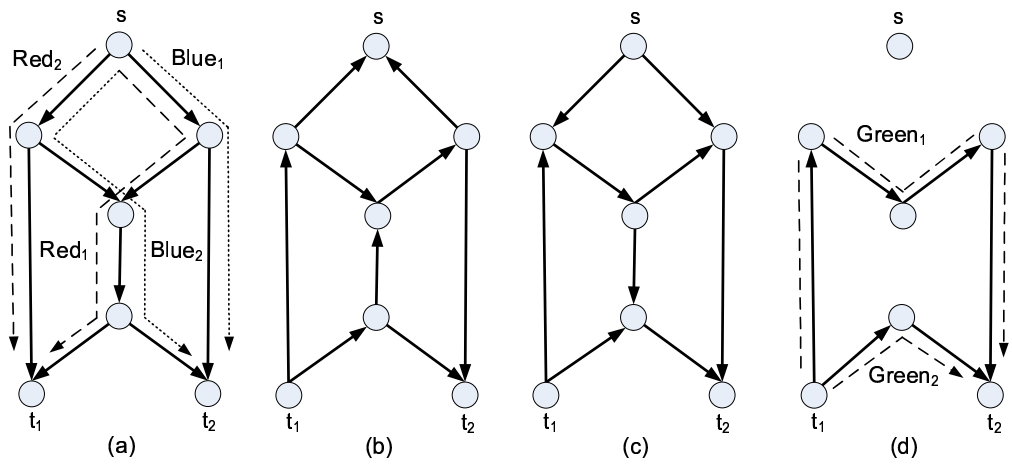

Figure 3: (a) A simple instance $\mathbb{N}(G, s, T, h)$ with corresponding red and blue paths. (b) The corresponding residual graph G_1. (c) The corresponding residual graph G_2. (d) The corresponding auxiliary graph $\hat{G}$ consisting of green paths.

Theorem 14 ([1, 2, 12]) *Let $\mathbb{N}(G, s, T, h)$ be an instance of the network coding problem. Then, there exists a feasible network code $\mathbb{F}(\mathbb{N})$ if and only if for each terminal $t_i \in T$ there exist h link-disjoint paths that connect s and t_i.*

We begin by introducing the concept of *residual graphs*. The residual graphs capture the minimality of the instance at hand with respect to link removal. In particular, the residual graphs that correspond to minimal instances contain no cycles.

4.1.1 Residual graphs and link-disjoint paths

Let $\mathbb{N}(G, s, T, h)$ be a simple (and therefore feasible) instance of the network coding problem with two destinations t_1 and t_2. By Theorem 14, there exist h link-disjoint paths that connect s and terminal t_1. Throughout this section, we fix one set of these paths and refer to them as *red* paths. Similarly, there exist h link-disjoint paths between s and t_2. We fix one set of these paths and refer to them as *blue* paths. We refer to links that belong to red paths as red links, and links that belong to blue paths as blue links. As a link in G can belong to a red path and to a blue path, it can be both red and blue.

We observe that a simple network $\mathbb{N}(G, s, T, h)$ has the following properties. First, every link in G belongs to either a blue or a red path. Second, at most one red (blue) path may pass through any node v in G which is not a terminal or the source. Third, all links entering t_1 are exclusively red, and all links entering t_2 are exclusively blue. Finally, the source node s has no incoming links. The first and final properties follow from the minimality of $\mathbb{N}(G, s, T, h)$. The second property follows from the fact that the degree of each node which is not a terminal or a source is at most 3. The third property follows from the fact that the terminals have no outgoing links and the fact that the red (blue) paths must terminate at t_1 (t_2).

A simple instance $\mathbb{N}(G, s, T, h)$ with corresponding red and blue paths is depicted in Figure 3(a). We are ready now to define the concept of a residual graph.

Definition 15 (Residual Graph G_1) *Let $\mathbb{N}(G, s, T, h)$ be a simple instance of the network coding problem with two destinations (i.e., $k = 2$). The residual graph G_1 is formed from G by reversing all links that belong to red paths (including links that belong both to red and blue paths).*

Definition 16 (Residual Graph G_2) *Let $\mathbb{N}(G, s, T, h)$ be a simple instance of the network coding problem with two destinations. The residual graph G_2 is formed from G by reversing all links that belong to red paths only (not including links that belong both to red and blue paths).*

Examples of the residual graphs G_1 and G_2 are depicted in Figure 3(b) and 3(c), respectively,

Lemma 17 *If $\mathbb{N}(G, s, T, h)$ is a simple acyclic instance, then the residual graphs G_1 and G_2 are acyclic.*

Proof: We denote the h red paths between s and t_1 by $P_1, \ldots, P_h$, and the h blue paths between s and t_2 by $\hat{P}_1, \ldots, \hat{P}_h$. Since $\mathbb{N}(G, s, T, h)$ is simple (and thus minimal with respect to link removal), every link in G is either red, blue or both red and blue. Suppose that the residual graph G_1 contains a cycle C. Each link $e \in C$ has a corresponding link e' in G which is either identical to e or is the reverse of e. In what follows, the color of a link e in C is defined to be the color of its corresponding link e' in G.

First, we note that C does not include terminals t_1 and t_2 (because terminal t_1 has in-degree 0 in G_1 and terminal t_2 has out-degree 0 in G_1). Second, we observe that C contains at least one link e_1 whose color is exclusively blue. Otherwise, the links that belong to C form a cycle in the original graph G consisting exclusively of red links (which contradicts our assumption that G is acyclic).

Third, we prove that C contains a link e_2 whose color is exclusively red. Otherwise, consider the case in which all links that belong to C are either exclusively blue or red and blue. At least one of them, e.g., e_1, is exclusively blue (which implies that the direction of e_1 in G is the same as in C). It can not be the case that all links of C are exclusively blue, otherwise there would be a cycle in G. Thus, there are links e in C which are both red and blue (which implies that they appear in opposite directions in G and C). Hence, there exists two links (u, v) and (v, w) in C, such that (u, v) is exclusively blue and (v, w) is both red and blue. This implies that in the original graph G node v has two input links, both of them belonging to blue paths. Since v is not a terminal node, it must have two output links, one for each blue path. Thus, the degree of this node in G is at least 4, in contradiction to our assumption that the degree of each internal node in G is at most 3.

We have shown that the cycle C of G_1 includes a link e_1 which is blue and not red, and a link e_2 which is exclusively red. Let $e_2' \in G$ be the reverse link of $e_2 \in C$. We now show that e_2' can be removed from the original graph G (which contradicts the minimality of the instance $\mathbb{N}(G, s, T, h)$). To that end, we show that there exists h link-disjoint paths $P_1', \ldots, P_h'$ from s to t_1 in G which do not include link e_2'.

Specifically, we use the techniques from the theory of network flows [11]. Let E' be a set of links that belong to red paths, i.e., $E' = \{e \mid e \in P_i, i = 1, \ldots, h\}$. The set E' represents a flow between s and t_1 of value h. We observe that G_1 is a residual graph with respect to this flow and C is an *augmenting cycle* in the residual graph G_1. We now augment E' along C and denote the resulting flow by E''. Flow E'' includes the following links: (a) each link $e = (v, u)$ in E' whose reverse link (u, v) does not belong to C, (b) each link $e = (v, u)$ that belongs to C and whose reverse link (u, v) does not belong to E', i.e.,

$$E'' = \{(v, u) \mid (v, u) \in E' \text{ and } (u, v) \notin C\} \cup \{(v, u) \mid (v, u) \in C \text{ and } (u, v) \notin E'\}$$

Note that E'' does not include link e_2'. Also, it is easy to verify that any cut that separates s and t_1 in G has at least h links that belong to E''. Indeed, for any cut (V_1, V_2), the number of links of C that cross (V_1, V_2) in the forward direction is identical to the number of links of C that cross (V_1, V_2) in the reverse direction. This implies [12] that E'' can be decomposed to h link-disjoint paths $P_1', \ldots, P_h'$ between s and t_1 in G. We note that these paths do not include the link e_2'. This implies that e_2' can be removed from the G, which contradicts the minimality of the instance $\mathbb{N}(G, s, T, h)$.

Our proof is illustrated in Figure 4. Figure 4(a) depicts a coding network $\mathbb{N}(G, s, T, h)$, where $h = 2$ and $T = \{t_1, t_2\}$ and the initial sets of red and blue paths. The corresponding residual graph G_1 is depicted in Figure 4(a).

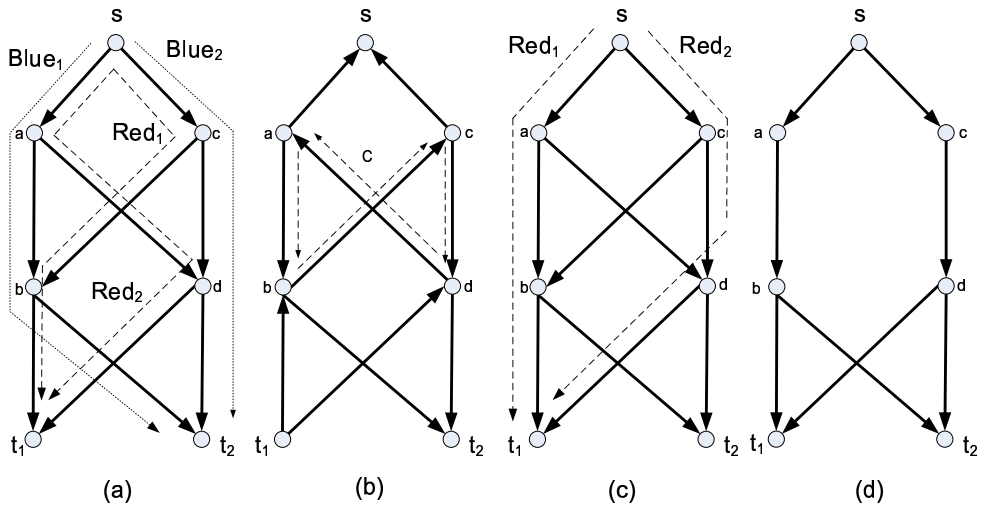

Figure 4: (a) An example of a non-minimal instance $\mathbb{N}(G, s, T, h)$ with red and blue paths. (b) The corresponding residual graph G_1. The graph has a cycle C. (c) The new set of red paths (after augmentation along cycle C). (d) A minimal instance formed from $\mathbb{N}(G, s, T, h)$ by removing links (a, d) and (c, b).

Note that G_1 has a cycle $\{a, b, c, d\}$ that includes two links (b, c) and (d, a) which are exclusively red. Figure 4(c) depicts a new set of red paths in G obtained by augmentation along cycle C. Since links (c, b) and (a, d) no longer belong to red paths, they can be removed from the graph. A minimal instance formed from $\mathbb{N}(G, s, T, h)$ by removing links (a, d) and (c, b) is depicted on Figure 4(d).

Using a similar argument, we can show that the same property holds for the residual graph G_2. Specifically, consider the graph $\hat{G}_2$ formed from G_2 by reversing all its links, and replace "red" with "blue" in the arguments above. ■

To this point, after fixing the red and blue paths of G, we have defined two residual graphs G_1 and G_2. We now define an additional (and final) graph $\hat{G}$, and a set of *green* paths. As in the case of G_1 and G_2, the definition of $\hat{G}$ depends on the set of red and blue path chosen above.

Definition 18 (Auxiliary Graph $\hat{G}$) *Let $\mathbb{N}(G, s, T, h)$ be a simple instance of the network coding problem with two destinations (i.e., $k = 2$). Let $P_1, \ldots, P_h$ be a set of red paths and let $P'_1, \ldots, P'_h$ be a set of blue paths. Let $\hat{G}$ be the graph formed from G by (a) Deleting links that belong to both a red path and a blue path in $\mathbb{N}$; and (b) Reversing links that belong to a red path and do not belong to a blue path.*

In the following lemma we prove that $\hat{G}$ consists of h link-disjoint paths between t_1 and t_2. We refer to these paths as *green* paths. An example of an auxiliary graph $\hat{G}$ is depicted in Figure 3(d).

Lemma 19 *Let $\mathbb{N}(G, s, T, h)$ be a simple instance of the network coding problem. Then, the corresponding graph $\hat{G}$ consists of h link-disjoint paths between t_1 and t_2 ($\hat{G}$ may also include isolated vertices of total degree 0).*

Proof: First we prove that each node $v \in \hat{G}$, excluding the terminals t_1, t_2 and the source node s, has exactly one incoming link and exactly one outgoing link. For each such node v in the original graph G, one of the following holds.

Case 1: Node v has one incoming link e and one outgoing link e'. In this case either a blue path or a red path or both red and blue paths (together) pass though v. In the first two cases, both links appear in $\hat{G}$, either in the original

direction or in the reverse direction, hence the node v has one incoming link and one outgoing link in the auxiliary graph $\hat{G}$. In the last case, both links e and e' do not appear in $\hat{G}$.

Case 2: Node v has two incoming links e_1, e_2 and one outgoing link e'. In this case, a red and a blue path enter node v through separate links and exit v (together) through e'. Thus, e' does not appear in $\hat{G}$ and one of the incoming links (e_1 or e_2) are reversed. Thus, the node v has one incoming link and one outgoing link in the auxiliary graph $\hat{G}$.

Case 3: Node v has one incoming link e and two outgoing links e'_1 and e'_2. In this case, a red and a blue path enter node v through link e and exit v through separate links e'_1 and e'_2. Thus, e does not appear in $\hat{G}$ and one of the outgoing links (e_1 or e_2) are reversed. Thus, the node v has one incoming link and one outgoing link in the auxiliary graph $\hat{G}$.

Next, we show that the in-degree and the out-degree of the source node s in the auxiliary graph $\hat{G}$ are equal. Since $\mathbb{N}(G, s, T, h)$ is minimal, s has no incoming links and each outgoing link of s belongs to a blue path or a red path. Let h' be the number of outgoing links of s in G that belong to red paths and do not belong to blue paths. Note that h' is the in-degree of s in $\hat{G}$. Since the number of blue paths is equal to the number of red paths, the number of outgoing links of s in G that belong to blue paths and do not belong to red paths is also h'. Thus, the out-degree of s in $\hat{G}$ is also h'.

Finally, we observe that in the original graph G, the in-degree of each terminal t_1, t_2 is exactly h. This follows from the fact that any link entering t_1 (t_2) is exclusively red (blue).

In summary, the auxiliary graph $\hat{G}$ has the following properties. First, the out-degree of t_1 and in-degree of t_2 are equal to h. Second, the in-degree of s is equal to its out-degree. Third, each other node $v \in \hat{G}$ has in-degree 1 and out-degree 1. Finally, $\hat{G}$ does not contain a cycle, otherwise there would be a cycle in G_1 (note that $\hat{G} \subseteq G_1$), which contradicts the minimality of $\mathbb{N}(G, s, T, h)$. This implies that there exist h link-disjoint paths in $\hat{G}$ that connect t_1 and t_2. These paths are referred to as *green* paths. It remains to show that each link in $\hat{G}$ belongs to a green path. Suppose, by way of contradiction, that there exists a link $e = (v, u) \in \hat{G}$ that does not belong to a green path. By the analysis above, one can extend this link into a path from v to t_2. This path will consist of links that do not belong to the h green paths described above, implying that t_2 has in-degree larger than h, a contradiction. ■

We denote the h link-disjoint paths between t_1 and t_2 in $\hat{G}$ as green paths. We observe that since $\hat{G}$ is a subgraph of the residual graphs G_1 and G_2, the green paths belong to G_1 and G_2 as well.

The proof of the following Lemma follows directly from Lemma 19 and the definitions of G_1 and G_2.

Lemma 20 *Let $\mathbb{N}(G, s, T, h)$ be a simple instance of the network coding problem. For each link $e = (v, u) \in G$ exactly one of the following conditions hold: (a) e belongs to a red and to a blue path in G, (b) e belongs to a red path in G and the reverse link (u, v) of e belongs to a green path in $\hat{G}$, (c) e belongs to a blue path in G and to a green path in $\hat{G}$.*

Proof: We consider three cases: e is a red link, a blue link, or a red and blue link in G. If e is both red and blue then e satisfies property (a) above. Note that e does not appear in $\hat{G}$, hence properties (b) and (c) stated in the lemma do not hold on e. Assume e is a red link (but not blue). Clearly, properties (a) and (c) stated in the lemma do not hold on e. Moreover, the reverse of e belongs to $\hat{G}$. Hence (by Lemma 19) e belongs to a green path and satisfies property (b). Finally, if $e = (v, u)$ is a blue link (but not red), properties (a) and (b) stated in the lemma do not hold on e. Again as e belongs to $\hat{G}$ it belongs to a green path and satisfies property (c). ■

4.1.2 The upper bound

We are ready to establish the upper bound on $\alpha(\mathbb{N})$ for simple instances $\mathbb{N}$ of the network coding problem with two terminals. We begin with the following lemma.

Lemma 21 *Let $\mathbb{N}(G, s, T, h)$ be a simple acyclic instance of the network coding problem with two terminals. Fix a set $\mathbb{S}_r$ of link-disjoint red paths in G between s and t_1 and a set $\mathbb{S}_b$ of link-disjoint blue paths in G between s and t_2. Let $\hat{G}$ be the auxiliary graph, defined above, and let $\mathbb{S}_g$ be a set of green paths between t_1 and t_2 in $\hat{G}$. Let $P_r \in \mathbb{S}_r$ be a red path, $P_b \in \mathbb{S}_b$ be a blue path, and $P_g \in \mathbb{S}_g$ be a green path. Then, there exists at most one node $v \in G$ that belongs to all three paths P_r, P_b and P_g.*

Proof: Let G_1, G_2 be residual graphs of G, as defined in Section 4.1.1. We observe that path P_g belongs to both G_1 and G_2. We also observe that for each red path $P_r \in G$ there exists a path $P'_r \in G_1$ formed by reversing the links of P_r. Suppose, by the way of contradiction, that there exist two nodes that belong to P_r, P_b and P_g. We denote these nodes by v_1 and v_2, such that v_1 is a predecessor of v_2 in P_g. First, we note that v_1 must be a predecessor of node v_2 in path P_b. Indeed, if this does not hold, then there exists a cycle in the residual graph G_2 formed by links that belong to paths P_b and P_g, which, by Lemma 17, contradicts the minimality of $\mathbb{N}(G, s, T, h)$. Second, the node v_2 must be a predecessor of node v_1 in P_r, otherwise there would be a cycle in the residual graph G_1 between the links that belong to paths P_g and P'_r, again contradicting the minimality of $\mathbb{N}(G, s, T, h)$. We conclude that node v_1 is a predecessor of node v_2 in P_b, while v_2 is a predecessor of node v_1 in P_r. This, however, implies that there exists a cycle in the original graph G, which contradicts the assumption that G is acyclic. ■

We summarize our results by the following theorem:

Theorem 22 *Let $\mathbb{N}(G, s, T, h)$ be a simple acyclic instance of the network coding problem with two terminals. Then, $\alpha(\mathbb{N}) \leq h^3$.*

Proof: Fix a set $\mathbb{S}_r$ of link-disjoint red paths in G between s and t_1 and a set $\mathbb{S}_b$ of link-disjoint blue paths in G between s and t_2. Let G_1, G_2, $\hat{G}$ be auxiliary graphs, as defined in Section 4.1.1 and let $\mathbb{S}_g$ be a set of green paths between t_1 and t_2 in $\hat{G}$. We denote by V' the set of nodes of G that are not terminals and whose in-degree is 2. Note that $|V'| = \alpha(\mathbb{N})$. Note also that each node $v \in V'$ has two incoming links: one belonging to a blue path (only) and one to a red path (only); and a single outgoing link that belongs to a red and to a blue path (the above follows directly from the simplicity of $\mathbb{N}$). Thus, by Lemma 20, each such node belongs to a red path $P_r \in \mathbb{S}_r$, a blue path $P_b \in \mathbb{S}_b$ and a green path $P_g \in \mathbb{S}_g$. Lemma 21 implies that at most one node belongs to the same three paths of different colors. Since there are exactly h red paths, h blue paths and h green paths, this implies that the number of nodes in V' is at most h^3. ■

4.2 Theorem 4: networks with $k > 2$

In this section we establish an upper bound on the size of $\alpha(\mathbb{N})$ for simple acyclic instances $\mathbb{N}(G, s, T, h)$ in which $k = |T|$ is arbitrary. This suffices to prove Theorem 4, stated in the Introduction.

Theorem 23 *Let $\mathbb{N}(G, s, T, h)$ be a simple acyclic instance of the network coding problem. Let $k = |T|$. Then, $\alpha(\mathbb{N}) \leq h^3 k^2$.*

Proof: We begin with the following observation. Let $t_i, t_j \in T$ be two terminals. By Theorem 14, there are h link-disjoint paths between s and t_i and h link-disjoint paths between s and t_j. Let $G'_{(i,j)}$ be a subgraph of G induced by links that belong to these disjoint paths. We note that the instance $\mathbb{N}(G'_{(i,j)}, s, \{t_i, t_j\}, h)$ is feasible, but not necessarily minimal. Let $\mathbb{N}_{(i,j)}(G_{(i,j)}, s, \{t_i, t_j\}, h)$ be a minimal instance obtained by repeatedly removing redundant links from $G'_{(i,j)}$. Note that $\mathbb{N}_{(i,j)}$ is simple. We denote by $\mathbb{P}^i_{(i,j)}$ and $\mathbb{P}^j_{(i,j)}$ a set of h link-disjoint paths in $G_{(i,j)}$, between s and t_i and between s and t_j, respectively. Consider the set of nodes $V_{(i,j)}$ in $G_{(i,j)}$ that have in-degree 2, not including the source and terminals. Note that by definition of $\alpha(\mathbb{N})$, $\alpha(\mathbb{N}_{(i,j)}) = |V_{(i,j)}|$. By Theorem 22, $\alpha(\mathbb{N}_{(i,j)}) \leq h^3$.

Assume, by way of contradiction, that $\alpha(\mathbb{N})$ exceeds h^3k^2. Then, there exists at least one node v with in-degree 2, which is not the source or any of the terminals; and does not belong to $V_{(i,j)}$ for all pairs $t_i, t_j \in T$. Let e' and e'' be the incoming links of v and let e be the outgoing link of v.

We show that the instance $\mathbb{N}(G, s, T, h)$ remains feasible after the deletion of either e' or e'', which contradicts the minimality of $\mathbb{N}(G, s, T, h)$. Note that for each terminal t_i, there are several possible sets $\{\mathbb{P}^i_{(i,j)} \mid t_j \in T\}$ of h link-disjoint paths between s and t_i. If for each terminal $t_i \in T$ it holds that there exists $\mathbb{P}^i_{(i,j)}$ that does not include e', then link e' can be omitted from $\mathbb{N}(G, s, T, h)$ without violating its feasibility. Otherwise there exists a terminal t_i such that link e' is included by all sets $\{\mathbb{P}^i_{(i,j)} \mid t_j \in T\}$. However, in this case it holds that all sets $\{\mathbb{P}^j_{(i,j)} \mid t_j \in T\}$ do not include e'', which contradicts the minimality of $\mathbb{N}$. Indeed, if e'' belongs to some set $\mathbb{P}^j_{(i,j)}$, then v necessarily has degree 2 in $G_{(i,j)}$ and thus belongs to $V_{(i,j)}$. ■

5 Upper bound for general (cyclic) networks

In this section we establish an upper bound of $(2B+1)h^3k^2$ on the number of encoding nodes for coding networks $\mathbb{N}(G, s, T, h)$ with cycles (Theorem 7). Here B is the size of the feedback link set of G. The proof of the upper bound for cyclic networks is very similar to the proof presented for Theorem 4. Specifically, we show that in a cyclic network, the value of $\alpha(\mathbb{N})$ is bounded by $(2B+1)h^3k^2$. This fact, coupled with Lemmas 11 and 13, is sufficient to prove the correctness of Theorem 7.

In what follows we roughly outline the proof of Theorem 7 emphasizing only on the changes required to extend the proof of Theorem 4 to the case in which the given network $\mathbb{N}(G, s, T, h)$ includes cycles.

5.1 Residual graphs and link-disjoint paths

Let $\mathbb{N}(G, s, T, h)$ be a simple network in which $T = \{t_1, t_2\}$ (i.e., $k = 2$). As in Section 4, $\mathbb{N}(G, s, T, h)$ is feasible, thus by Theorem 14 there exist h link-disjoint paths $P_1, \ldots, P_h$ that connect s and terminal t_1 (referred to as red paths) and h link-disjoint paths $\hat{P}_1, \ldots, \hat{P}_h$ between s and t_2 (referred to as blue paths). We assume, without loss of generality, that paths $P_1, \ldots P_h$ ($\hat{P}_1, \ldots \hat{P}_h$) do not include cycles. Indeed, if the size of a minimum cut between s and a terminal t is h, then there exist h link-disjoint paths between s and t that do not include cycles [11]. Note that we did not need to make this assumption in Section 4 as the given network was acyclic.

With this additional property of our red and blue paths, Lemmma 17, 19, and 20 hold in the cyclic case as well. The proof of Lemma 17 needs minor modifications, and the proofs of the latter lemmas hold without any modifications at all. We now prove an analog to Lemma 21, and state the resulting analogs to Theorem 22 and 23.

Lemma 24 *Let $\mathbb{N}(G, s, T, h)$ be a simple instance of the network coding problem with two terminals and let B be the size of the minimum feedback link set in G. Fix a set $\mathbb{S}_r$ of link-disjoint red paths in G between s and t_1 and a set $\mathbb{S}_b$ of link-disjoint blue paths in G between s and t_2. Let $\hat{G}$ be the auxiliary graph, as defined in Section 4.1, and let $\mathbb{S}_g$ be a set of green paths between t_1 and t_2 in $\hat{G}$. Let $P_r \in \mathbb{S}_r$ be a red path, $P_b \in \mathbb{S}_b$ be a blue path, and $P_g \in \mathbb{S}_g$ be a green path. Then, there exists at most $(2B+1)$ nodes in G that belongs to all three paths P_r, P_b and P_g.*

Proof: Let G_1, G_2 be residual graphs of G, as defined in Section 4.1.1. We observe that path P_g belongs to G_1 and G_2. We also observe that for each red path $P_r \in G$ there exists a path $P'_r \in G_1$ formed by reversing the links of P_r. Suppose, by the way of contradiction, that there exist $2B+2$ nodes that belong to P_r, P_b and P_g. We denote these nodes by $v_1, \ldots, v_{2B+2}$ such that for $i = 1$ to $2B+1$, v_i is a predecessor of v_{i+1} in P_g. First, we note that v_i must be a predecessor of node v_{i+1} in path P_b. Indeed, if this does not hold, then there exists a cycle in the residual graph G_2 formed by links that belong to paths P_b and P_g, which, by Lemma 17, contradicts the minimality

of $\mathbb{N}(G, s, T, h)$. Second, v_{i+1} must be a predecessor of v_i in P_r, otherwise there would be a cycle in the residual graph G_1 between the links that belong to paths P_g and P'_r, again contradicting the minimality of $\mathbb{N}(G, s, T, h)$.

We conclude that for each i, v_i is a predecessor of v_{i+1} in P_b, while v_{i+1} is a predecessor of v_i in P_r. This implies that there exist at least $2B + 1$ cycles in the subgraph G' of G induced by links that belong to P_r and P_b. We now show that the minimum feedback link set of G' is at least $B + 1$, which contradicts our assumption that the minimum feedback link set of G is B. Consider a link e in G'. If e is exclusively red or blue, then e appears in only one of the $2B + 1$ cycles in G'. Otherwise if e is both red and blue, it can appear in at most 2 cycles in G'. Thus, the minimum feedback link set of G' is of size at least $B + 1$. ■

5.2 Upper bound

The following theorems establish the upper bound on the number of encoding nodes in networks with cycles.

Theorem 25 *Let $\mathbb{N}(G, s, T, h)$ be a simple instance of the network coding problem with two terminals. Then, $\alpha(\mathbb{N}) \leq (2B + 1)h^3$.*

Proof: The proof follows the same lines as that of Theorem 22, using Lemma 24 instead of Lemma 21. ■

Theorem 26 *Let $\mathbb{N}(G, s, T, h)$ be a simple instance of the network coding problem. Let $k = |T|$. Then, $\alpha(\mathbb{N}) \leq (2B + 1)h^3k^2$.*

Proof: The proof follows the same lines as that of Theorem 23, using Lemma 24 instead of Lemma 21. ■

Let $\mathbb{N}(G, s, T, h)$ be a general (not necessarily simple) instance for the network coding problem. As mentioned above, Theorem 26 implies Theorem 7 by Lemmas 11 and 13. However, the careful reader may have noted that the bound on $Opt(\mathbb{N})$ stated in Theorem 7 may be strengthened by replacing the parameter B by the size of the minimum feedback link set in any simple instance $\hat{\mathbb{N}}$ obtained form $\mathbb{N}$ by link removal.

In the following definition $B(\mathbb{N})$ denotes the minimum size of a feedback link set of the underlying graph G of $\mathbb{N}(G, s, T, h)$.

Definition 27 *Let $\mathbb{N}(G, s, T, h)$ be a feasible instance of the network coding problem. Define $Opt_{FES}(\mathbb{N})$ to be $\min_{\hat{\mathbb{N}}} B(\hat{\mathbb{N}})$ taken over all feasible subinstances $\hat{\mathbb{N}}(\hat{G}, s, T, h)$ obtained from $\mathbb{N}(G, s, T, h)$ by removing links from the underlying graph G (i.e., $\hat{G} \subseteq G$).*

Theorem 28 (Strong upper bound for cyclic networks) *Let $\mathbb{N}(G, s, T, h)$ be a feasible instance of the multicast network coding problem. Let $k = |T|$. Then there exists a feasible network code for $\mathbb{N}$ in which the number of encoding nodes is at most $(2Opt_{FES}(\mathbb{N}) + 1)h^3k^2$.*

Theorem 28 implies that given a coding network $\mathbb{N}(G, s, T, h)$ it is desirable to find a simple network $\hat{\mathbb{N}}(\hat{G}, s, T, h)$, such that $\hat{G} \subseteq G$ and the size of the minimum feedback link set of $\hat{G}$ is $Opt_{FES}(\mathbb{N})$. However, in Section 6.2 we show that this task is $\mathcal{NP}$-hard. Moreover, we show that even finding a network $\hat{\mathbb{N}}$ with a feedback link set that is *somewhat close* to being of size $Opt_{FES}(\mathbb{N})$ is $\mathcal{NP}$-hard. Hence, Theorem 28 should be viewed as an existential result only rather than one which can be efficiently constructed.

6 Negative results

In this section we prove Theorems 3, 5, 8, and the $\mathcal{NP}$-hardness of computing or approximating $Opt_{FES}(\mathbb{N})$ (as defined in Section 5).

6.1 Theorem 3: Exact and approximate computation of $Opt(\mathbb{N})$

Proof: (of Theorem 3) We present a reduction between the Link-Disjoint Path (LDP) problem in directed graphs and the problem of computing the minimum number of encoding nodes required in a multicast network coding instance $\mathbb{N}$. Given a directed graph $G = (V, E)$ and two pairs of nodes in V, (s_1, t_1) and (s_2, t_2) the LDP problem is the problem of finding two link-disjoint paths in G connecting s_1 to t_1 and s_2 to t_2. The decision version of this problem is known to be $\mathcal{NP}$-complete [13]. This reduction will imply the hardness results stated in Theorem 3.

Consider an instance $\{G', s_1, s_2, t_1, t_2\}$ to Problem LDP. We construct an instance $\mathbb{N}$ to the multicast network coding problem. The underlying graph G of $\mathbb{N}$ is the graph G enhanced with one additional node s, and four additional links: (s, s_1), (s, s_2), (s_1, t_2) and (s_2, t_1). The set of terminals in $\mathbb{N}$ is $T = \{t_1, t_2\}$, and $h = 2$. We show that $Opt(\mathbb{N}) = 0$ iff there exist two link-disjoint paths (one between s_1 and t_1 and the second between s_2 and t_2) in G'.

Suppose that there exist two link-disjoint paths in G' connecting s_1 to t_1 and s_2 to t_2. This implies that there exist two link-disjoint (Steiner) trees in G with root s and terminals t_1 and t_2, such that the first tree includes links (s, s_1), (s_1, t_2) and a path between s_1 and t_1, while the second tree includes links (s, s_2), (s_2, t_1) and a path between s_2 and t_2. We conclude that $Opt(\mathbb{N}) = 0$.

Assume that $\mathbb{N}$ has a network code with no encoding nodes. This implies the existence of two link-disjoint (Steiner) trees with s as their root and t_1 and t_2 as their terminals. Our construction implies that one of the trees include a path P_1 between s_1 and t_1, while the other must include a path P_2 between s_2 and t_2. Note that paths P_1 and P_2 must be link-disjoint, which completes the proof of our assertion.

To complete the proof of Theorem 3, let n be the number of nodes in the underlying graph of $\mathbb{N}$ and let $\varepsilon > 0$ be any constant. Due to the gap location of our reduction, it is clear that $Opt(\mathbb{N})$ cannot be efficiently approximated by any multiplicative factor unless $\mathcal{P} = \mathcal{NP}$.

We now sketch the proof for the additive approximation gap. We use a slightly different reduction which guarantees an additive gap of $n^{1-\varepsilon}$. Namely, instead of defining $\mathbb{N}$ as above, we define an new instance $\mathbb{N}_{new} = (G_{new}, s_{new}, T_{new}, h)$ of the network coding problem. Roughly speaking, the new underlying graph G_{new} is set to be $n^{1/\varepsilon}$ copies of the previous graph G used above. A single source node s_{new} is connected to s_1 and s_2 in each copy of G, and two new nodes t_1^{new} and t_2^{new} are added with two multiple links connecting each terminal t_i in each copy of G to t_i^{new}. $T_{new} = \{t_1^{new}, t_2^{new}\}$, and $h_{new} = 2n^{1/\varepsilon}$. It is now not hard to verify that $Opt(\mathbb{N}_{new}) = 0$ iff there exist two link-disjoint paths, one between s_1 and t_1 and the second between s_2 and t_2 in G'. Moreover if $Opt(\mathbb{N}_{new}) \neq 0$ then it must be the case that $Opt(\mathbb{N}_{new}) > n^{1/\varepsilon}$ (one encoding node per each copy of G), which suffices to prove our assertion. The detailed proof is omitted. ■

6.2 Finding feasible subgraphs with small feedback link set

Theorem 29 *Let $\varepsilon > 0$ be any constant. Let $\mathbb{N}(G, s, T, h)$ be an instance of the multicast network coding problem in which the underlying graph has m links. Approximating the value of $Opt_{FES}(\mathbb{N})$ within any multiplicative factor or within an additive factor of $m^{1-\varepsilon}$ is $\mathcal{NP}$-hard.*

Proof: Our proof is very similar in nature to that presented in Section 6.1. Namely, we start by presenting a reduction between the Node-Disjoint Path (NDP) problem in directed graphs and the problem of computing $Opt_{FES}(\mathbb{N})$. Given a directed graph $G = (V, E)$ and two pairs of nodes in V, (s_1, t_1) and (s_2, t_2) the NDP problem is the problem of finding two node-disjoint paths in G connecting s_1 to t_1 and s_2 to t_2. This problem is known to be $\mathcal{NP}$-hard (this can be seen by a simple reduction from Problem LDP).

In Figure 5, we show how an instance $\{G', s_1, t_1, s_2, t_2\}$ of Problem NDP is reduced to an instance $\mathbb{N}(G, s, T, h)$ of the network coding problem (where $T = \{\hat{t}_1, \hat{t}_2\}$). In this reduction, it is not hard to verify that there exists node-

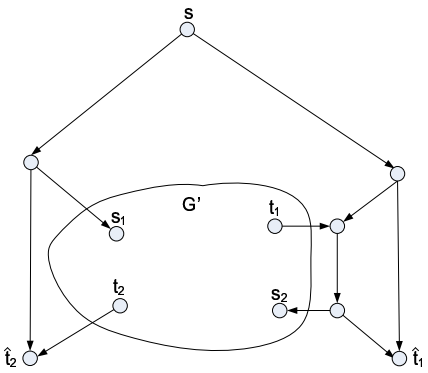

Figure 5: Reduction from NDP to computing $Opt_{FES}(\mathbb{N})$. The graph G' is depicted by an oval object including nodes s_1, s_2, t_1 and t_2. The network $\mathbb{N}(G, s, T, h)$ includes two terminals $\hat{t}_1$ and $\hat{t}_2$ and has $h = 2$.

disjoint paths between s_1 and t_1 and between s_2 and t_2 iff $Opt_{FES}(\mathbb{N}) = 0$ (i.e., there exists a feasible acyclic network $\hat{\mathbb{N}}(\hat{G}, s, T, h)$ in which $\hat{G} \subseteq G$). This shows that it is $\mathcal{NP}$-hard to distinguish between instances $\mathbb{N}$ in which $Opt_{FES}(\mathbb{N}) = 0$ and instances in which $Opt_{FES}(\mathbb{N}) \geq 1$.

To obtain our inapproximability results, we follow the line of proof given in Section 6.1, and use multiple copies of the reduction above. The detailed proof is omitted. ■

6.3 Theorem 5: Global lower bound on $Opt(\mathbb{N})$

In this section we establish a lower bound on the number of encoding nodes that may be required to obtain a multicast connection (Theorem 5) in acyclic networks. We start by presenting a lower bound for instances to the network coding problem with only two terminals ($k = 2$). We then generalize our instance to arbitrary values of k. We start by proving the following lemma.

Lemma 30 *Let h be any integer larger than 2. There exist instances $\mathbb{N}(G, s, T, h)$ to the multicast network coding problem with $k = |T| = 2$ that require at least $\frac{(h-1)h}{2} = \Omega(h^2)$ encoding nodes.*

Proof: For any value of $h \geq 2$ we present a coding network $\mathbb{N}$ that requires $\frac{(h-1)h}{2}$ encoding nodes. An example of the network $\mathbb{N}$ for $h = 5$ is depicted in Fig. 6. The network includes the following nodes: The source node s and two terminals t_1 and t_2; intermediate v-nodes $v_1^1, \ldots, v_h^1$, $v_2^2, \ldots, v_h^2$, $\ldots$, v_{h-1}^{h-1}, v_h^{h-1} and v_h^h; and intermediate u-nodes $u_2^1, \ldots, u_h^1$, $u_3^2, \ldots, u_h^2$, $\ldots$, u_{h-1}^{h-2}, u_h^{h-2} and u_h^{h-1}. The links of the network are defined as follows: s is connected by a link to each of the nodes $v_1^1, \ldots, v_h^1$; each node v_i^i, $i = 1, \ldots, h$ is connected by a link to t_1; each node v_h^i, $i = 1, \ldots, h$ is connected by a link to t_2; each node v_i^j, $i = j, \ldots, h-1$, $j = 1, \ldots, h-1$ is connected by a link to u_{i+1}^j; each node v_i^j, $i = j+1, \ldots, h$, $j = 1, \ldots, h-1$ is connected by a link to u_i^j; finally, each node u_i^j is connected to v_i^{j+1}. In $\mathbb{N}$, the source node is to transmit h packets to the terminals.

We now analyze the properties of $\mathbb{N}$. It is not hard to verify that $\mathbb{N}$ is acyclic, and satisfies the degree constraints specified in Definition 10. We now show that $\mathbb{N}$ is feasible and minimal with respect to link removal. Note that the underlying graph G of $\mathbb{N}$ contains h link-disjoint paths from s to each terminal; and removal of any link in G will result in a min-cut between s and some terminal of value $h-1$. This implies that $\mathbb{N}$ is simple and that $Opt(\mathbb{N}) = \alpha(\mathbb{N})$ (recall that $\alpha(\mathbb{N})$ is the number of nodes in G of in-degree 2, excluding the terminals). It is not hard to verify that there are $\frac{(h-1)h}{2} = \Omega(h^2)$ such nodes (marked as u_i^j). ■

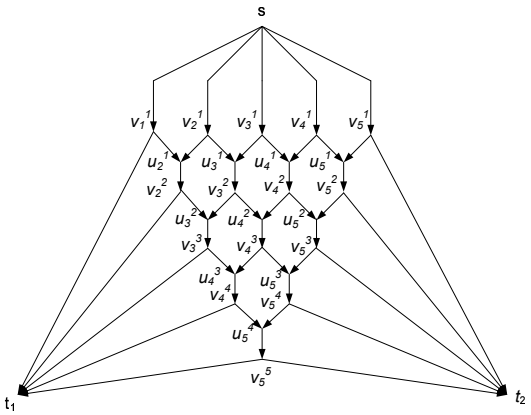

Figure 6: Lower bound on the number of encoding nodes for $h = 5$.

To prove Theorem 5, we extend Lemma 30 to capture the general case in which $k \geq 2$. This is done by concatenating many instances of $\mathbb{N}$ presented above.

Proof: (of Theorem 5) We build a network $\mathbb{N}_{new}(G_{new}, s_{new}, T_{new}, h)$ by using the network $\mathbb{N}$ defined in the proof of Lemma 30 as a basic building block. Let G be the underlying network of $\mathbb{N}$, we define G_{new}. G_{new} will have one source node s_{new}, and k terminals t_1^{new} to t_k^{new}, and is defined by $k-1$ copies of the graph G (which we denote $G_1, \ldots, G_{k-1}$) as follows. The source s_{new} is connected by h multiple links to the source s of G_1 (to avoid multiple links, we could have defined s_{new} to be the source node s of G_1 also). For $i = 1, \ldots, k-2$, the node t_2 of copy i is connected to the source node s of copy $i+1$ (again achieved by h multiple links or by unifying the node t_2 of copy i with s of copy $i+1$). For $i = 1, \ldots, k-2$, node t_1 of G_i is connected (see remark above) to the terminal t_i^{new}. Finally, node t_2 of G_{k-1} is connected (see remark above) to the terminal t_k^{new}.

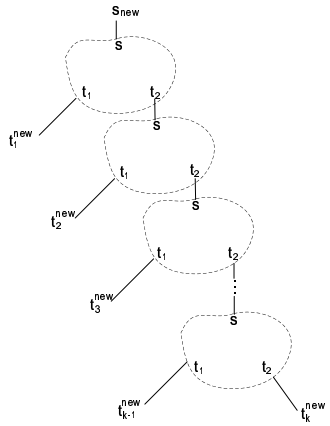

Figure 7: Lower bound on the number of encoding nodes for $k > 2$.

The construction of $\mathbb{N}_{new}$ is demonstrated in Fig. 7. As in Lemma 30, it can be shown that the removal of any link from G_{new} will result in a min-cut between s_{new} and one of the terminals of value $h-1$, implying that each and every link of G_{new} must be used in a network code that transfers all h packets of s_{new} to the terminals of G_{new}. This implies that the links of the form (u_i^j, v_i^{j+1}) in each copy of G must be encoding links, which, in turn, completes our proof. The detailed proof is omitted. ■

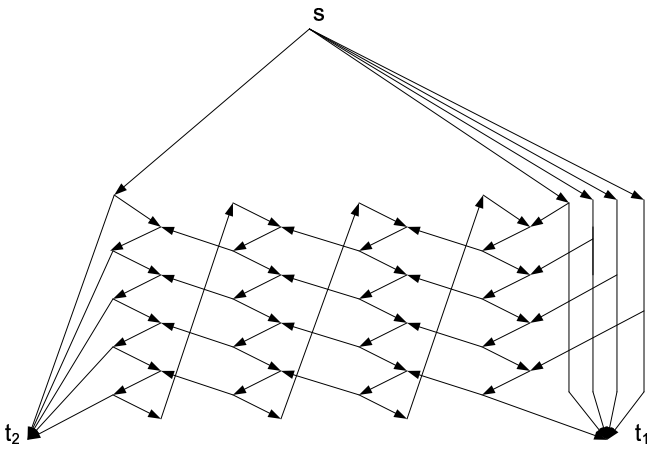

Figure 8: Lower bound for $Opt(\mathbb{N})$ in cyclic networks.

6.4 Theorem 8: Lower bound for cyclic networks

In this section, we study instances $\mathbb{N}$ in which the underlying graph is cyclic, and sketch the proof Theorem 8.

Proof: (of Theorem 8) The multicast instance $\mathbb{N}(G, s, T, h)$ we suggest is depicted in Fig. 8 (for the case $h = 5$) and is defined as follows. The underlying graph G has a source s, and two terminals t_1 and t_2. There are $h-1$ links leaving s towards the right, and a single link leaving s towards the left. Call the links going right, *regular* links and the remaining link leaving s *special*. The terminal t_1 is connected to s via $h-1$ paths that start at regular links, and a single path that starts in the special link. The same holds for t_2. The graph is constructed such that the single path from s to t_1 starting at the special link, intersects the paths from s to t_2 starting at regular links (in a systematic manner, as depicted in Fig. 8). Let B be the size of the minimum feedback link set in G. Now, it is not hard to verify that (a) the graph G is minimal with respect to link removal (every link appears in a minimum cut of size h), and (b) the number of encoding nodes in $\mathbb{N}(G, s, T, h)$ is $(B+1)(h-1) = \Omega(Bh)$. This proves the first assertion of the theorem. For the second assertion, set $h = 2$ and notice that the number of nodes in G is $2(B+1)(h-1) + h + 3 = 2(B+1) + 5$, while the number of encoding nodes is $(B+1)(h-1) = B+1$.

■

7 Conclusion

We consider the design of network codes which enable the source to transmit at rate h to k terminals and include a bounded number of encoding nodes. For acyclic networks, we present an efficient and simple procedure which finds a network code that enables the source to transmit at capacity, in which the number of encoding nodes is independent of the size of the network and is bounded by h^3k^2. We show that our bound on the number of encoding nodes may depend both on h and k as we present networks in which any feasible network code has at least $\Omega(h^2k)$ encoding nodes. It would be interesting if the hk gap between are upper and lower bound could be settled.

For general (cyclic) networks we present results of similar nature. Namely, we present an upper bound which depends on the size of the minimum feedback link set B of the network of size $(2B+1)h^3k^2$. Our lower bound in this case is of order $\max(|V|/2, Bh, h^2k)$ where $|V|$ is the total number of nodes in the network.

Acknowledgments

We would like to thank Matthew Cook for useful discussions.

References

[1] 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.

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

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

[4] 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.

[5] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen. Polynomial Time Algorithms for Multicast Network Code Construction. *Submitted to IEEE Transactions on Information Theory*, 2003.

[6] C. Fragouli, E. Soljanin, and A. Shokrollahi. Network Coding as a Coloring Problem. In *Proceedings of CISS*, 2004.

[7] C. Fragouli and E. Soljanin. Information Flow Decomposition for Network Coding. *Submitted to IEEE Transactions on Information Theory*, 2004.

[8] A. Tavory, M. Feder, and D. Ron. Bounds on Linear Codes for Network Multicast. *Electronic Colloquium on Computational Complexity (ECCC)*, 10(033), 2003.

[9] Y. Wu, K. Jain, and S.Y. Kung. A Unification of Menger's and Edmonds' Graph Theorems and Aslswede et al's Network Coding Theorem. *Allerton Conference on Communications, Control, and Computing*, 2004.

[10] M.R. Garey and D.S. Johnson. *Computers and Intractability*. Freeman, San Francisco, CA, USA, 1979.

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

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

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