Network Coding as a Coloring Problem

Christina Fragouli, Emina Soljanin, Amin Shokrollahi
christina.fragouli@epfl.ch, emina@lucent.com, amin.shokrollahi@epfl.ch

***Abstract*— We consider a multicast configuration with two sources, and translate the network code design problem to vertex coloring of an appropriately defined graph. This observation enables to derive code design algorithms and alphabet size bounds, as well as establish a connection with a number of well-known results from discrete mathematics that increase our insight in the different trade-offs possible for network coding.**

I. Introduction

Network coding is an emerging area in coding theory which attempts to make connections between algebraic tools used in coding and information transmission on communication graphs. The min-cut, max-flow theorem states that a source node can send a commodity through a network to a sink node at the rate determined by the flow of the min-cut separating the source and the sink. By min-cut we refer to the minimum number of edges we need to remove to disconnect the source and the sink. Recently, Ahleswede et al. [1] have shown that if the nodes in the network can decode and re-encode incoming bits, the min-cut rate can be also achieved in multicasting to several sinks. Shortly afterwards Li et al. [2] showed that linear coding suffices to achieve the optimal rate.

This area is expected to attract research interest and have a significant impact on network management and design. Indeed, preliminary studies show that network coding may increase the achievable multicast throughput by significant amounts. Thus deployment of network coding could help better exploit shared resources such as Internet connections or wireless bandwidth.

Moreover, from a theoretical point of view, this is a very attractive interdisciplinary study area that poses interesting questions across diverse areas such as information theory [1], [3], algorithms [4], [5], algebra and coding theory [6], and graph theory [2]. In this paper we continue this trend by establishing connections with coloring problems for graphs.

We restrict our attention to a multicast configuration with $h = 2$ sources. Some of the results extend in higher dimension ($h > 2$ sources) as is discussed in [7]. Moreover, algorithms for $h = 2$ sources can be used as a basis to develop suboptimal algorithms for the case $h > 2$.

We start by relating the network code design problem to the problem of coloring an appropriately defined graph. A crucial step in facilitating the connection is the subtree decomposition method. Since this method is interesting in its own, it is described first. With this starting point, we propose code design algorithms, derive alphabet size bounds, and apply a number of well-known results from discrete mathematics to increase our insight into network coding.

The paper is organized as follows. Section II describes our notation and reviews the subtree decomposition. Section III establishes the connection with coloring. Sections IV, V, and VI discuss combinatorial results.

II. Subtree Decomposition

The subtree decomposition can be thought of as keeping only the "sufficient information" of the underlying graph structure that is necessary for the network code design. The main idea is that we can "group together" the parts of the network through which the same information flows. Thus, starting from an arbitrary graph, we map it to a graph with a much smaller number of edges and vertices, and still retain all the necessary information for the code design.

A. Subtree Graph

Consider an acyclic directed graph $G = (V, E)$ with unit capacity edges that models a communication network. Let h unit rate information sources $\{S_1, \ldots, S_h\}$ located on the same vertex simultaneously multicast information to N receivers $\{R_1, \ldots, R_N\}$. Assume that the min-cut between the

source and each receiver is greater or equal to h (*min-cut condition*).

In linear network coding through each edge e of G flows a linear combination of the sources. We refer to the vector of linear coefficients $c(e)$ as the coding vector associated with edge e. The coding vector of an output edge of a node lies in the linear span of the coding vectors of the node's input edges. The network code design problem is to select a coding vector for each edge of the network so that each receiver has a full-rank system of linear equations to solve. Throughout this discussion we use the example in Fig. 1, which is an example of a network topology with two sources multicasting to the same set of three receivers.

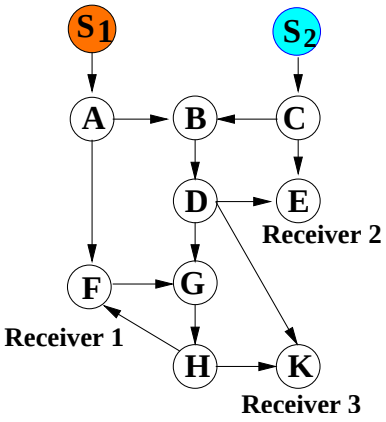

Fig. 1. Topology with two sources $\{S_1, S_2\}$ and three receivers $\{F, E, K\}$.

For a given graph G, the associated line graph $L(G)$ is the graph with vertex set $E(G)$ in which two vertices are joined if and only if they are adjacent as edges in G. The line graph for the example in Fig. 1 is depicted in Fig. 2.

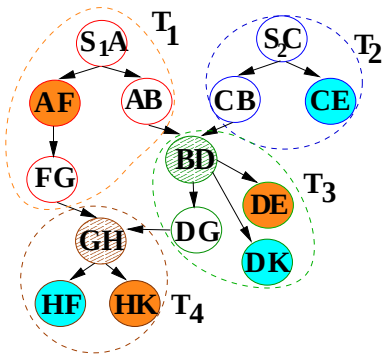

Fig. 2. Line graph that illustrates the coding points BD and GH and the subtree decomposition.

Without loss of generality we may assume that the line graph contains a node corresponding to each of the h sources. We refer to these nodes as *source nodes*. Each node with a single input edge merely forwards its input symbol to its output edges. Each node with two or more input edges performs a coding operation (linear combination) on its input symbols, and forwards the result to all of its output edges. We refer to these last nodes as *coding points*. We also refer to the node corresponding the last edge of the path (S_i, R_j), as the *receiver node* for receiver R_j and source S_i. For a configuration with h sources and N receivers there exist hN receiver nodes. For example, in Fig. 2, S_1A and S_2C are source nodes, BD and GH are coding points, and AF, HF, HK, DK, DE and CE are receiver nodes.

We partition the line graph into a disjoint union of subsets T_i so that the following properties hold:

1) each T_i contains exactly one source node or a coding point, and
2) every other node belongs to the T_i containing its first ancestral coding or source node.

It is easy to see that the above conditions imply the following:

- each T_i is a tree because the only nodes with two or more input edges in the line graph are the coding points,
- the same linear combination of source symbols flows through all the nodes that belong to the same T_i.

We shall call the subset T_i a *source subtree* if it starts with a source node or a *coding subtree* if it starts with a coding point. Fig. 2 shows the four subtrees $\{T_1, T_2, T_3, T_4\}$ of the network in Fig. 1; T_1 and T_2 are source subtrees, T_3 and T_4 are coding subtrees.

For the network code design problem, we only need to know how the subtrees are connected and which receiver nodes are in each T_i, whereas the structure of the network inside a subtree does not play any role. Thus we can contract each subtree to a node and retain only the edges that connect the subtrees, to get the *subtree graph* Γ.

Indeed, all nodes inside each T_i share the same coding vector, which we denote by $c(T_i)$. Thus, the network multicast problem is reduced to assigning an h-dimensional coding vector $c(T_i)$ to each subtree T_i, which will be observed by all receivers that have receiver nodes contained in T_i, so that each

receiver has a full-rank system of linear equations to solve. We refer to an assignment of coding vectors that achieves this goal as a *valid* network code.

The subtree graph for our example network of Fig. 1 is shown in Fig. 3. The receiver nodes

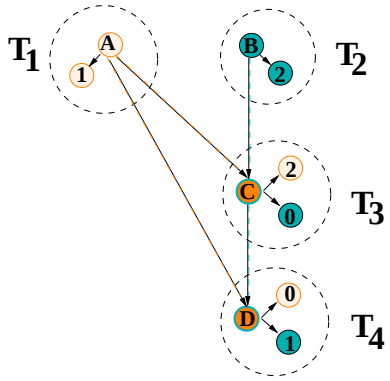

Fig. 3. Subtree Graph.

corresponding to source-receiver pairs inside each subtree are represented pictorially in Fig. 3. An example of a valid network code is

$$c(T_1) = [10] \;\; c(T_2) = [01] \;\; c(T_3) = [11] \;\; c(T_4) = [01].$$

The fact that the min-cut condition is satisfied for every user imposes structural properties on the subtree graph. More specifically,

- for each receiver R_j, the h receiver nodes, corresponding to the last edges on the paths (S_i, R_j), $1 \leq i \leq h$, belong to distinct subtrees. Thus,
- each subtree contains at most N receiver nodes.

In particular, we see that the number of subtrees is at least h.

B. Minimal Subtree Graphs and Their Properties

We define minimal subtree graphs as:

Definition 1: A subtree graph is called *minimal* with the min-cut property if removing any edge would violate the min-cut condition for at least one receiver.

We can think of minimal subtree graphs as graphs where no subtree can be assigned the same coding vector as one of its parents.

Minimal subtree graphs have structural properties, that follow directly from the Definition 1. Here we present the properties that we are going to use in our discussion. Proofs can be found in [7].

Lemma 1: Consider a minimal subtree graph.

- For all valid assignments of coding vectors the vectors assigned to the parents of any given subtree are linearly independent.
- Each coding subtree has at least 2 and at most h parent subtrees.
- Assume a coding subtree has p parents, c children and contains r receiver nodes. Then $p \leq c + r$. For example, for $c = 0$, $r \geq p$, that is, a coding subtree with no children contains at least as many receiver nodes as parents.
- In a minimal configuration with $h = 2$ sources each coding subtree contains at least two receiver nodes.

III. Connection with coloring

Coding vectors for networks with h sources live in the h dimensional space $\mathbb{F}_q^h$. Since in network coding, we only need to ensure that the coding vectors assigned to the subtrees having receivers in common be linearly independent, it is enough to consider only the vectors in the projective space $\mathbb{PG}(h-1, q)$ defined as follows:

Definition 2: The projective $(h-1)$-space over $\mathbb{F}_q$ is the set of h-tuples of elements of $\mathbb{F}_q$, not all zero, under the equivalence relation given by

$$[a_1 \dots a_h] \sim [\lambda a_1 \dots \lambda a_h], \;\; \lambda \neq 0, \;\; \lambda \in \mathbb{F}_q.$$

For networks with two sources, it is sufficient to consider the points on the projective space of dimension 1, *i.e.,* the projective line $\mathbb{PG}(1, q)$

$$[0\,1], \;\; [1\,0], \;\; \text{and } [1\,\alpha^i] \text{ for } 0 \leq i \leq q-2, \qquad (1)$$

where α is a primitive element of $\mathbb{F}_q$. Any two different points on the projective line $\mathbb{PG}(1, q)$ form a basis for $\mathbb{F}_q^2$. Geometric objects that have that property are known as *arcs*. In combinatorics, arcs correspond to vectors *in general position*:

Definition 3: Set $\mathcal{A}$ of vectors in $\mathbb{F}_q^h$ are said to be general position if any h vectors in $\mathcal{A}$ are linearly independent.

Lemma 2: ([8, Chapter 11]) Let $g(h, q)$ denote the maximum number of points in general position in an h-dimensional space over a finite field F_q where q is a prime power and $h \geq 2$. Then

$$g(h = 2, q) = q + 1,$$
$$g(h, q) \geq q + 1, \quad n \geq 2.$$

To conclude, to design a network code for $h = 2$ sources, we need to assign a 2-dimensional coding vector over F_q to each subtree. Without loss of

generality, we can restrict the coding vectors we employ to belong to the set of vectors in general position described by Eq. (1).

We can equivalently think of the $q+1$ vectors in Eq. (1) as colors, and require that every receiver observes two different colors. Thus we can relate the problem of designing a network code to the problem of vertex coloring of a suitably defined graph Ω that we describe in the following.

Let Γ be a minimal subtree graph with $n > 2$ number of vertices (subtrees); $n-2$ is the number of coding subtrees. Let Ω be a graph with n vertices, each vertex corresponding to a different subtree in Γ. We connect two vertices in Ω with an edge when the corresponding subtrees cannot be allocated the same coding vector.

More specifically, if two subtrees have a receiver node in common, they cannot be allocated the same coding vector. We connect the corresponding vertices in Ω with an edge which we call *receiver edge*. Similarly, if two subtrees have a common child, from Lemma 1 they cannot be allocated the same coding vector. We connect the corresponding vertices in Ω with an edge which we call a *flow edge*. Fig. 4 plots Ω for our example subtree graph.

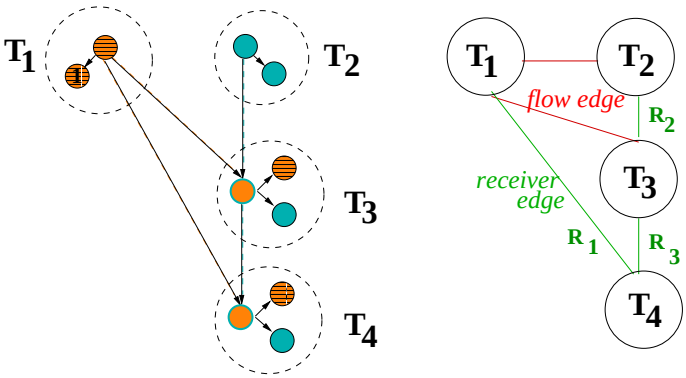

Fig. 4. Graph Γ and the associated graph Ω. Next to each receiver edge in graph Ω we denote the corresponding receiver.

A coloring is an assignment of colors to the vertices of Ω such that no two adjacent vertices have the same color. Thus, designing a valid netword code is equivalent to identifying a coloring for Ω.

IV. Algorithms for Coding

In the previous section we established that for the $h=2$ case, the algorithms for network code design are equivalent to algorithms for coloring the graph Ω. Thus, we can traslate all algorithms for coloring of Ω to algorithms for designing a network code for the corresponding minimal subtree graph Γ. In the following we briefly discuss some straightforward approaches.

Case 1: no information

Given the number of receivers N, we can upper bound the number of vertices of Ω as follows.

Lemma 3: For a multicast configuration with N receivers, the graph Ω has at most $N+1$ vertices.

Proof: For $h=2$ each receiver contributes two receiver nodes. For a minimal subtree graph each coding subtree contains at least two receiver nodes, and at least one of the source subtrees contains one receiver node. Since there exist exactly 2 source subtrees and at most $N-1$ coding subtrees, Ω has at most $N+1$ vertices. ■

Thus, if we use an alphabet of size $q=N$, we have $N+1$ available colors and we can assign to each vertex of Ω a different color. The corresponding algorithm (on the minimal subtree graph) would be to sequentially visit each subtree and assign to it one of the unused colors. This is a completely decentralized algorithm, since the color assigned to a subtree does not depend on the overall graph structure.

One of the main advantages of decentralized codes is that they do not have to be changed with the growth of the network as long as its subtree decomposition remains the same. In some cases even the codes which are not decentralized can remain the same, and the subtree decomposition method shows us how to ensure that. For more information, see [7].

Moreover, note that if we are employing a set A of coding vectors over a finite field F, and we need to increase the number of coding vectors to accomodate additional users, we can always add coding vectors to the set A from an extension field of F, and operate over the extension field.

Case 2: partial information

Having some information about the structure of the underlying graph can help reduce the number of colors employed and design new algorithms. The authors in [3] have derived alphabet size bounds in this direction. For example, if we know the number of vertices of Ω, we can use this number to upper

bound the number of colors we need in the previous algorithm.

Similarly, we may know what is the maximum number of receiver nodes inside a subtree, that is, what is the maximum number of receivers that observe the same coding vector. For the graph Ω, this quantity corresponds to $\Delta(\Omega)$, where $\Delta(\Omega)$ is defined as the maximum degree of its vertices, and the degree d_i of vertex i is the number of edges adjacent to it.

The *greedy coloring algorithm* ([9], pg.98) sequentially visits the vertices of the graph and colors each vertex with a color not already used to color any of its neighbors. This algorithm uses a maximum of $\Delta(\Omega)+1$ colors. Thus, the maximum alphabet size required would be $q=\Delta(\Omega)$.

V. Alphabet Size Bound

In the previous section we discussed algorithms where we have partial or no information about the underlying graph structure. If we have perfect knowledge of the graph structure we can calculate the exact number of colors we need. For example, if no network coding is required, a binary alphabet is sufficient. In this section we calculate an upper bound on the alphabet size a particular configuration may require.

We prove that an alphabet size proportional to $\sqrt{N}$ is always sufficient for any configuration with two sources and N receivers, that is, we will never need a larger alphabet size. This upper bound is tight in that there exist configurations that achieve it. The best previous result upper-bounded the required alphabet size by N [4].

To prove that an alphabet of size q is sufficient, we can equivalently prove (Lemma 2) that $k=q+1$ colors are sufficient to construct a coloring for Ω.

Lemma 4: For a minimal configuration with $n>2$, every vertex i in Ω has degree at least two, that is, $d_i = 2+d,$ for some $d\geq 0$.

Proof:

1) *Source subtrees:* If $n=3$, the two source subtrees have exactly one child which shares a receiver with each parent. If $n>3$, the two source subtrees have at least one child which shares a receiver or a child with each parent.
2) *Coding subtrees:* Each coding subtree has two parents. Since the configuration is minimal it cannot be allocated the same coding vector as any of its parents.This implies that in Ω there should exist edges between a subtree and its parents, that may be either flow edges, or receiver edges, and the corresponding vertex has degree at least two.

■

Lemma 5: ([10], chapter 8) Every k-chromatic graph has at least k vertices of degree at least $k-1$.

Theorem 1: For any minimal configuration with $h=2$ sources and N receivers, we can employ alphabet $\mathbb{F}_q$ of size

$$q \leq \lfloor \sqrt{2N-7/4}+1/2 \rfloor \tag{2}$$

This bound is tight, that is, there exist configurations that achieve it.

Proof: Assume that our graph Ω has n nodes and chromatic number $\chi(\Omega)=k\leq n$. Let $m=n-k$, where m is a nonnegative integer.

We are going to count the degree of the vertices in Ω in two different ways:

1) *Required degree* to have chromatic number k and a minimal configuration with n nodes. From Lemmas 4 and 5, we can lower bound the sum of the degree of the vertices of Ω as
$$\sum d_i \geq k(k-1)+(2+d)m, \tag{3}$$
for some $d\geq 0$.
2) *Provided degree* from the flow edges and the receiver edges. We have N receivers and $n-2$ coding subtrees, which implies that we have N receiver edges and $n-2$ flow edges. Thus
$$\sum d_i \leq 2(N+n-2) = 2(N+k+m-2). \tag{4}$$

From Equations (3) and (4) we get that

$$N \geq \frac{k(k-1)}{2} - k + 2 + dm. \tag{5}$$

This equation provides a lower bound on the number of receivers we need in order to have chromatic number k. Solving for $q=k-1$ to get the bound for $m=0$.

If $m=0$ then $n=k$ and Ω is a complete graph with $n=k=q+1$ vertices and $E(\Omega)=\frac{k(k-1)}{2}$ edges. We can construct such a configuration with $N=\frac{k(k-1)}{2}-k+2$ receivers and $k-2$ flow edges. Thus the bound is tight. ■

This bound offers a benchmark to evaluate the performance of different labeling algorithms with respect to the employed alphabet size.

VI. Application of other results

Once the connection with coloring is realized, a number of combinatorial results can be readily applied. We present here some of the most exciting ones, and refer the interested reader to (chapter 7, [11]) and [12] and the references therein.

A. Min-cut alphabet-size trade-off

The bound in Eq. (2) expresses the connection between required alphabet size and maximum possible number of users to accomodate. An underlying assumption of this bound is that the min-cut towards each user is exactly equal to the number of sources. We would expect that, if the min-cut towards some or all of the users is greater than the number of sources, a smaller alphabet size would be possible.

For the special case where the subtree graph is a bipartite graph, we can readily apply the following result.

Consider a set of points X and a family F of subsets of X. A coloring of the points X is legal if no subset of F is monochromatic. If a family admits a legal coloring with q colors then it is called q-colorable.

Theorem 2: (Erdös 1963) Let F be a family of sets each of size at least m. If $|F| < q^{m-1}$ then F is q-colorable.

In our case, X is the set of coding subtrees, m is the min-cut from the sources to each receiver, and each subset of F corresponds to the subtrees that a receiver observes. We want to find a coloring such that each receiver observes at least 2 different colors, i.e. has a basis of the 2-dimensional space. Theorem 2 tells us that by increasing the min-cut m we can accomodate the same number of users $N = |F|$ with a smaller alphabet size (alphabet size=$q-1$).

An algorithm for identifying a legal q-coloring can be found for example in [13].

B. Almost good codes

Again we consider the case where the subtree graph is bipartite. Assume that a legal coloring does not exist. The question here is, what is the maximum number of legally colored subsets that we can have.

Theorem 3: (chapter 19, [12]) For every m-uniform family F there exists a q-coloring of its points which colors at most $|F|q^{1-m}$ of the sets of F monochromatically.

A family of sets is m-uniform if all its members have size m. Thus if we have $|F| = N$ receivers, the min-cut to each receiver is m, and we use an alphabet of size $q-1$, at most Nq^{1-m} receivers will not be able to decode.

C. Structural Information

As discussed in Section IV, if we have some information about the structure of the underlying graph we should be able to derive bounds that apply to specific configurations. Again there is also a number of results in extremal combinatorics, such as the following theorem.

Theorem 4: (Erdös-Lovasz 1975) If every member of a m-uniform family intersects at most q^{m-3} other members, then the family is q-colorable.
Thus if the min-cut to each receiver is m, and every coding subtree is observed by at most $q^{m-3}+1$ receivers, then it is sufficient to use an alphabet of size $q+1$, irrespective of the number of receivers.

VII. Conclusions

In this paper we established a connection between network coding and coloring, used this connection to propose code design algorithms and alphabet size bounds, and pointed out a number of interesting results applicable in the area.

References

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

[2] R. W. Yeung S-Y. R. Li and N. Cai. Linear network coding. *IEEE Trans. Inform. Theory*, 49:371–381, February 2003.

[3] D. Ron M. Feder and A. Tavory. Bounds on linear codes for network multicast. Technical report, Electronic Colloquium on Computational Complexity, 2003.

[4] S. Egner P. Sanders and L. Tolhuizen. Polynomial time algorithms for network information flow. *Proc. 15th ACM Symposium on Parallel Algorithms and Architectures*, 2003.

[5] P. Chou S. Jaggi and K. Jain. Low complexity algebraic multicast network codes. citeseer.nj.nec.com/jaggi03low.html, 2003.

[6] R. Koetter and M. Medard. Beyond routing: an algebraic approach to network coding. *Proc. IEEE INFOCOM 2002*, 1, June 2002. New York.

[7] C. Fragouli and E. Soljanin. Subtree decomposition for network coding. Submitted to Transactions in Information Theory.

[8] N. Sloane and F. J. MacWilliams. *Error Correcting Codes*. North-Holland, 1998.

[9] R. Diestel. *Graph Theory*. Springer, 2000.

[10] J. A. Bondy and U. S. R. Murty. *Graph Theory with Applications*. 1979. Amsterdam: North-Holland.

[11] *Handbook of Combinatorics, vol. 1*. MIT Press, 1995.

[12] Stasys Jukna. *Extremal Combinatorics*. Springer, 2001.

[13] U. Manber. *Introduction to Algorithms: A Creative Approach*. Adison-Wesley, 1989.