Abstract—This work addresses the computational complexity of achieving the capacity of a general network coding instance. It has been shown [Lehman and Lehman, SODA 2005] that determining the “scalar linear” capacity of a general network coding instance is NP-hard.

In this work we address the notion of approximation in the context of both linear and non-linear network coding. Loosely speaking, we show that given an instance of the general network coding problem of capacity $C$, constructing a code of rate $\alpha C$ for any universal (i.e., independent of the size of the instance) constant $\alpha \leq 1$ is “hard”. Specifically, finding such network codes would solve a long standing open problem in the field of graph coloring.

Our results refer to scalar linear, vector linear, and non-linear encoding functions and are the first results that address the computational complexity of achieving the network coding capacity in both the vector linear and general network coding scenarios.

In addition, we consider the problem of determining the (scalar) linear capacity of a planar network coding instance (i.e., an instance in which the underlying graph is planar). We show that even for planar networks this problem remains NP-hard.

Index Terms—Network coding, index coding, complexity, approximation, capacity.

I. INTRODUCTION

In the network coding paradigm, internal nodes of the network may mix the information content of the received packets before forwarding them. This mixing (or encoding) of information has been extensively studied over the last decade (see e.g., [1]–[5] and references therein). While the advantages of network coding in the multicast setting are currently well understood, this is far from being the case in the context of general network coding. In particular, the complexity of determining the capacity of a general network coding instance is a central open problem, e.g., [6], [7].

Michael Langberg is with the Computer Science Division, the Open University of Israel, Raanana, 43107, Israel, email: mikel@openu.ac.il. Alex Sprintson is with the Department of Electrical and Computer Engineering, Texas A&M University, College Station, TX, 77843, USA, email: spalex@tamu.edu.

The work of Michael Langberg was supported in part by ISF grant 480/08 and grant 46114 of the Open University of Israel. The work of Alex Sprintson was supported by NSF grant CNS-0954153.

The material in this paper was presented at ISIT 2008, Toronto, Canada, June 2008.

This work addresses the computational complexity of designing network codes that achieve or come close to achieving the network capacity. An instance of the Network Coding problem is a directed graph $G = (V,E)$, a set of source nodes $\{s_i\} \subseteq V$, a set of terminal nodes $\{t_j\} \subseteq V$, and a set of source/terminal requirements $\{(s_i,t_j)\}$ (implying that terminal $t_j$ is interested in the information available at source $s_i$). In this paper we will consider acyclic graphs $G$, and follow the standard definitions appearing for example in [8]. Each source $s_i$ holds a message $p_i$ that is to be transmitted to a certain subset of terminals. Each message is assumed to consist of $k$ characters of a given finite alphabet $\Sigma$ (each character of $\Sigma$ will be referred to as a packet$^1$) and each edge of the network is assumed to have the capability of transmitting $\ell$ characters of $\Sigma$. We assume that each edge $e$ is used at most once, namely at most $\ell$ packets are transmitted over $e$. With each edge $e = (u,v)$ we associate an encoding function $g_e : \Sigma^{d_u \ell} \rightarrow \Sigma^{\ell}$ which ties the packets transmitted on the $d_u$ edges entering $u$ with the $\ell$ packets transmitted on $e$.

The objective is to define the encoding functions corresponding to edges in $E$ such that each terminal $t_i$ will be able to decode the messages it demands from the packets it receives on its incoming edges. More formally, we need to define for each terminal $t_i$ a decoding function $g_i : \Sigma^{d_i \ell} \rightarrow \Sigma^{r_i \ell}$ that enables $t_i$ to decode the messages it demands from the information transmitted on its incoming edges (here $d_i$ denotes the in-degree of $t_i$ and $r_i$ is the number of messages $t_i$ requires). If such encoding and decoding functions exist, we say that the instance $I$ of the Network Coding problem is $(k,\ell)$-solvable over $\Sigma$. If $\Sigma$ is a field and if the decoding and encoding functions are linear over $\Sigma$, we say that $I$ has a linear or vector linear $(k,\ell)$-solution over $\Sigma$. If the encoding and decoding functions are linear and $k = 1$ we say that the instance has a $(1,\ell)$-scalar linear solution over $\Sigma$.

Since, any $(k,\ell)$-solution implies an $(rk,\ell)$-solution for any integer $r$, we refer to the ratio $\frac{k}{\ell}$ as the rate of a $(k,\ell)$-solution to $I$ and denote the capacity $C(I)$

$^1$In practical applications, each packet contains several characters of $\Sigma$ to reduce overhead (see e.g., [9]). For clarity, in this paper we assume that each packet contains a single character of $\Sigma$. 

of $I$ over $\Sigma$ as the supremum of the ratio $\frac{C}{I}$ taken over all $(k, \ell)$ solutions to $I$ over $\Sigma$. Similarly, we define the vector linear capacity $C_{v}(I)$ and scalar linear capacity $C_{s}(I)$ of $I$ as the maximum rate achievable by vector-linear or scalar-linear solutions, respectively.\footnote{Notice that $C(I)$, $C_{v}(I)$ and $C_{s}(I)$ depend on the alphabet $\Sigma$. To simplify our presentation, we omit $\Sigma$ from our capacity notation.}

A. Previous work

The complexity of determining the capacity of a general Network Coding instance is a central open problem in the field of network coding. Specifically, it is currently not known whether this problem is solvable in polynomial time, is NP-hard, or maybe it is even undecidable [10] (the undecidability assumes that the alphabet size can be arbitrary and unbounded). It is shown in [11] that determining the scalar linear capacity $C_{s}$ is an NP-hard problem. However, it is not known whether this holds for the vector-linear or general capacity, as the result of [11] does not extend to $(k, \ell)$ vector linear codes even for $k = 2$. In [12], [13] it is shown that vector linear codes outperform scalar linear codes in terms of obtainable rate. Moreover, in [8] it is shown that nonlinear codes have an advantage over linear solutions as there exist instances in which linear codes do not suffice to achieve capacity.

For specific instances to the Network Coding problem, it has been shown that combining combinatorial bounds with “Shannon-type” information inequalities suffice to characterize the capacity, e.g., [14], [15], although this is not the case in general [13].

Finally, there are several connections between the capacity of general Network Coding instances and the family $\check{\Gamma}^{*}$ of entropic functions [16], [17]. The characterization of $\check{\Gamma}^{*}$ is an intriguing open problem. Reference [18] shows that determining the capacity region for the general Network Coding problem implies a characterization of $\check{\Gamma}^{*}$.

B. Our contribution

In this work we study the notion of approximation in the context of network coding. Namely, we show that given an instance $I$ to the Network Coding problem of capacity $C(I)$, constructing a code with rate at least $\alpha C(I)$ for any universal constant $\alpha \leq 1$ (independent of the size of the network) is “hard”. Specifically, finding such network codes would solve a long standing open problem in the field of graph coloring. Our results apply to instances with constant size alphabets, and hold for scalar linear, $(k, \ell)$ vector linear (for constant values of $k$), and general (not necessarily linear) encoding/decoding functions. Here a parameter is referred to as constant if its size is independent of the size of the given instance (i.e., the given network size). This implies the first hardness result for the general Network Coding problem for both vector linear and general (not necessarily linear) solutions.

We note that after the conference publication of this work [19], Yao and Verbin in [20] studied a different notion of approximation in the context of network coding. Namely, for any constant $\varepsilon$, given an instance of the Network Coding problem with $k$ terminals, they show that it is NP-hard to distinguish between the case in which there exists a network coding solution which satisfies $k^{1-\varepsilon}$ of the terminals and the case in which there does not exist any network coding solution that satisfies $k^{\varepsilon}$ of the terminals. Here a terminal is satisfied if it can decode the information it desires. This implies that estimating (i.e., approximating) the number of terminal nodes that can be satisfied in a given Network Coding instance (with $k$ terminals) beyond $k^{1-\varepsilon}$ is NP-hard. We stress that the techniques used in [20] and the problem they are estimating differ substantially from the work at hand.

In addition, we consider the problem of determining the scalar linear capacity of a planar instance (i.e., an instance in which the underlying graph is planar). We show that even if the network is planar, this problem remains NP-hard. We note that the reduction presented in [11] does not result in a planar graph. We now state our main results in detail. We begin with some preliminaries on the topic of “graph coloring”.

1) Graph coloring: An independent set in an undirected graph $G = (V, E)$ is a set of vertices that induce a subgraph which does not contain any edges. For an integer $k$, a $k$-coloring of $G$ is a function $\sigma : V \rightarrow [1 .. k]$ which assigns colors to the vertices of $G$. A valid $k$-coloring of $G$ is a coloring in which each color class is an independent set. The chromatic number $\chi(G)$ of $G$ is the smallest $k$ for which there exists a valid $k$-coloring of $G$. Finding $\chi(G)$ is a fundamental NP-hard problem. Hence, when limited to polynomial time algorithms, one turns to the question of estimating the value of $\chi(G)$ or to the closely related problem of approximate coloring in which one seeks to find a coloring of $G$ with $\alpha \cdot \chi(G)$ colors, for some approximation ratio $\alpha \geq 1$, where the objective is to minimize $\alpha$.

For a graph $G$ of size $n$, the approximate coloring of $G$ can be solved efficiently within an approximation ratio of $\alpha = O \left( \frac{n \log \log n}{\log n} \right)$ [21]. This result may seem rather weak — as a trivial approximation algorithm, with approximation ratio $n$, just colors each vertex with a different color. However, it turns out that one probably cannot do much better than the trivial algorithm. Namely, it is NP-hard to approximate $\chi(G)$ within a ratio of $n^{1-\varepsilon}$ for any constant $\varepsilon > 0$ [22].

In light of the above, there has been a long line of
work addressing the (easier problem) of coloring graphs $G$ which are known to have small chromatic number, e.g., [23]–[26]. For example (and most relevant to our work addressing the (easier problem) of coloring graphs with as few colors as possible has been extensively studied, e.g., [23], [24], and [26].

The current state of affairs in the study of coloring 3-colorable graphs is an intriguing one. Results in this area have one of two flavors: “achievability” results, which specify an efficient algorithm for coloring the given graphs, or “lower bounds”, which show that coloring these graphs with a small number of colors is a provably “hard” problem. On one hand, the currently best polynomial time algorithm [26] can color a 3-colorable graph $G$ in roughly $n^{0.21}$ colors — we stress that in this state of affairs there is still a polynomial blowup in the number of colors used! On the other hand, not much is known regarding lower bounds. It is NP-hard to color a 3-colorable graph $G$ with 4 colors [27], [28]. Under stronger complexity assumptions (related to the Unique Games Conjecture [29]) it is hard to color a 3-colorable graph with any constant number of colors $\alpha$ [30] (here $\alpha$ is a universal constant independent of the size of the input graph). Resolving this gap between the upper and lower bounds presented above is a long standing open problem.

In this work we show that achieving a constant approximation ratio for the Network Coding problem is at least as hard as coloring a 3-colorable graph with a constant number of colors, which, in turn, is at least as hard as the hardness assumptions specified in [30]. Regardless of the validity of the assumptions given in [30], our reduction shows that approximating the Network Coding problem within constant quality will solve a long standing and extensively studied open problem in approximate coloring.

2) Statement of results: Our main result is summarized by the following theorem. In what follows, and throughout the paper, an efficient algorithm is one that runs in time polynomial in the instance size. Moreover, a network coding solution, i.e., a collection of encoding/decoding functions, is considered to be a collection of efficient algorithms that compute the functions at hand. Throughout the paper, our alphabets $\Sigma$ are assumed to be finite.

Theorem 1: Let $\alpha < 1$, $k$, and $|\Sigma|$ be constants. If one can efficiently find a (not necessarily linear) $(k, \ell)$ solution to every instance $I$ of the Network Coding problem over $\Sigma$ of rate at least $\alpha C_{sl}(I)$, then one can efficiently color 3-colorable graphs with a constant number of colors.

We would like to note that in the theorem above, we state that it is “hard” to find $(k, \ell)$ network coding solutions that are greater than $\alpha$ times the scalar linear capacity of the instance at hand. As $C(I) \geq C_{sl}(I) \geq C_{ul}(I)$, this implies (a) that it is “hard” to find a (not necessarily linear) $(k, \ell)$ network coding solution of rate $\alpha C(I)$, and (b) that it is “hard” to find a linear $(k, \ell)$ network coding solution of rate $\alpha C(I)$.

A network coding instance is said to be planar if the underlying network can be drawn in the plane in such a way that no two edges cross each other. Planar graphs have seen a significant amount of research in the field of combinatorial optimization, e.g., [31], and many computational tasks known to be “hard” on general graphs become significantly easier when the input graph is planar. One may suspect that the same occurs in the case of network coding. However, in this work we show:

Theorem 2: Given a planar instance $I$ to the Network Coding problem and a finite field $\Sigma$, the problem of deciding whether $C_{sl}(I) \geq 1$ is NP-complete.

The instances $I$ implied by theorems 1 and 2 are simple in nature and resemble the instances that have been used in the literature to illustrate the advantage of network coding, e.g., [1], [32]. More specifically, given a graph $G$, the instances we construct correspond to the IndexCoding problem recently studied in [33]–[36]. The IndexCoding problem and its connection to network coding are described in Section II. We then turn to prove Theorem 1 and Theorem 2 in sections III and IV respectively.

II. PRELIMINARIES: THE INDEXCODING PROBLEM

The IndexCoding problem encapsulates the “source coding with side information” problem in which a single server wishes to communicate with several clients each having different side information. Formally, an instance to IndexCoding includes a set of clients $C = \{c_1, \ldots, c_n\}$ and a set of messages $P = \{p_1, p_2, \ldots, p_m\}$ to be transmitted by the server. Each client requires a certain subset of messages in $P$, while some messages in $P$ are already available to it. Specifically, each client $c_i \in C$ is associated with two sets:

- $W(c_i) \subseteq P$ - the set of messages required by $c_i$.
- $H(c_i) \subseteq P$ - the set of messages available at $c_i$.

We refer to $W(c_i)$ and $H(c_i)$ as the “wants” and “has” sets of $c_i$, respectively. The server uses a broadcast channel to transmit messages to clients, each message is an encoding of messages in $P$. We assume that all messages transmitted by the server are received by all clients without an error. The objective is to design an encoding scheme which minimizes the number of transmissions.

We consider a fractional setting in which each message $p_i \in P$ consists of $k$ packets $p_{i1}^1, \ldots, p_{ik}^k$ and
each packet is a character of a given alphabet \( \Sigma \). In each round of communication the server can transmit a single character of \( \Sigma \) (i.e., a single packet). The \( j \)’th round of communication is specified by an encoding function \( g_j : \Sigma^m \rightarrow \Sigma \). Namely, in the \( j \)’th round of communication the character \( x_j = g_j(P) \) is transmitted by the server, where \( P = \{ p_i^j \mid p_i \in P \} \).

The goal in the Index Coding problem is to find a set of encoding functions \( \Phi = \{ g_i \}_{i=1}^p \) that will allow each client to decode the messages it requested, while minimizing \( \ell = |\Phi| \). More formally, we need to define for each client \( c_i \) a decoding function \( \gamma_i : \Sigma^l \times (\Sigma^k)^{I(c_i)} \rightarrow (\Sigma^k)^{W(c_i)} \) that enables the client to decode the required messages in its “wants” set from the transmitted messages and the messages in its “has” set. If such encoding and decoding functions exist, we say that the instance to Index Coding is \((k, \ell)\)-solvable over \( \Sigma \), and \( \Phi \) is its solution. If \( \Sigma \) is a field and the encoding and decoding functions are linear we say that the instance has a \((k, \ell)\)-linear solution over \( \Sigma \).

As in the case of the Network Coding problem, any \((k, \ell)\)-solution implies an \((rk, rl)\)-solution for any integer \( r \). For an instance \( I \) to the Index Coding problem, we refer to the ratio \( \frac{k}{\ell} \) as the rate of a \((k, \ell)\)-solution to \( I \) and denote the capacity \( Opt(I) \) of an instance \( I \) to Index Coding over \( \Sigma \) as the supremum of the ratio \( \frac{k}{\ell} \) taken over \((k, \ell)\) solutions to \( I \) over \( \Sigma \). We also define by \( Opt_t(I) \) and \( Opt_s(I) \) the vector-linear and scalar-linear capacities, i.e., capacities that can be achieved by using vector linear and scalar linear solutions, respectively. As before, the capacities above may depend on \( \Sigma \). Nevertheless, we omit this dependence in our notations.

There is a natural reduction from the Index Coding problem to the problem of designing a network code for a certain network with general requirements. This connection can be summarized by the following Proposition.

**Proposition 3:** Let \( \Sigma \) be any alphabet. For every instance \( I \) to the Index Coding problem one can efficiently construct an instance \( \tilde{I} \) to the Network Coding problem such that any \((k, \ell)\)-solution to \( I \) over \( \Sigma \) can be efficiently converted to a \((k, \ell)\)-solution to \( \tilde{I} \) and vice-versa. These conversions preserve linearity and thus \( Opt_{sl}(I) = C_{sl}(\tilde{I}) \), \( Opt_s(I) = C_I(\tilde{I}) \) and \( Opt_t(I) = C_t(\tilde{I}) \).

The proof of Proposition 3 is well-known (see e.g., [35]). For completeness, we present the reduction in Figure 1.

---

Fig. 1. The instance \( \tilde{I} \) of Proposition 3. There is a source node \( p_j \) for each message in \( P \), a terminal node \( c_i \) for each client in \( C \), and two additional nodes \( u \) and \( v \). Each terminal \( c_i \) requires all messages from sources in \( W(c_i) \). Each \( p_j \) is connected to \( u \), \( u \) is connected to all terminals \( c_i \), and \( v \) is connected to \( u \). In addition we add edges \( (p_j, c_i) \) iff \( p_j \in H(c_i) \). The broadcast channel corresponds to packets transmitted on \((v, u)\), while the side information corresponds to packets transmitted on edges \((p_j, c_i)\).

The Index Coding problem was recently studied in [33], [34], [36] where special instances of Index Coding were considered. Namely, instances \( I \) in which \( |P| = |C| = n \) and the only message client \( c_i \) wants is the message \( p_i \) \( (W(c_i) = \{ p_i \}) \). In this case, the side information \( \{ H(c_i) \mid c_i \in C \} \) can be represented by a graph \( G = (C, E) \) with vertex set \( C \) such that \( G \) contains an edge \((i, j)\) if and only if client \( c_i \) has message \( p_j \) (i.e., \( p_j \in H(c_i) \)).

For instances \( I \) corresponding to undirected graphs \( G \) (in which edges are bi-directional) [33], [34] present certain connections between the combinatorial properties of \( G \) and the value of \( Opt(I) \). In order to present these connections, we first elaborate on an algebraic property of \( G \) referred to as the \( Minrank \) of \( G \) [37].

### A. Minrank

Let \( G = (V, E) \) be an undirected graph with \( |V| = n \). An \( n \times n \) matrix \( A = [a_{ij}] \) over \( \Sigma \) fits \( G \) if:

(a) \( \forall i, a_{ii} = 1 \),

(b) \( a_{ij} = 0 \) for all \( i, j \) such that \( (i, j) \notin E \).

Let \( \text{Rank}_\Sigma(A) \) denote the rank of a matrix \( A \) (over \( \Sigma \)). The \( Minrank \) of \( G \) over \( \Sigma \) is now defined to be the minimum value in the set \( \{ \text{Rank}_\Sigma(A) \mid A \text{ fits } G \} \).

There exist beautiful connections between combinatorial properties of \( G \) and the value of \( Minrank \). Specifically, in [37] it is shown that the value of \( Minrank \) is “sandwiched” between the maximum sized independent set in \( G \) (denoted by \( \alpha(G) \)) and the chromatic number of the complement graph \( G^c \) to \( G \) (here, an edge \((i, j)\) appears in \( G \) if and only if it does not appear in \( G^c \)). Namely, \( \chi(G^c) \geq Minrank(G) \geq \alpha(G) \).
For an undirected graph $G$, computing the $\text{Minrank}$ of $G$ was proven to be NP-complete in [38] via a reduction from the 3 coloring problem. Namely, given any undirected graph $G$ and any field $\Sigma$, [38] constructs an instance $\tilde{G}$ such that $\chi(\tilde{G}) \leq 3$ iff $\text{Minrank}(G) \leq 3$. This implies that a decision process for $\text{Minrank}$ will imply a decision process for 3 colorability. (To be precise the exact statement in [38] implies $\chi(G) = 3$ iff $\text{Minrank}(G) = 3$, however the slightly modified version above also follows from [38].)

As stated above, references [33] and [34] present certain connections between the $\text{Minrank}$ of a given undirected graph $G$ and the corresponding IndexCoding instance $I$. Specifically, they show that the scalar linear capacity $\text{Opt}_{sl}(I)$ is equal to $1/\text{Minrank}(G)$, which in turn is at least $1/\chi(G^c)$. Recall that $G^c$ is the complement graph of $G$.

Combining the above with the results of [38] stated earlier implies a reduction between the question of 3 colorability and that of deciding the capacity of IndexCoding instances. Namely, given $G$ one can efficiently construct a graph $\tilde{G}$ and a corresponding instance $I$ to the IndexCoding problem for which $\chi(\tilde{G}) \leq 3$ iff $\text{Minrank}(\tilde{G}) \leq 3$ iff $\text{Opt}_{sl}(I) \geq 1/3$. In other words, deciding whether $\text{Opt}_{sl}(I) \geq 1/3$ for a given instance $I$ to the IndexCoding problem will imply a decision process for 3 colorability (and thus deciding whether $\text{Opt}_{sl}(I) \geq 1/3$ is NP-hard).

Finally, we note that combining the discussion above with Proposition 3, one can establish that determining the scalar linear capacity of a general network coding instance is NP-hard. Notice that this may be viewed as an alternative proof to that given in [11]. We stress that the results in [38] do not imply inapproximability results of the nature presented in this work as in their reduction one can easily find a scalar linear solution which satisfies $\text{Opt}_{sl}(I) \geq 1/4$.

III. PROOF OF THEOREM 1

We begin by proving Theorem 4 below. Then, by combining Theorem 4 with Proposition 3, we obtain Theorem 1 stated in the Introduction. Recall that an efficient algorithm is one that runs in time polynomial in the instance size, and that a network coding (or index coding) solution, i.e., a collection of encoding/decoding functions, is considered to be a collection of efficient algorithms that compute the functions at hand. Throughout this section, the term $\text{poly}(\text{vars})$ refers to a polynomial in the variables $\text{vars}$ of constant degree and with constant coefficients (i.e., a polynomial whose coefficients and degrees of its variables are independent of the variables $\text{vars}$).

**Theorem 4:** Given any 3-colorable graph $G(V, E)$, one can construct in time $\text{poly}(|V|)$ an instance $I$ to the IndexCoding problem with the following properties:

(a) For any alphabet $\Sigma$, if for $\alpha \in (0, 1]$ one can find in time $T$ a $(k, \ell)$ solution to $I$ over $\Sigma$ that satisfies $\frac{k}{\ell} \geq \alpha \text{Opt}_{sl}(I)$ then one can color $G$ with $|\Sigma|^{3k/\alpha}$ colors in time $T + \text{poly}(|\Sigma|^{k/\alpha})|G|$.  

(b) For any field $\Sigma$, if for $\alpha \in (0, 1]$ one can find in time $T$ a linear $(k, \ell)$-solution to $I$ over $\Sigma$ that satisfies $\frac{k}{\ell} \geq \alpha \text{Opt}_{sl}(I)$ then one can color $G$ with $|\Sigma|^{3k/\alpha}$ colors in time $T + \text{poly}(|\Sigma|^{k/\alpha})|G|$.  

**Proof:** Let $G = (V, E)$ be an undirected graph with $V = \{v_1, \ldots, v_n\}$. We start by defining a corresponding instance $I$ to IndexCoding (as explained in Section II). The instance includes $n$ clients $C = \{c_1, \ldots, c_n\}$ and $n$ messages $P = \{p_1, \ldots, p_n\}$. For each client we define $W(c_i) = \{p_i\}$ and $H(c_i) = \{p_j \mid (i, j) \in E\}$, i.e., client $c_i$ wants message $p_i$ and has all messages wanted by its neighbors in $G$. As we will be discussing $(k, \ell)$ solutions to $I$, we consider each message $p_i = p_{i1}^1, \ldots, p_{ik}^k$ to lie in $\Sigma^k$ and denote the $kn$ packets $\{p_i^j\}$ as $P_i$.

Assume that the complement graph $G^c$ of $G$ is 3-colorable (notice that the role of $G$ in the theorem statement is played here by $G^c$). It follows that $\text{Opt}_{sl}(I) \geq \frac{1}{3}$. Indeed, as $G^c$ is 3-colorable, $G$ can be covered by 3 cliques\(^5\). For each such clique $C$, define the corresponding transmitted packet $x_C = \sum_{v \in C} p_v$.

To prove part (a) of our theorem assume that we have an $\alpha$ approximation algorithm for constructing solutions to instances $I$ of IndexCoding. That is, we can find a $(k, \ell)$-solution for $I$ with $\frac{k}{\ell} \geq \alpha \text{Opt}_{sl}(I) \geq \frac{\ell}{3}$. This, in turn, implies that $\ell \leq 3k/\alpha$.

For $j = 1, \ldots, \ell$ let $g_j : \Sigma^{kn} \to \Sigma$ characterize the solution to $I$ in which the packet $x_j$ transmitted in communication round $j$ is equal to $g_j(P)$; and for $i = 1, \ldots, n$ let $\gamma_i : \Sigma^\ell \times (\Sigma^k)^{|H(c_i)|} \to \Sigma^k$ be the decoding function of client $c_i$.

We will now show, given the set $\{g_j\}_{j=1}^\ell$ and $\{\gamma_i\}_{i=1}^n$, how to construct a coloring of $G^c$ of size $|\Sigma| |E|^{\ell} \leq |\Sigma| |E|^{3k/\alpha}$. This will suffice to prove part (a) of our theorem. To this end, we will define a partition of the set $\{\gamma_i\}_{i=1}^n$ such that any two functions $\gamma_i$ and $\gamma_j$ that belong to the same group in the partition will have corresponding vertices that share an edge in $G$ (and thus any group will form a clique).

To obtain our partition we will need the following projected version of the decoding functions $\{\gamma_i\}$. Roughly speaking, we will restrict our decoding functions $\gamma$ to a predefined side information value and then only consider the first output of $\gamma$ (out of $k$ outputs). More

\(^5\)Namely, the vertex set of $G$ can be partitioned into three subsets, each comprising of a clique. Here, as is common in graph theory, a clique $C$ is a subset of vertices in which each pair are connected by an edge.

\(^6\)Here, we refer to $\Sigma$ as an additive group and take the summation over this group structure. We note that any alphabet can be mapped to such a group.
specifically, let $\sigma$ be an arbitrary element in $\Sigma$. For each $\gamma_i : \Sigma^\ell \times (\Sigma^k)^{|H(c_i)|} \rightarrow \Sigma^k$ we define a corresponding function $\tilde{\gamma}_i : \Sigma^\ell \rightarrow \Sigma$ as follows: For every $x \in \Sigma^\ell$ $\tilde{\gamma}_i(x)$ is defined to be equal the first (out of the $k$) output elements of $\gamma_i(x, \sigma^{|H(c_i)|})$.

We are now ready to partition the set $\{\gamma_1, \ldots, \gamma_n\}$ into groups based on the functions $\tilde{\gamma}_i$: $\gamma_i$ and $\gamma_j$ will belong to the same group of the partition iff their projected versions $\tilde{\gamma}_i$ and $\tilde{\gamma}_j$ are equal. As there are only $|\Sigma|^{2\ell}$ possibly different functions $\tilde{\gamma}_i$, we have a partition of at most this size.

It is left to show that vertices $v_i$ and $v_j$ corresponding to $\tilde{\gamma}_i$ and $\tilde{\gamma}_j$ that belong to the same group indeed share an edge in $G$. Assume by way of contradiction that vertices $v_i$ and $v_j$ do not share an edge in $G$, namely $p_j \notin H(c_1)$ and $p_j \notin H(c_2)$. By our definitions, $p_j \notin H(c_1)$ and $p_j \notin H(c_2)$.

Now, consider the case in which all packets in $P$ are set to $\sigma$, except for the packet $p_i^j$ which is set to a value different from $\sigma$. This implies that $p_i^j \neq p_j^j$ (the messages $p_i$ and $p_j$ differ in their first packet). Note that since the side information $H(c_i)$ of $c_i$ consists of messages in $P \setminus \{p_i, p_j\}$, it contains $|H(c_i)|$ messages of value $\sigma^k$. Similarly, since the side information $H(c_j)$ in this case consists of messages in $P \setminus \{p_i, p_j\}$, it contains $|H(c_j)|$ messages of value $\sigma^k$.

Let $x = x_1, \ldots, x_\ell$ where $x_\ell = g_i(P)$ be the transmitted symbols in this case. Since $\gamma_i$ and $\gamma_j$ are decoding functions, it holds that $p_i = \gamma_i(x, (\sigma^k)|H(c_i)|)$ and $p_j = \gamma_j(x, (\sigma^k)|H(c_j)|)$. By our definition of $\tilde{\gamma}_i$, this implies that $p_i^j = \tilde{\gamma}_i(x)$ and $p_j^j = \tilde{\gamma}_j(x)$. However, $\gamma_i \equiv \gamma_j$ (as we assumed that $\gamma_i$ and $\gamma_j$ are in the same group) and thus $p_i^j = \tilde{\gamma}_i(x) = \tilde{\gamma}_j(x) = p_j^j$. This contradicts the fact that $p_i^j \neq p_j^j$, and, in turn, our assumption that vertices $v_i$ and $v_j$ do not share an edge in $G$.

To complete the proof of part (a) we need to address the computational aspects of the reduction. The construction of $I$ can be done in time $\text{poly}(n)$. Assume that the $(k, \ell)$ solution $\{g_j\}_{j=1}^n$, $\{\gamma_i\}_{i=1}^m$ to $I$ is found in time $T$. To obtain the desired coloring, one must partition the set $\{\gamma_i\}_{i=1}^m$ accordingly. This can be done in time $\text{poly}(nk \log |\Sigma|)|\Sigma|^\ell$ by exhaustively checking all inputs $x \in \Sigma^\ell$ to $\tilde{\gamma}_i$. All in all, we obtain the asserted running time.

We now turn to address vector linear solutions (part (b) of the theorem). The proof in this case is identical to that presented above with the exception that, for a field $\Sigma$, the number of linear functions $\{\gamma_i\}$ is now $|\Sigma|^\ell$ instead of $|\Sigma|^{2\ell}$.

IV. PROOF OF THEOREM 2

We now turn to prove Theorem 2. Our starting point is a reduction discussed previously from a given undirected graph $G$ to an instance $I$ of the IndexCoding problem that satisfies $\chi(G) \leq 3$ iff $\text{Opt}_{s,v}(I) \geq 1/3$. In what follows, we present a reduction from the instance $I$ to a planar instance $I_p$ of the Network Coding problem which satisfies $C_{\text{sat}}(I_p) \geq 1$ if and only if $\text{Opt}_{s,v}(I) \geq 1/3$. This will suffice to prove Theorem 2.

The instance $I$ above includes a set of $n$ clients $C = \{c_1, \ldots, c_n\}$ and $m$ messages $P = \{p_1, \ldots, p_m\}$. Moreover, as before, each client $c_i$ only wants the single message $p_i$, i.e., $W(c_i) = \{p_i\}$.

We proceed to construct an instance $I_p$ to the Network Coding problem. We begin by describing the structure of the underlying planar network $G_p(V, E)$ (schematically depicted in Figure 2). The set of nodes $V$ of $G_p$ includes a node $c_i$ for each client $c_i \in C$, and two additional vertices, $s$ and $v$. In addition, for each client $c_i \in C$ we add $|H(c_i)|$ vertices $h_i^1, \ldots, h_i^{|H(c_i)|}$ that correspond to elements of $H(c_i)$. The set of edges $E$ of $G_p$ is constructed as follows. First, we connect $s$ to $v$ by three parallel edges. Next, $v$ is connected to each node $c_i$ by three parallel edges. For each client $c_i \in C$ we connect each vertex in $\{h_i^1, \ldots, h_i^{|H(c_i)|}\}$ to vertex $c_i$ by a single edge. Finally, the node $s$ is connected to the nodes that correspond to $\{H(c_i) \mid c_i \in C\}$.

The instance includes a set of $n$ sources $\{s_i\}_{i=1}^n$ that correspond to messages in $P$ (source $s_i$ having message $p_i$). All sources are co-located at vertex $s$. For each client $c_i \in C$ we add $1 + |H(c_i)|$ terminals such that one terminal is located at node $c_i$ and requires packet $p_i$ from source $s_i$ (this corresponds to the message $p_i$ wanted by $c_i$ in $I$). The rest of the terminals are located at vertices $\{h_i^1, \ldots, h_i^{|H(c_i)|}\}$. Terminal $h_j^k$ will require the $j$’th message in the set $H(c_i)$ (from the corresponding source). The latter terminals will force the sources to transmit the “side information” of $I$ on the edges $(s, h_i^k)$.
Clearly, \( G_p(V, E) \) is planar. We now show that any (scalar linear) \((1,3)\)-code for \( I \) corresponds to a (scalar linear) \((1,1)\)-network code for \( I_p \). In addition, we show that any (scalar linear) \((1,1)\)-network code for \( I_p \) corresponds to a (scalar linear) \((1,3)\)-solution for \( I \). This will complete the proof of the theorem, as we have established (for any \( \Sigma \)) the following connection: \( \chi(G) \leq 3 \) iff \( \text{Opt}_{c,s}(I) \geq 1/3 \) iff \( C_{sl}(I_p) \geq 1 \).

Let \( \Sigma \) be any field. Consider any linear (1,3)-code \( \{g_1, g_2, g_3\} \) for \( I \) in which message \( x_i \) transmitted in communication round \( i \) is equal to \( \langle g_i, \bar{x} \rangle \). Here \( g_i \in \Sigma^n \) represents the linear function transmitted in round \( i \), and \( \bar{x} = p_1, \ldots, p_n \). We now present a scalar-linear (1,1) network code for \( I_p \). The network code is very natural. The (multiple) source \( s \) transmits \( x_1, x_2, x_3 \) on the three parallel edges connecting \( s \) and \( v \). The node \( v \) then forwards these messages to each node \( c_i \). In addition, the source \( s \) transmits message \( p \) to terminal \( h_i \) if and only if \( j \)’th message in the set \( H(c_i) \) is \( p \). Finally, nodes \( h_i \) forward their information to the corresponding nodes \( c_i \). Clearly, the terminals \( h_i \) are satisfied. To see that the terminals \( c_i \) (for \( i = 1, \ldots, n \)) are satisfied, we notice that the information reaching \( c_i \) in \( I_p \) is exactly that available to client \( c_i \) after the broadcast of \( x_1, x_2, x_3 \). As the latter allows \( c_i \) to recover packet \( p_i \), so does the former.

The opposite direction is very similar. Consider any (1,1) linear network coding solution to \( I_p \). In this solution it must hold that message \( p \) is transmitted on edge \( (s, h_i) \) iff \( j \)’th message in the set \( H(c_i) \) is \( p \) (otherwise, terminal \( h_i \) would not be satisfied). Let \( x_1, x_2, x_3 \) be the information transmitted on the three parallel edges connecting \( s \) and \( v \). Each \( x_i \) is some linear function of the (multiple) source information \( P \). The (1,3) code for \( I \) will consist of exactly the transmitted information \( x_1, x_2, x_3 \). Thus, each \( c_i \) in \( I \) receives the information \( x_1, x_2, x_3 \) and all packets in \( H(c_i) \). It is left to show that this suffices to decode \( p_i \). This follows simply from the fact that in \( I_p \) each terminal \( c_i \) can decode \( p_i \) from the information transmitted from nodes \( h_i \) and \( v \). Moreover, the information transmitted from \( h_i \) is a linear function of the packets in \( H(c_i) \), and the information transmitted from \( v \) is a linear function of \( x_1, x_2, x_3 \). As the packets of \( H(c_i) \) and the information \( x_1, x_2, x_3 \) are present at client \( c_i \) of \( I \), it can use the functions specified above to decode \( p_i \).

V. CONCLUSION

In this work we focus on approximating the capacity of general Network Coding instances. For constant values of \( \alpha \), \( k \), and \( |\Sigma| \) we show that given an instance \( I \), efficiently constructing a \((k, \ell)\) network code with \( k^{\ell} \geq \alpha C(I) \) will solve a long standing open problem in graph coloring. This is the first study that addresses the computational complexity of finding (almost) optimal vector linear or non-linear network coding solutions. In addition, we show that the task of determining the scalar linear capacity for Network Coding instances remains NP-hard even if the underlying network is planar.

The considered problems present significant challenges and provide fertile ground for future research. In particular, two major questions remain open. Can one prove similar inapproximability results when \( k \) or \( |\Sigma| \) are not necessarily constant? Can one base the conditional inapproximability on any NP-complete problem?

VI. ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees for their helpful comments. Specifically, the main result of this work (Theorem 1) was originally proven by the authors for vector linear solutions alone (Theorem 4(b)) and due to the remarks of an anonymous referee the proof was changed to hold for general encoding/decoding functions also.

REFERENCES

9


