The Serializability of Network Codes

Anna Blasiak Robert Kleinberg †

Abstract

Network coding theory studies the transmission of information in networks whose vertices may perform nontrivial encoding and decoding operations on data as it passes through the network. A solution to a network coding problem is a specification of a coding function on each edge of the network. This specification is subject to constraints that ensure the existence of a protocol by which the messages on each vertex's outgoing edges can be computed from the data it received on its incoming edges. In directed acyclic graphs it is clear how to verify these causality constraints, but in graphs with cycles this becomes more subtle because of the possibility of cyclic dependencies among the coding functions. Sometimes the system of coding functions is *serializable* — meaning that the cyclic dependencies (if any) can be "unraveled" by a protocol in which a vertex sends a few bits of its outgoing messages, waits to receive more information, then send a few more bits, and so on — but in other cases, there is no way to eliminate a cyclic dependency by an appropriate sequencing of partial messages. How can we decide whether a given system of coding functions is serializable? When it is not serializable, how much extra information must be transmitted in order to permit a serialization? Our work addresses both of these questions. We show that the first one is decidable in polynomial time, whereas the second one is NP-hard, and in fact it is logarithmically inapproximable.

[1]Department of Computer Science, Cornell University, Ithaca NY 14853. E-mail: ablasiak@cs.cornell.edu. Supported an by an NDSEG Graduate Fellowship, an AT&T Labs Graduate Fellowship, and an NSF Graduate Fellowship.

[2]Department of Computer Science, Cornell University, Ithaca NY 14853. E-mail: rdk@cs.cornell.edu. Supported by NSF grant CCF-0729102, a Microsoft Research New Faculty Fellowship, and an Alfred P. Sloan Foundation Fellowship.

1 Introduction

Network coding theory studies the transmission of information in networks whose vertices may perform nontrivial encoding and decoding operations on data as it passes through the network. More specifically, a network code consists of a network with specified sender and receiver edges and coding functions on each edge. The classic definition of a network code requires that each vertex can compute the message on every outgoing edge from the messages received on its incoming edges, and that each receiver is sent the message it requires. In directed acyclic graphs, a network code that satisfies these requirements specifies a valid communication protocol. However, in graphs with cycles this need not be the case; the definition does not preclude the possibility of cyclic dependencies among coding functions. Therefore, in graphs with cycles we also require that a network code is *serializable*, meaning it correctly summarizes a communication protocol in which symbols are transmitted on edges over time, and each symbol transmitted by a vertex is computed without knowledge of information it will receive in the future. The present paper is devoted to the study of characterizing the constraint of serializability.

Motivation. The central question in the area of network coding is to determine the amount by which coding can increase the rate of information flow as compared to transmitting information without coding. One of the most important open problems in network coding, the *undirected k-pairs conjecture*, states that in undirected graphs with k sender-receiver pairs, coding cannot increase the maximum rate of information flow; that is, the network coding rate is the same as the multicommodity flow rate. Apart from its intrinsic interest, the conjecture also has important complexity-theoretic implications: for example, if true, it implies an affirmative answer to a 20-year-old conjecture regarding the I/O complexity of matrix transposition [1]. In order to answer this question in the affirmative we need to find upper bounds on the network coding rate. On graphs with cycles, we must understand how the condition of serializability restricts the set of feasible codes in order to find tight upper bounds.

Almost all efforts to produce upper bounds on the network coding rate have focused on the following construction. We regard each edge of the network as defining a random variable on a probability space and then associate each set of edges with the Shannon entropy of the joint distribution of their random variables. This gives us a vector of non-negative numbers, one for each edge set, called the *entropic vector* of the network code. The closure of the set of entropic vectors of network codes forms a convex set, and network coding problems can be expressed as optimization problems over this set [10]. In much previous work, tight upper bounds have been constructed by combining the constraints that define this convex set. Thus, one might hope that the serializability of a network code is equivalent to a set of constraints on its entropic vector. In this paper we show that this is not the case, and we explore alternative characterizations of serializability and their implications.

Our contributions. Our work is the first systematic study of criteria for serializability of network codes. We find that serializability cannot be detected solely from the entropic vector of the network code; a counter-example is given in Section 3. Nevertheless, in Section 4 we give a polynomial-time algorithm to decide the serializability of a network code. Associated to this decision problem is a natural optimization problem: given a set of coding functions, what is the minimum number of extra bits of information that must be sent in order to make the given coding functions serializable? We call this parameter the *serializability deficit*, and in Section 5 we prove that computing it is NP-hard, and moreover, that the serializability deficit is logarithmically inapproximable. We also prove a surprising subadditivity phenomenon involving the serializability deficit: the serializability

deficit for p parallel executions of a network code may be much less than p times as great as the serializability deficit of the original code. In fact, for any $\delta > 0$ there exists a network code whose p-fold parallel repetition has a serializability deficit less than δp times the serializability deficit of the original code.

Beyond providing an algorithm for deciding if a network code is serializable, our work provides important insights into the property of serializability. In Section 4 we define a certificate that we call a *non-trivial information vortex* whose existence is a necessary and sufficient condition for non-serializability. For linear network codes, an information vortex consists of linear subspaces of the dual of the message space. For general network codes, it consists of Boolean subalgebras of the power set of the message set. We prove a number of theorems about information vortices that suggest their role in the theory of network coding may be similar to the role of fractional cuts in network flow theory. In particular, we prove a type of min-max relation between serializable codes and information vortices: under a suitable definition of *serializable restriction* it holds that every network code has a unique maximal serializable restriction, a unique minimal information vortex, and these two objects coincide. Information vortices also play a key role in our theorems about serializability deficits. For example, in Section 6 we show that information vortices allow us to prove limits on the severity of the subadditivity phenomenon noted above: for every non-serializable linear code Φ, the serializability deficit of p parallel executions of Φ grows as $\Omega(p)$, where the constant inside the $\Omega(\cdot)$ depends on Φ.

As mentioned earlier, one of the motivations for our work is the objective of characterizing the information inequalities (i.e., constraints on the entropic vector) implied by serializability. While our algorithm for deciding serializability does not lead directly to such a characterization, we are able to provide one for the special case in which the underlying network is a 2-cycle. In Section 7, we present four inequalities and show that any entropic vector satisfying those inequalities as well as Shannon's inequalities can be realized by a serializable network code. (Though structurally simple, the 2-cycle graph has been an important source of inspiration for information inequalities in prior work, including the crypto inequality [6], the informational dominance bound [4], and the Chicken and Egg inequality [5].) Extending this characterization beyond the 2-cycle to general graphs is the most important open question left by our work.

Related work. For a general introduction to network coding we refer the reader to [9, 11]. There is a standard definition of network codes in directed acyclic graphs (Definition 2.1 below) but in many papers on graphs with cycles the definition is either not explicit (e.g. [2]) or is restricted to special classes of codes (e.g. [3]). Precise and general definitions of network codes in graphs with cycles appear in [7, 1, 4] and the equivalence of these definitions (modulo some differing assumptions about nodes' memory) is proven in [9]. The definition for serializability that we set forth in Section 2 was used, but never formally defined, in [5]. In its essence it is the same as the "graph over time" definition given in [9] but requires less cumbersome notation.

Although our work is the first to give precise necessary and sufficient conditions for serializability, several prior papers gave necessary conditions based on information inequalities. Large classes of such inequalities in graphs with cycles were discovered independently by Jain et al. [6], Kramer and Savari [8], and Harvey et al. [4]. These go by the names *crypto inequality*, *PdE bound*, and *informational dominance bound*, respectively. In various forms, all of them describe a situation in which the information on one set of edges completely determines the information on another set of edges. A more general necessary condition for serializability was presented in a recent paper by Harvey et al. [5]; we will henceforth refer to this information inequality as the *Chicken and Egg* inequality; see Theorem 7.1. As far as we are aware, our work is the first to consider the question of whether this set of inequalities (or any set of information inequalities) provides a complete

characterization of serializability.

2 Definitions

We define a network code to operate on a directed multigraph we call a *sourced graph*, denoted $G = (V, E, S)$.[1] S is a set of special edges, called *sources* or *source edges*, that have a head but no tail. We denote a source with head s by an ordered pair $(\bullet, s)$. Elements of $E \cup S$ are called *edges* and elements of E are called *ordinary edges*. For a vertex v, we let $\mathsf{In}(v) = \{(u, v) \in E\}$ be the set of edges whose head is v. For an edge $e = (u, v) \in E$, we also use $\mathsf{In}(e) = \mathsf{In}(u)$ to denote the set of *incoming edges* to e.

A network code in a sourced graph specifies a protocol for communicating symbols on error-free channels corresponding to the graph's ordinary edges, given the tuple of messages that originate at the source edges.

Definition 2.1 A *network code* is specified by a 4-tuple $\Phi = (G, \mathfrak{M}, \{\Sigma_e\}_{e \in E \cup S}, \{f_e\}_{e \in E \cup S})$ where $G = (V, E, S)$ is a sourced graph, $\mathfrak{M}$ is a set whose elements are called *message-tuples*, and for all edges $e \in E \cup S$, Σ_e is a set called the *alphabet* of e and $f_e : \mathfrak{M} \to \Sigma_e$ is a function called the *coding function* of e. If e is an edge and $e_1, \ldots, e_k$ are the elements of $\mathsf{In}(e)$ then the value of the coding function f_e must be completely determined by the values of $f_{e_1}, \ldots, f_{e_k}$. In other words, there must exist a function $g_e : \prod_{i=1}^{k} \Sigma_{e_i} \to \Sigma_e$ such that for all $m \in \mathfrak{M}$, $f_e(m) = g_e(f_{e_1}(m), \ldots, f_{e_k}(m))$.

In graphs with cycles a code can have cyclic dependencies so Definition 2.1 does not suffice to characterize the notion of a valid network code. We must impose a further constraint that we call *serializability*, which requires that the network code summarizes a complete execution of a communication protocol in which every bit transmitted by a vertex depends only on bits that it has already received.

Below we define serializability formally using a definition implicit in [5].

Definition 2.2 A network code Φ is serializable if for all $e \in E$ there exists a set of alphabets $\Sigma_e^{(1\ldots k)} = \left\{\Sigma_e^{(1)}, \Sigma_e^{(2)}, \ldots, \Sigma_e^{(k)}\right\}$ and a set of functions $f_e^{(1\ldots k)} = \left\{f_e^{(1)}, f_e^{(2)}, \ldots, f_e^{(k)}\right\}$ such that

1. $f_e^{(i)} : \mathfrak{M} \to \Sigma_e^{(i)}$,
2. $\forall\, m_1, m_2 \in \mathfrak{M}$, if $f_e(m_1) = f_e(m_2)$, then $\forall i,\ f_e^{(i)}(m_1) = f_e^{(i)}(m_2)$,
3. $\forall\, m_1, m_2 \in \mathfrak{M}$, if $f_e(m_1) \neq f_e(m_2)$, then $\exists i,\ f_e^{(i)}(m_1) \neq f_e^{(i)}(m_2)$, and
4. $\forall\, m \in \mathfrak{M}$, $e \in E$, $j \in \{1 \ldots k\}$ there is some function $h_e^{(j)}$ such that

$$f_e^{(j)}(m) = h_e^{(j)}\left(\prod_{\hat{e} \in \mathsf{In}(e)} f_{\hat{e}}^{(1..j-1)}\right).^{2}$$

We call such a $\Sigma_e^{(1\ldots k)}, f_e^{(1\ldots k)}$ a *serialization* of Φ.

The function $f_e^{(i)}$ describes the information sent on edge e at time step i. Item 2 requires that together the functions $f_e^{(1..k)}$ send no more information than f_e and Item 3 requires that $f_e^{(1..k)}$

[1] In prior work it is customary for the underlying network to also have a special set of receiving edges. Specifying a special set of receivers is irrelevant in our work, so we omit them for convenience, but everything we do can be easily extended to include receivers.

[2] Throughout this paper, when the operator $\prod$ is applied to functions rather then sets we mean it to denote the operation of forming an ordered tuple from an indexed list of elements.

sends at least as much information as f_e. Item 4 requires that we can compute $f_e^{(j)}$ given the information sent on all of e's incoming edges at previous time steps.

In working with network codes, we will occasionally want to compare two network codes Φ, Φ' such that Φ' "transmits all the information that is transmitted by Φ." In this case, we say that Φ' is an extension of Φ, and Φ is a restriction of Φ'.

Definition 2.3 Suppose that $\Phi = (G, \mathfrak{M}, \{\Sigma_e\}, \{f_e\})$ and $\Phi' = (G, \mathfrak{M}, \{\Sigma'_e\}, \{f'_e\})$ are two network codes with the same sourced graph G and the same message set $\mathfrak{M}$. We say that Φ is a restriction of Φ', and Φ' is an extension of Φ, if it is the case that for every $m \in \mathfrak{M}$ and $e \in E$, the value of $f'_e(m)$ completely determines the value of $f_e(m)$; in other words, $f_e = g_e \circ f'_e$ for some function $g_e : \Sigma'_e \to \Sigma_e$.

The entropic vector of a network code gives a non-negative value for each subset of a network code. The value of an edge set F is the Shannon entropy of the joint distribution of the random variables associated with each element of F, as is formalized in the following definition.

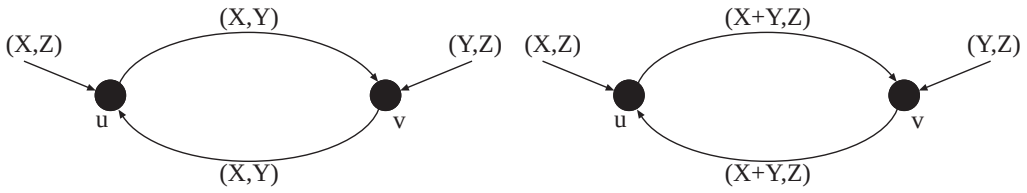
Definition 2.4 Given a network code $\Phi = (G, \mathfrak{M}, \{\Sigma_e\}, \{f_e\}), G = (V, E, S)$, the *entropic vector* of Φ has coordinates $H(F)$ defined for each edge set $F = \{e_1, \ldots, e_j\} \subseteq E \cup S$ by:

$$H(F) = H(e_1 e_2 \ldots e_j) = \sum_{x_1 \in \Sigma_{e_1}, x_2 \in \Sigma_{e_2}, \ldots, x_j \in \Sigma_{e_j}} -p(x_1, x_2, \ldots, x_j) \log(p(x_1, x_2, \ldots, x_j)),$$

where the probabilities are computed assuming a uniform distribution over $\mathfrak{M}$.

3 Serializability and Entropy Inequalities

Constraints imposed on the entropic vector alone suffice to characterize serializability for DAGs, but, the addition of one cycle causes the entopic vector to be an insufficient characterization. We show that the entropic vector is not enough to determine serializability even on the 2-cycle by giving a serializable and non-serializable code with the same entropic vector.



(a) Serializable (b) Not Serializable

Figure 1: Two network codes with the same entropy function.

The two codes illustrated in Figure 1 apply to the message tuple (X, Y, Z), where X, Y, Z are uniformly distributed random variable over $\mathbb{F}_2$. It is easy to check that the entropy of every subset of corresponding source and edge functions is the same, and thus the codes have the same entropic vector. The code in Figure 1(a) is clearly serializable: at time step one we send X on (u, v) and Y on (v, u); then Y on (u, v) and X on (v, u). On the other hand, the code in Figure 1(b) is not serializable because, informally, to send $X + Y$ on the top edge requires that we already sent $X + Y$ on the bottom edge, and vice versa. A formal proof that the code in Figure 2(b) is not serializable can be obtained by applying the characterization of serializability in Theorem 4.6.

4 A Characterization of Serializability

4.1 Linear Codes

A characterization of serializability for linear network codes is simpler than the general case because it relies on more standard algebraic tools. Accordingly, we treat this case first before moving on to the general case. Throughout this section we use $\mathfrak{M}^*$ to denote the dual of the message space over a field $\mathbb{F}$. Additionally, we use $T_e \subseteq \mathfrak{M}^*$ to denote the row space of the matrix representing the linear transformation f_e. (It makes sense to talk about such a matrix because we can pick a basis for $\mathfrak{M}$ and each Σ_e.)

Though it is impossible to characterize the serializability of a network code in terms of its entropic vector, computationally there is a straightforward solution. In polynomial time we can either determine a serialization for a code or show that no serialization exists using the obvious algorithm: try to serialize the code by "sending new information when possible." When we can no longer send any new information along any edge we terminate. If we have sent all the information required along each edge, then the greedy algorithm finds a serialization; otherwise, we show that no serialization exists by presenting a succinct certificate of non-serializability. Though our algorithm is straightforward, we believe that the change in mindset from characterizing codes in terms of the entropic vector is an important one, and that our certificate of non-serializability (see Definition 4.1) furnishes an effective tool for addressing other questions about serializability, as we shall see in later sections.

Before giving a more formal description of the algorithm, we would like to remark on one potential source of confusion due to the manner in which we write examples in this paper. In the examples, the coding function on an edge is not written as an abtract function f_e, but rather, we take an f_e, pick a basis, and multiply the matrix representing f_e by the message vector and write the resulting code. Given the simple greedy algorithm stated above, and such a picture representation, it is easy to think that the algorithm translates to "in each step check to see if one of the symbols written on an edge can be sent." However, this is not what we want to do. Our choice of the matrix used to represent f_e was arbitrary, so if X, Y is written on an edge, it is equally valid to write $X, X+Y$ or $Y, X+Y$, and we need the greedy algorithm to reflect that. Thus, our greedy algorithm needs to look at linear subspaces to determine if it is possible to send new information along any edge.

Given a network code $\Phi = (G, \mathfrak{M}, \{\Sigma_e\}, \{f_e\})$, with coding functions over the field $\mathbb{F}$, our greedy algorithm (pseudocode, **LinSerialize**(Φ), is given in Algorithm 1), constructs a set of edge functions $f_e^{(1..k)}$ and alphabets $\Sigma_e^{(1..k)}$ for each edge. These objects are constructed iteratively, defining the edge alphabets $\Sigma_e^{(i)}$ and coding functions $f_e^{(i)}$ in the i^{th} iteration. Throughout this process, we maintain a pair of linear subspaces A_e, $B_e \subseteq \mathfrak{M}^*$ for each edge $e = (u, v)$ of G. A_e is the linear span[3] of all the messages transmitted on e so far, and B_e is intersection of T_e with the linear span of all the messages transmitted to u so far. (In other words, B_e spans all the messages that could currently be sent on e without receiving any additional messages at u.) In the i^{th} iteration, if there exists an edge e' such that $B_{e'}$ contains a dual vector $x_{e'}$ that does not belong to $A_{e'}$, then we create coding function $f_e^{(i)}$ for all e. The coding function of $f_{e'}^{(i)}$ is set to be $x_{e'}$ and its alphabet is set to be $\mathbb{F}$. For all other edges we set $f_e^{(i)} = 0$, and update the A_e's accordingly. This process continues until $B_e = A_e$ for every e. At that point, we report that the code is serializable if and only if $A_e = T_e$ for all e. At the end, the algorithm returns the functions $f_e^{(1..k)}$ and the alphabets $\Sigma_e^{(1..k)}$, where k is the number of iterations of the algorithm, as well as the subspaces $\{A_e\}$. If the code

[3]If $\{V_i : i \in \mathcal{I}\}$ is a collection of linear subspaces of a vector space V, their linear span is the minimal linear subspace containing the union $\bigcup_{i\in\mathcal{I}} V_i$. We denote the linear span by $+_{i\in\mathcal{I}} V_i$.

was not serializable, then $\{A_e\}$ is interpreted as a certificate of non-serializability (a "non-trivial information vortex") as explained below.

Algorithm 1 Greedy Algorithm for Linear Codes

LinSerialize(Φ)

/* $\Phi = (G, \mathfrak{M}, \{\Sigma_e\}, \{f_e\}), G = (V, E, S)$ *is a network code with coding functions over field* $\mathbb{F}$. *We construct* $\Sigma_e^{(1..k)}$ *and* $f_e^{(1..k)}$. */
$A_e \leftarrow 0$ for all $e \in E$. /* $A_e \subseteq T_e$ *represents the information we have sent over edge* e */
$A_s \leftarrow T_s$ for all $s \in S$.
$B_e \leftarrow T_e \cap \left(+_{s \in \mathsf{In}(e)} A_s\right)$ for all $e \in E$. /* $B_e \subseteq T_e$ *represents the information that the tail of* e *knows about* T_e */
$i = 1$
while $\exists\, e = (u, v)$ in G such that $A_e \neq B_e$ **do**
  Let x_e be any vector in B_e that doesn't lie in A_e
  $\Sigma_e^{(i)} \leftarrow \mathbb{F}$, $f_e^{(i)} \leftarrow x_e$
  $A_e \leftarrow A_e + \langle x_e \rangle$
  $\forall\, e' \in E, e' \neq e, \Sigma_{e'}^{(i)} \leftarrow 0$, $f_{e'}^{(i)} \leftarrow 0$
  $\forall\, e' = (v, \cdot) \in E$, $B_{e'} \leftarrow T_{e'} \cap (B_{e'} + \{x_e\})$ /* *Node* v *"learns"* x_e */
  $i{+}{+}$
end while

LinSerialize(Φ) runs in time polynomial in the size of the coding functions of Φ. In every iteration of the while loop we increase the dimension of some A_e by one. A_e is initialized with dimension zero and can have dimension at most $\dim(T_e)$. Therefore, the algorithm goes through at most $\sum_{e \in E} \dim(T_e)$ iterations of the while loop. Additionally, each iteration of the while loop, aside from constant time assignments, computes only intersections and spans of vector spaces, all of which can be done in polynomial time.

To prove the algorithm's correctness, we define the following certificate of non-serializability.

Definition 4.1 An *information vortex (IV)* of a network code consists of a linear subspace $W_e \subseteq \mathfrak{M}^*$ for each edge e, such that:

1. For a source edge s, $W_s = T_s$.
2. For every other edge e, $W_e = T_e \cap \left(+_{e' \in \mathsf{In}(e)} W_{e'}\right)$.

An information vortex is *nontrivial* if $W_e \neq T_e$ for some edge e.

We think of W_e as the information that we can send over e given that its incoming edges, $e' \in In(e)$, can send $W_{e'}$. In our analysis of the greedy algorithm, we show that the messages the greedy algorithm succeeds in sending (i.e., the linear subspaces $\{A_e\}$) form an IV and it is non-trivial if and only if the code isn't serializable.

In Section 5 and Section 6 we will see that information vortices provide a concise way for proving the non-serializability of a network code. Moreover, the notion of an information vortex was critical to our discovery of the result in Section 6.

The following theorem shows the relationship between IVs, serialization, and the greedy algorithm.

Theorem 4.2 For a network code $\Phi = (G, \mathfrak{M}, \{\Sigma_e\}, \{f_e\})$, the following are equivalent:

1. Φ is not serializable
2. **LinSerialize**(Φ) returns $\{A_e\}$ s.t. $\exists\, e, A_e \neq T_e$

3. Φ has a non-trivial information vortex

Proof. $\underline{\neg\mathbf{2} \Rightarrow \neg\mathbf{1}}$ If **LinSerialize**(Φ) returns $\{A_e\}$ s.t. $\forall e$, $A_e = T_e$ then Φ is serializable:
We show that the $f_e^{(1..k)}, \Sigma_e^{(1..k)}$ created by **LinSerialize**(Φ) satisfy the conditions in Definition 2.2:

1. $f_e^{(i)} : \mathfrak{M} \to \Sigma_e^{(i)}$ by construction.
2. The non-zero functions $f_e^{(i)}$ form a basis for T_e. Because linear maps are indifferent to the choice of basis, if $f_e(m_1) = f_e(m_2)$ then in any basis, each coordinate of $f_e(m_1)$ equals the corresponding coordinate of $f_e(m_2)$, and thus $f_e^{(i)}(m_1) = f_e^{(i)}(m_2)$ for all i.
3. If $f_e(m_1) \neq f_e(m_2)$ then for any basis we choose to represent f_e, the values $f_e(m_1), f_e(m_2)$ will differ in at least one coordinate, and thus $\exists i,\ f_e^{(i)}(m_2) \neq f_e^{(i)}(m_2)$.
4. When we assign a function $f_e^{(i)} = x_e$ we have that x_e is in B_e which guarantees it is computable from information already sent to the tail of e.

$\underline{\mathbf{2} \Rightarrow \mathbf{3}}$ If **LinSerialize**(Φ) returns $\{A_e\}$ s.t. $\exists\, e\, A_e \neq T_e$ then Φ has a non-trivial IV.
We claim the the vector spaces $\{A_e\}$ returned by **LinSerialize**(Φ) form a non-trivial IV. $\{A_e\}$ is non-trivial by hypothesis, so it remains to show it is an *IV*. $\{A_e\}$ satisfies property (1): For each $S \in S$, $A_S = T_S$ by construction (Line 3 of **LinSerialize**(Φ)).
$\{A_e\}$ satisfies property (2): By induction on our algorithm, B_e is exactly $T_e \cap \left(+_{e' \in \mathsf{In}(e)} A_{e'}\right)$. At termination, $B_e = A_e$ for all $e \in E$. So, we have that $A_e = T_e \cap \left(+_{e' \in \mathsf{In}(e)} A_{e'}\right)$.

$\underline{\mathbf{3} \Rightarrow \mathbf{1}}$ If Φ has a non-trivial IV then it isn't serializable.
Suppose for contradiction that $\Phi = (G, \mathfrak{M}, \{\Sigma_e\}, \{f_e\}), G = (V, E, S)$ is serializable. Let $f_e^{(1..k)}$ and $\Sigma_e^{(1..k)}$ satisfy the conditions of definition 2.2. Let $\{W_e\}$ be a non-trivial IV for Φ.

We say that a function $f_e^{(j)}$ has property P if there $\exists\, m_1, m_2 \in \mathfrak{M}$ such that $f_e^{(j)}(m_1) \neq f_e^{(j)}(m_2)$ and $m_1, m_2 \in W_e^{\perp}$. There must be such a function since our IV is non-trivial and $\Sigma_e^{(1..k)}, f_e^{(1..k)}$ is a serialization of Φ. Let i^* be the smallest i such that any function satisfies property P and suppose $f_{e^*}^{(i^*)}$ satisfies P with messages m_1^*, m_2^*.

By definition, $W_{e^*} = T_{e^*} \cap \left(+_{e' \in \mathsf{In}(e^*)} W_{e'}\right)$, so $m_1^*, m_2^* \in W_{e^*}^{\perp}$ implies that for all $e' \in \mathsf{In}(e^*)$, $m_1^*, m_2^* \in W_{e'}^{\perp}$. But, $f_{e^*}^{(i^*)}$ can distinguish between m_1^*, m_2^* so at least one of $e' \in \mathsf{In}(e^*)$ must also be able to distinguish between m_1^*, m_2^* at a time *before* i^*. Therefore, there exists some $f_{e'}^{(i')}$, $i' < i^*$ that satisfies property P, a contradiction to the fact that i^* was the smallest such i. □

4.2 General codes

Our characterization theorem extends to the case of general network codes, provided that we generalize the greedy algorithm and the definition of information vortex appropriately. The message space $\mathfrak{M}$ is no longer a vector space, so instead of defining information vortices using the vector space $\mathfrak{M}^*$ of all linear functions on $\mathfrak{M}$, we use the Boolean algebra $2^{\mathfrak{M}}$ of all binary-valued functions on $\mathfrak{M}$. We begin by recalling some notions from the theory of Boolean algebras.

Definition 4.3 Let S be a set. The Boolean algebra 2^S is the algebra consisting of all $\{0,1\}$-valued functions on S, under AND ($\wedge$), OR ($\vee$), and NOT ($\neg$). If $f : S \to T$ is a function, then the *Boolean algebra generated by* f, denoted by $\langle f \rangle$, is the subalgebra of 2^S consisting of all functions $b \circ f$, where b is a $\{0,1\}$-valued function on T. If A_1, A_2 are subalgebras of a Boolean algebra A, their intersection $A_1 \cap A_2$ is a subalgebra as well. Their union is not, but it generates a subalgebra that we will denote by $A_1 + A_2$.

If S is a finite set and $A \subseteq 2^S$ is a Boolean subalgebra, then there is an equivalence relation on S defined by setting $x \sim y$ if and only if $b(x) = b(y)$ for all $b \in A$. The equivalence classes of this relation are called the *atoms* of A, and we denote the set of atoms by $\mathrm{At}(A)$. There is a canonical function $f_A : S \to \mathrm{At}(A)$ that maps each element to its equivalence class. Note that $A = \langle f_A \rangle$.

The relevance of Boolean subalgebras to network coding is as follows. A subalgebra $A \subseteq 2^{\mathfrak{M}}$ is a set of binary-valued functions, and can be interpreted as describing the complete state of knowledge of a party that knows the value of each of these functions but no others. In particular, if a sender knows the value of $f(m)$ for some function $f : \mathfrak{M} \to T$, then the binary-valued messages this sender can transmit given its current state of knowledge correspond precisely to the elements of $\langle f \rangle$. This observation supplies the raw materials for our definition of the greedy algorithm for general network codes, which we denote by **GenSerialize**(Φ).

As before, the edge alphabets and coding functions are constructed iteratively, with $\Sigma_e^{(i)}$ and $f_e^{(i)}$ defined in the i^{th} iteration of the main loop. Throughout this process, we maintain a pair of Boolean subalgebras A_e, $B_e \subseteq 2^{\mathfrak{M}}$ for each edge $e = (u, v)$ of G. A_e is generated by all the messages transmitted on e so far, and B_e is intersection of $\langle f_e \rangle$ with the subalgebra generated by all messages transmitted to u so far. (In other words, B_e spans all the binary-valued messages that could currently be sent on e without receiving any additional messages at u.) In the i^{th} iteration, if there exists an edge e' such that $B_{e'}$ contains a binary function $x_{e'} \notin A_{e'}$, then we create a binary-valued coding function $f_e^{(i)}$ for all e, which is set to be $x_{e'}$ if $e = e'$ and the constant function 0 if $e \neq e'$. This process continues until $B_e = A_e$ for every e. At that point, we report that the code is serializable if and only if $A_e = \langle f_e \rangle$ for all e. At the end, the algorithm returns the functions $f_e^{(1..k)}$ and the alphabets $\Sigma_e^{(1..k)}$, where k is the number of iterations of the algorithm, as well as the subspaces $\{A_e\}$. The pseudocode for this algorithm **GenSerialize**(Φ) is presented in Appendix B.

If Φ has finite alphabets, then **GenSerialize**(Φ) must terminate because the total number of atoms in all the Boolean algebras A_e ($e \in E$) is strictly increasing in each iteration of the main loop, so $\sum_{e \in E} |\Sigma_e|$ is an upper bound on the total number of loop iterations. In implementing the algorithm, each of the Boolean algebras can be represented as a partition of $\mathfrak{M}$ into atoms, and all of the operations the algorithm performs on Boolean algebras can be implemented in polynomial time in this representation. Thus, the running time of **GenSerialize**(Φ) is polynomial in $\sum_{e \in E} |\Sigma_e|$. In light of the algorithm's termination condition, the following definition is natural.

Definition 4.4 If $G = (V, E, S)$ is a sourced graph, a *generalized information vortex (GIV)* in a network code $\Phi = (G, \mathfrak{M}, \{\Sigma_e\}, \{f_e\})$ is an assignment of Boolean subalgebras $A_e \subseteq 2^{\mathfrak{M}}$ to every $e \in E \cup S$, satisfying:

1. $A_s = \langle f_s \rangle$ for all $s \in S$;
2. $A_e = \left(+_{\hat{e} \in \mathsf{In}(u)} A_{\hat{e}}\right) \cap \langle f_e \rangle$ for all $e = (u, v) \in E$.

A GIV is *nontrivial* if $A_e \neq \langle f_e \rangle$ for some $e \in E$. A tuple of Boolean subalgebras $\Gamma = (A_e)_{e \in E \cup S}$ is a *semi-vortex* if it satisfies (1) but only satisfies one-sided containment in (2), i.e.,

3. $A_e \subseteq \left(+_{\hat{e} \in \mathsf{In}(u)} A_{\hat{e}}\right) \cap \langle f_e \rangle$ for all $e = (u, v) \in E$.

If $\Gamma = (A_e)$ and $\Upsilon = (A'_e)$ are semi-vortices, we say that Γ is contained in Υ if $A_e \subseteq A'_e$ for all e.

In Appendix B we prove a series of statements (Lemmas B.2-B.5) showing that:

- Semi-vortices are in one-to-one correspondence with restrictions of Φ. The correspondence maps a semi-vortex $(A_e)_{e \in E \cup S}$ to the network code with edge alphabets $\mathrm{At}(A_e)$ and coding functions given by the canonical maps $\mathfrak{M} \to \mathrm{At}(A_e)$ defined in Definition 4.3.
- There is a set of semi-vortices corresponding to serializable restrictions of Φ under this correspondence. They can be thought of as representing *partial serializations* of Φ.

- There is a set of semi-vortices corresponding to GIV's of Φ. These can be thought of as *certificates of infeasibility* for serializing Φ.
- **GenSerialize**(Φ) computes a semi-vortex Γ which is both a GIV and a partial serialization.

These lemmas combine to yield a "min-max theorem" showing that the every network code has a maximal serializable restriction that coincides with its minimal GIV, as well as an analogue of Theorem 4.2; proofs of both theorems are in Appendix B.

Theorem 4.5 In the ordering of semi-vortices by containment, the ones corresponding to partial serializations have a maximal element and the GIV's have a minimal element. These maximal and minimal elements coincide, and they are both equal to the semi-vortex $\Gamma = (A_e)_{e \in E \cup S}$ computed by **GenSerialize**(Φ).

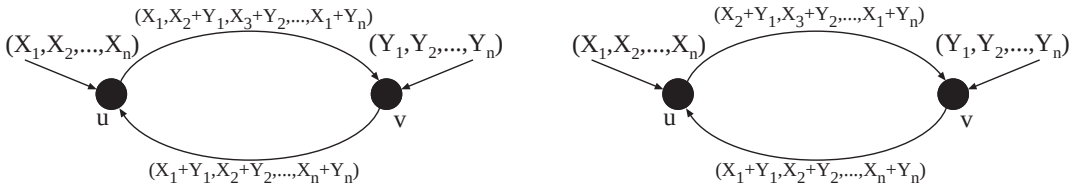
Theorem 4.6 For a network code Φ with finite alphabets, the following are equivalent.

1. Φ is serializable.
2. **GenSerialize**(Φ) outputs $\{A_e\}_{e \in E}$ s.t. $\forall\, e, A_e = \langle f_e \rangle$.
3. Φ has no nontrivial GIV.

5 The Serializability Deficit of Linear Network Codes

The min-max relationship between serializable restrictions and information vortices (Theorem 4.5) is reminiscent of classical results like the max-flow min-cut theorem. However, there is an important difference: one can use the minimum cut in a network to detect *how far* a network flow problem is from feasibility, i.e. the minimum amount by which edge capacities would need to increase in order to make the problem feasible. In this section, we will see that determining how far a network code is from serializability is more subtle: two network codes can be quite similar-looking, with similar-looking minimal information vortices, yet one of them can be serialized by sending only one extra bit while the other requires many more bits to be sent.

We begin with an example to illustrate this point. The codes in Figure 2 apply to the message tuple $(X_1, \ldots, X_n, Y_1, \ldots, Y_n)$ where X_i, Y_i are independent, uniformly distributed random variables over $\mathbb{F}_2$. The codes in Figures 2(a) and 2(b) are almost identical; the only difference is that the code in Figure 2(a) has one extra bit along the top edge. The code in Figure 2(a) is serializable: transmit X_1 along (u, v), then $X_1 + Y_1$ on edge (v, u), then $X_2 + Y_1$ on (u, v), ... ,$X_n + Y_n$ on (v, u), and finally $X_1 + Y_n$ on (u, v). On the other hand, the code in Figure 2(b) is not serializable, which can be seen by applying our greedy algorithm.



(a) Serializable (b) Not Serializable

Figure 2: Two almost identical network codes.

Thus, the code in Figure 2(b) is very close to serializable because we can consider an extension of the code in which we add one bit[4] to the edge (u, v) to obtain the code in Figure 2(a) that is

[4]In this section, for simplicity, we refer to one scalar-valued linear function on an $\mathbb{F}$-vector space as a "bit" even if $|\mathbb{F}| > 2$.

serializable. On the other hand, there are similar codes that are very far from being serializable. If we consider the code with the same sources and $f_{(u,v)} = f_{(v,u)} = \prod_{i=1}^{n} X_i + Y_i$, its edge alphabets have the same size and its minimal information vortex is identical, yet any serializable extension requires adding n bits. To completely characterize serializability we would like to be able to separate codes that are close to serializable from those that are far. This motivates the following definition.

Definition 5.1 For a network code $\Phi = (G, \mathfrak{M}, \{\Sigma_e\}, \{f_e\})$ and an extension $\Phi' = (G', \mathfrak{M}, \{\Sigma'_e\}, \{f'_e\})$, the *gap* of Φ', defined by $\gamma(\Phi') = \sum_{e \in E} \log_2 |\Sigma'_e| - \log_2 |\Sigma_e|$, represents the combined number of extra bits transmitted on all edges in Φ' as compared to Φ. The *serializability deficit* of Φ, denoted by $\mathsf{SD}(\Phi)$, is defined to be the minimum of $\gamma(\Phi')$ over all serializable extensions Φ' of Φ. The *linear serializability deficit* of a linear code Φ, denoted $\mathsf{LSD}(\Phi)$, is the minimum of $\gamma(\Phi')$ over all linear serializable extensions Φ'.

Unfortunately, determining the serialization deficit is much more difficult than simply determining serializability.

Theorem 5.2 Given a linear network code Φ, it is NP-hard to approximate the size of the minimal linear serializable extension of Φ. Moreover, there is a linear network code Φ and a positive integer n such that $\mathsf{LSD}(\Phi^n)/(n\mathsf{LSD}(\Phi)) < \mathcal{O}(\frac{1}{\log_2(n)})$.

Both statements in the theorem follow directly from the following lemma.

Lemma 5.3 Given a hitting set instance (N, S) with universe N, $|N| = n$, subsets $S \subseteq 2^N$, an optimal integral solution k, and an optimal fractional solution $\frac{z_1}{q}, \frac{z_2}{q}, .., \frac{z_n}{q}$, with $\sum_{i=1}^{n} \frac{z_i}{q} = \frac{p}{q}$, in polynomial time we can construct a linear network code such that $\mathsf{LSD}(\Phi) = k$, but $\mathsf{LSD}(\Phi^q) \leq p$.

Proof sketch.

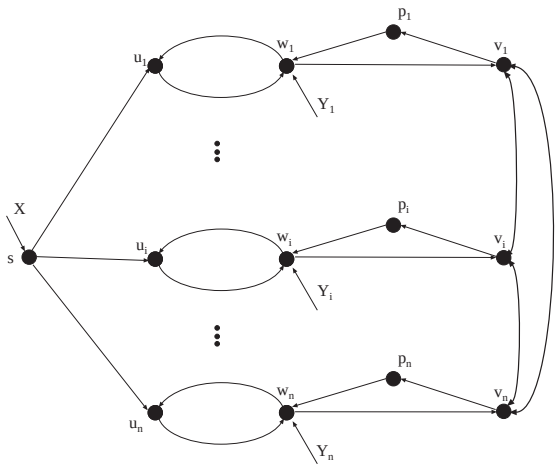


Figure 3: The reduction from Hitting Set

The full proof of the Lemma is in Appendix C. Here, due to space limitations, we merely sketch the main ideas. The graph used in the reduction is illustrated pictorially in Figure 3. Given a hitting set instance (N, S) we create a network code with one source for each $i \in N$ (source message denoted by $\vec{Y}_i$) and a super-source s (source message denoted $(\vec{X}_1, \vec{X}_2, \ldots, \vec{X}_n)$). The symbols $\vec{X}_i, \vec{Y}_i$ don't refer to bits, but actually to blocks of n_i bits, where n_i is the number of sets in S containing i; each of the bits in $\vec{X}_i$ or $\vec{Y}_i$ corresponds to one of the sets that i belongs to. For each $i \in N$ we use a gadget consisting of a 2-cycle on vertices u_i, w_i, with $\vec{Y}_i$ feeding into w_i and $\vec{X}_i$ feeding from the super-source s into u_i. The edges between u_i and w_i are a copy of the gadget in Figure 2. We exploit the fact that sending one extra bit in this gadget allows the information vortex in the gadget to "unravel", leading to transmission of all the bits encoded on the edges of the 2-cycle. The 2-cycle (u_i, w_i) participates in a larger 4-vertex gadget $\{u_i, w_i, p_i, v_i\}$ corresponding to the element i. The role of v_i is to participate in "set gadgets", where the gadget corresponding to a set A consists of a bidirected clique on all the vertices $\{v_j | j \in A\}$. The role of p_i is less important; it plays a necessary part in disseminating bits to leftover parts of the network after the "important" parts have been serialized. If there is a hitting set of size k then we send one bit on each of the 2-cycles (u_i, w_i) corresponding to elements i in the hitting set. This "unlocks" the bits that were locked up in those 2-cycles, which allows a sufficient amount of

information to flow into the set gadgets that they become serialized. The vertices p_j are then used for disseminating the remaining bits to the unused 2-cycles (u_j, w_j) where j did not belong to the hitting set.

To prove, conversely, that a serializability deficit of at most k implies that there is a hitting set of size k, we make use of the fact that the network code constructed by our reduction has a large number of information vortices, one for each pair consisting of an element of N and a set S that it belongs to. If C is the set of all i such that an extra bit is transmitted somewhere in the 4-vertex gadget for i, and C fails to contain an element of some set A, then this in turn implies that one of the aforementioned information vortices remains an information vortex in the extension of the code. Thus, C must be a hitting set.

The more difficult step in proving Lemma 5.3 lies in showing that fractional solutions of the hitting set problem can be transformed into efficient serializable extensions of Φ^q. For this, we make use of the fact that the edge alphabets in Φ^q can be regarded as $\Sigma_e \otimes \mathbb{F}^q$. and their duals can be regarded as $\Sigma_e^* \otimes (\mathbb{F}^*)^q$. If $|\mathbb{F}|$ is large enough, then the uniform matroid $U_{q,p}$ is representable as a set $\{t_1, \ldots, t_p\}$ of p vectors in $(\mathbb{F}^*)^q$. For each of the "fractional elements" z_i/q in the fractional hitting set, we send z_i bits of the form $x \otimes t$ in the extension of Φ^q, where t is one of the elements of our matroid representation of $U_{q,p}$ in $(\mathbb{F}^*)^q$ and x is the bit that we would have sent for element i in the hitting set reduction described two paragraphs earlier. The fact that $\sum_i z_i = p$ implies that every element of the matroid representation is used exactly once in this construction. The fact that we have a fractional set cover implies that in each set gadget, we receive extra bits corresponding to q distinct elements of S. Since these elements are a basis for $(\mathbb{F}^*)^q$, it is then possible to show that they combine to allow a serialization of all the "missing bits" in that gadget, and from there we finish serializing the entire network code Φ^q as before.

6 Asymptotic Serializability

The results in the previous section showed both that there are non-serializable codes with large edge alphabets that become serializable by adding only one bit (example in Figure 2) and that the serialization deficit can behave sub-additively when we take the n-fold cartesian product of Φ (Theorem 5.2). This prompts the investigation of whether there exists a code that isn't serializable, but the n-fold parallel repetition of the code can be serialized by extending it by only a constant number of bits, and thus it is essentially indistinguishable from serializable. We formalize this idea with the following definition.

Definition 6.1 A network code Φ is *asymptotically serializable* if $\lim_{n\to\infty} \frac{1}{n}\mathsf{LSD}(\Phi^n)/\mathsf{LSD}(\Phi) = 0$ where Φ^n is n-fold cartesian product of Φ with cartesian product define in the obvious way.

If one is using a network code to transmit infinite streams of data by chopping each stream up into a sequence of finite blocks and applying the specified coding functions to each block, then an asymptotically serializable network code is almost as good as a serializable one, since it can be serialized by adding a side channel of arbitrarily small bit-rate to each edge of the network.

Despite indications to the contrary in Section 5, we show that any non-serializable linear code is not asymptotically serializable via the following theorem.

Theorem 6.2 For a linear network code $\Phi = (G, \mathfrak{M}, \{\Sigma_e\}, \{f_e\})$ over a field $\mathbb{F}$, then $\mathsf{LSD}(\Phi^n) \geq cn$ where c is a constant dependent on Φ.

The proof of the theorem considers the alphabets of the n-fold product of Φ as elements of a tensor product space. Using this machinery, we show that information vortices in the graph are preserved if we don't increase the amount of information we send down some edge by order n

bits. More specifically, if $\{W_e\}$ is a non-trivial information vortex in Φ, and e is an edge such that $\dim(W_e) < \dim(T_e) = m$, then if we add some edge function f to every edge in the graph, the information vortex remains non-trivial as long as the dimension of f is less than mn. A complete proof is provided in Appendix D.

7 A Characterization Theorem for the 2-Cycle

In order to use entropy inequalities to give tight upper bounds on network coding rates, we need an enumeration of the complete set of entropy inequalities implied by serializability, i.e. a list of necessary and sufficient conditions for a vector V to be the entropic vector of a serializable code. (Note that it need not be the case that every code whose entropic vector is V must be serializable.) For the 2-cycle we can enumerate the complete set of inequalities. In particular, we give four inequalities that must hold for any serializable code on the 2-cycle: two are a result of *downstreamness* which is a condition that must hold for all graphs (it says that the entropy of the incoming edges of a vertex must be at least as much as the entropy of the incoming and outgoing edges together), the third is the *Chicken and Egg* inequality due to [5], and the fourth is a new inequality that we call the *greedy* inequality. It is equivalent to being able to complete the first iteration of our greedy algorithm in Section 4. We show that these four inequalities together with Shannon's inequalities are the only inequalities implied by serializability, in the following sense:

Theorem 7.1 Given a rational-valued entropic vector, V, of a 2-cycle on nodes u, v, with source x into node u, source y into node v, and edges $a = (u, v)$ and $b = (v, u)$, there exists a serializable code that realizes cV, for some constant c, if and only if V satisfies Shannon's inequalities, downstreamness ($H(abx) = H(bx)$, $H(aby) = H(ay)$), the Chicken and Egg inequality ($H(ab) \geq H(abx) - H(x) + H(aby) - H(y)$), and the greedy inequality ($H(a) + H(b) > H(ax) - H(x) + H(by) - H(y)$ when $H(a) + H(b) \neq 0$).

Multiplication of the vector by a constant c is a natural relaxation because the theorem becomes oblivious to the base of the logarithm we use to compute the Shannon entropy.

The proof of Theorem 7.1 involves considering four cases corresponding to the relationship between $H(a)$ and $H(ax), H(ay), H(x), H(y)$ and between $H(b)$ and $H(bx), H(by)H(x), H(y)$. Each case requires a distinctly different coding function to realize the entropic vector. All the coding functions are relatively simple, involving only sending uncoded bits, and the XOR of two bits. Most of the work is limiting the values of coordinates of the entropic vector based on the inequalities that the entropic vector satisfies. The proof of Theorem 7.1 is provided in Appendix A.

The big open question left from this work is whether we can find a complete set of constraints on the entropic vector implied by the serializability of codes on arbitrary graphs. We currently do not know of any procedure for producing such a list of inequalities. Even if we had a conjecture for such a list, showing that it is complete is likely to be quite hard. If we have more than three sources, just determining the possible dependencies between sources is difficult because they are subject to non-Shannon information inequalities.

References

[1] Micah Adler, Nicholas J. A. Harvey, Kamal Jain, Robert Kleinberg, and April Rasala Lehman. On the capacity of information networks. In *Proc. of the 17th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 241–250, 2005.

[2] Rudolf Ahlswede, Ning Cai, Shuo-Yen Robert Li, and Raymond W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, 000.

[3] Elona Erez and Meir Feder. Efficient network codes for cyclic networks. In *Proc. 2005 International Symposium on Information Theory (ISIT)*, pages 1982–1986, 2005.

[4] Nicholas J. A. Harvey, Robert Kleinberg, and April Rasala Lehman. On the capacity of information networks. *IEEE Transactions on Information Theory*, 52(6):2345–2364, 2006.

[5] Nicholas J.A. Harvey, Robert Kleinberg, Chandra Nair, and Yunnan Wu. A "chicken & egg" network coding problem. In *Proc. 2007 IEEE International Symposium on Information Theory (ISIT)*, pages 131–135, 2007.

[6] Kamal Jain, Vijay Vazirani, Raymond W. Yeung, and Gideon Yuval. On the capacity of multiple unicast sessions in undirected graphs. In *Proc. 2005 IEEE International Symposium on Information Theory (ISIT)*, 2005.

[7] Ralf Koetter and Muriel Medard. An algebraic approach to network coding. *IEEE/ACM Transactions on Networking*, 11(5):782–795, 2003.

[8] Gerhard Kramer and Serap Savari. Edge-cut bounds on network coding rates. *Journal of Network and Systems Management*, 14(1):49–67, 2006.

[9] April Rasala Lehman. *Network Coding*. PhD thesis, MIT, 2005.

[10] Raymond W. Yeung. *A First Course in Information Theory*. Springer, 2002.

[11] Raymond W. Yeung, Shuo-Yen Robert Li, Ning Cai, and Zhen Zhang. *Network Coding Theory*. Now Publishers, 2006.

[12] Raymond W. Yeung and Zhen Zhang. Distributed source coding for satellite communication. *IEEE Transactions on Information Theory*, 45(4):1111–1120, 1999.

A Proof of Theorem 7.1

Theorem A.1 (Theorem 7.1 restated) Given a rational-valued entropic vector, V, of a 2-cycle on nodes u, v, with source x into node u, source y into node v, and edges $a = (u, v)$ and $b = (v, u)$, there exists a serializable code that realizes cV, for some constant c, if and only if V satisfies Shannon's inequalities, downstreamness ($H(abx) = H(bx)$, $H(aby) = H(ay)$), the Chicken and Egg inequality ($H(ab) \geq H(abx) - H(x) + H(aby) - H(y)$), and the Greedy inequality ($H(a) + H(b) > H(ax) - H(x) + H(by) - H(y)$ when $H(a) + H(b) \neq 0$).

Throughout this proof it will often be convenient to refer to the conditional entropy of two sets of edges.

Definition A.2 For two subsets of edges $F = \{e_1, e_2, \ldots, e_j\}$ and $F' = \{e'_1, e'_2, \ldots, e'_k\}$, the conditional entropy of F given F', denoted $H(F|F') = H(e_1 e_2 \ldots e_j | e'_1 e'_2 \ldots e'_k) = H(FF') - H(F')$.

We first show that all the inequalities are necessary. Shannon's inequalities must hold for the entropic vector of any set of random variables. Downstreamness (term coined by [9]) was shown to be necessary even for DAGS by Yeung and Zhang [12]. Harvey et al. [5] showed that the Chicken and Egg inequality is necessary. Thus, it remains to show that our greedy inequality is a necessary condition for serializability.

Lemma A.3 The inequality $H(a) + H(b) > H(ax) - H(x) + H(by) - H(y) = H(a|x) + H(b|y)$ holds for any serializable code on a 2-cycle when $H(a) + H(b) > 0$.

Proof. Suppose there is a serializable code such that $H(a) + H(b) \leq H(a|x) + H(b|y)$ and $H(a) + H(b) > 0$. Because conditioning reduces entropy, $H(a) \geq H(a|x)$, and likewise $H(b) \geq H(b|y)$. These three inequalities together imply that $H(a) = H(a|x)$ and $H(b) = H(b|y)$. It follows from the definition of serializability and $H(a) + H(b) > 0$ that there exists a non-zero $f_a^{(i)}$ or $f_b^{(i)}$. Let i^* be the smallest such i and let $f_a^{(i^*)}$ be the associated non-zero coding function (the choice of a is without loss of generality). We can rewrite $H(a|x)$ as $H(f_a^{(i^*)}|x) + H(a|f_a^{(i^*)}x)$. $H(f_a^{(i^*)}|x) = 0$ because i^* is the smallest such i implies that $f_a^{(i^*)}$ is computed soley from x. But, this gives us that $H(a) = H(a|f_a^{(i^*)}x)$, which is a contradiction to $f_a^{(i)}$ non-zero. □

To prove the other direction of Theorem 7.1 we will use a case based analysis, but first we make a few observations to bound the cases we need to consider.

Observation A.4 The following ten values completely determine the entropic vector of the 2-cycle: $I(x;y), H(x|y), H(y|x), H(a|x), H(b|x), H(a|y), H(b|y), H(a), H(b), H(ab)$.

Proof. Due to downstreamness and Shannon's inequalities the following equations hold: $H(xy) = I(x;y)+H(x|y)+H(y|x), H(axy) = H(bxy) = H(abxy) = H(xy), H(y) = I(x;y)+H(y|x), H(x) = I(x;y) + H(x|y), H(ax) = H(a|x) + H(x), H(aby) = H(ay) = H(a|y) + H(y), H(abx) = H(bx) = H(b|x) + H(x), H(by) = H(b|y) + H(y)$. This implies that the value of all 15 non-zero elements of the entropic vector are determined by the 10. □

Observation A.5 $H(b|x) \geq H(a|x)$

Proof. $H(bx) = H(abx) \geq H(ax)$ by downstreamness and then monotonicity. □

Observation A.6 $H(a|y) \geq H(b|y)$

Proof. Parallel to proof of observation A.5 □

Observation A.7 $\max(H(a|x), H(a|y)) \leq H(a) \leq H(a|x) + H(a|y) + I(x;y)$

Proof. $H(a) \leq H(a|x) + H(a|y) + I(x;y)$: Apply submodularity on ax and ay to get $H(ax) + H(ay) \geq H(axy) + H(a) = H(xy) + H(a)$, then subtract $H(x) + H(y)$ from both sides.
$H(a) \geq \max(H(a|x), H(a|y)) : H(a) \geq H(a|x)$ and $H(a) \geq H(a|y)$ because conditioning reduces entropy. □

Observation A.8 $\max(H(b|x), H(b|y)) \leq H(b) \leq H(b|x) + H(b|y) + I(x;y)$

Proof. Parallel to proof of observation A.7 □

Observation A.9 $H(b|x) + H(a|y) \leq H(ab) \leq H(b|x) + H(a|y) + I(x;y)$

Proof. $H(ab) \geq H(b|x) + H(a|y)$ by the chicken and egg inequality.
$H(ab) \leq H(b|x) + H(a|y) + I(x;y)$: by submodularity on ay and bx:
$H(ay) + H(bx) = H(aby) + H(abx) \geq H(abxy) + H(ab)$.
$\Rightarrow H(ab) \leq H(bx) + H(ay) - H(xy)$. □

Now, we come to our case analysis for proving the forward direction of Theorem 7.1.

We first multiply our entropic vector by the least common denominator so that all the elements of the vector are integer. We show that we can find a code that realizes this integer valued entropic vector.

Let $X_1, ...X_{H(x|y)}$,$Z_1, ...Z_{I(x;y)}$ be random variables originating at source x, and let $Y_1, ..., Y_{H(y|x)}$, $Z_1, ..., Z_{I(x;y)}$ be random variables originating at source y, where X_i, Y_j, Z_k are independent for all i, j, k.

We split up the proof into 4 cases. Case 1 corresponds to when $H(a)$ is greater than $H(a|x) + H(a|y)$ and $H(b)$ is greater than $H(b|x)+H(b|y)$. Case 4 takes care of the instances when both $H(a)$ is less than $H(a|x) + H(a|y)$ and $H(b)$ is less than $H(b|x) + H(b|y)$. Cases 2 and 3 are symmetric corresponding to when exactly one of $H(a)$ and $H(b)$ is greater than the sum of the conditional entropy on x and y. Cases 1,2 (or 3),4 correspond to distinctly different coding functions on edges a and b. Case 1 has the simplest codes - we send bits uncoded with the exception of possibly XORing X and Z or Y and Z. In cases 2 and 3 we need to XOR bits of X, Y on one edge, and in case 4 we need to XOR bits of X, Y on both edges in a manner similar to the example in Figure 2(b).

Case 1:
$H(a) = H(a|x) + H(a|y) + f$, $f \geq 0$ and note $f \leq I(x;y)$ by Observation A.7.
$H(b) = H(b|x) + H(b|y) + g$, $g \geq 0$ and note $g \leq I(x;y)$ by Observation A.8.
$H(ab) = H(b|x) + H(a|y) + h$, and note $0 \leq h \leq I(x;y)$ by Observation A.9.

Observation A.10 $h \leq H(a|x) + H(b|y) + f + g$

Proof. Implied by submodularity on a and b. □

Observation A.11 $h \geq \max(f, g)$

Proof.

$$\begin{aligned} H(x|a) &\geq H(x|ab) \text{ Conditioning reduces entropy} \\ H(ax) - H(a) &\geq H(abx) - H(ab) \\ H(ab) - H(bx) - H(a|y) + H(x) &\geq H(a) - H(ax) - H(a|y) + H(x) \\ H(ab) - H(b|x) - H(a|y) &\geq H(a) - H(a|x) - H(a|y) \\ h &\geq f \end{aligned}$$

The proof that $h \geq g$ is similar.

□

We claim that the following code realizes the entropic vector and is serializable:
For notational convenience let $Z'_1 = Z_{f+1}, Z'_2 = Z_{f+2}, ..., Z'_{h-f-g} = Z_{h-g}$. Any Z'_i with $i > h-f-g$ we will take to be 0.

$$f_a = X_1, \ldots, X_{H(a|y)}, Y_1 + Z'_1, \ldots, Y_{H(a|x)} + Z'_{H(a|x)}, Z_1, \ldots Z_f$$

$$f_b = X_1 + Z'_{H(a|x)+1}, \ldots, X_{H(b|y)} + Z'_{H(a|x)+H(b|y)}, Y_1, \ldots, Y_{H(b|x)}, Z_{h-g-1}, \ldots, Z_h$$

This is a valid code because $H(x|y) \geq H(a|y) \geq H(b|y)$, $H(y|x) \geq H(b|x) \geq H(a|x)$, $h \leq I(x;y)$, $h \leq H(a|x) + H(b|y) + f + g$, and $h \geq \max(f, g)$. It is easy to check that this code realizes the entropic vector. It is serializable because $H(a|y) \geq H(b|y)$, $H(b|x) \geq H(a|x)$ and both sources know Z.

Case 2:
$H(a) = H(a|x) + H(a|y) - f$, $f \geq 0$ and note $f \leq \min(H(a|x), H(a|y))$ by Observation A.7.
$H(b) = H(b|x) + H(b|y) + g$, $g \geq 0$ and note $g \leq I(x;y)$ by Observation A.8.

$H(ab) = H(b|x) + H(a|y) + h$, and note $0 \leq h \leq I(x;y)$ by Observation A.9.

Observation A.12 $h \leq (H(a|x) - f) + H(b|y) + g$

We claim that the following code realizes the entropic vector and is serializable:
Any Z_i with $i > h$ we will take to be 0.

$$f_a = X_1 + Y_1, X_2 + Y_2, \ldots, X_f + Y_f, X_{f+1}, \ldots, X_{H(a|y)}, Y_{f+1} + Z_{g+1}, \ldots, Y_{H(a|x)} + Z_{g+H(a|x)-f}$$
$$f_b = X_1 + Z_{g+H(a|x)-f+1}, \ldots, X_{H(b|y)} + Z_{g+H(a|x)-f+H(b|y)}, Y_1, \ldots, Y_{H(b|x)}, Z_1, \ldots, Z_g$$

This is a valid code for the same reasons as Case 1, and also because $h \leq (H(a|x) - f) + H(b|y) + g$, and $f \leq H(a|x)$ and $f \leq H(a|y)$. It is easy to check that this code realizes the entropic vector; here it is important that $g \leq h$ which is true by the argument from Observation A.11. It is serializable because we can send $Y_1, ..., Y_{H(b|x)}$ along edge b, then because $H(b|x) \geq H(b|y)$ we can send everything along edge a, and then because $H(a|y) \geq H(a|x)$ we can send all the Xs and Zs on edge b.

Case 3:
$H(a) = H(a|x) + H(a|y) + f$, $f \geq 0$ and note $f \leq I(x;y)$ by Observation A.7.
$H(b) = H(b|x) + H(b|y) - g$, $g \geq 0$ and note $g \leq \min(H(b|x), H(b|y))$ by Observation A.8.
$H(ab) = H(b|x) + H(a|y) + h$, and note $0 \leq h \leq I(x;y)$ by Observation A.9.

Symmetric to Case 2.

Case 4:
$H(a) = H(a|x) + H(a|y) - f$, $f \geq 0$ and note $f \leq \min(H(a|x), H(a|y))$ by Observation A.7.
$H(b) = H(b|x) + H(b|y) - g$, $g \geq 0$ and note $g \leq \min(H(b|x), H(b|y))$ by Observation A.8.
$H(ab) = H(b|x) + H(a|y) + h$, and note $0 \leq h \leq I(x;y)$ by Observation A.9.

Applying the inequality $H(a) + H(b) > H(a|x) + H(b|y)$, together with the fact that $H(a) \geq H(a|x)$ and $H(b) \geq H(b|x)$ implies that at least one of $H(a) > H(a|x)$, $H(b) > H(b|y)$ holds. Or, written in terms of f, g this means that at least one of $f < H(a|y)$, $g < H(b|x)$ holds.

Observation A.13 $h \leq (H(a|x) - f) + (H(b|y) - g)$

Case 4a: $f < H(a|y)$

We claim that the following code realizes the entropic vector and is serializable:
Any Z_i with $i > h$ we will take to be 0.

$$f_a = X_2 + Y_1, X_3 + Y_2, \ldots, X_{f+1} + Y_f, X_1, X_{f+2}, \ldots, X_{H(a|y)}, Y_{f+1} + Z_1, \ldots, Y_{H(a|x)} + Z_{H(a|x)-f}$$
$$f_b = X_1 + Y_1, X_2 + Y_2, \ldots, X_g + Y_g, X_{g+1} + Z_{H(a|x)-f+1}, \ldots, X_{H(b|y)} + Z_{H(a|x)-f+H(b|y)-g}, Y_{g+1}, \ldots, Y_{H(b|x)}$$

This is a valid code because $f+1 \leq H(a|y)$, $h \leq (H(a|x)-f)+(H(b|y)-g)$, $f \leq \min(H(a|x), H(a|y))$ and $g \leq \min(H(b|x), H(b|y))$. It is easy to check that this code realizes the entropic vector. To show it is serializable, we first consider the case when $f \leq g$: we can send X_1 along edge a; then $X_1 + Y_1$ along edge b; then $X_2 + Y_1$ along edge a; ...; then $X_{f+1} + Y_f, X_{f+2}, \ldots, X_{H(a|y)}$ along edge a; then because $H(a|y) \geq H(b|y)$, we can send then everything along edge b; and then since $H(b|x) \geq H(a|x)$ we can complete the transmission for edge a. The case for $f > g$ is very similar.

Case 4b: $g < H(b|x)$
This case is similar, but we switch the roles of edge a and edge b.

B Proofs omitted from Section 4.2

The following lemma is standard; for completeness, we provide a proof here.

Lemma B.1 Suppose $f_1 : S \to T_1$ and $f_2 : S \to T_2$ are two functions on a set S.

1. $\langle f_2 \rangle \subseteq \langle f_1 \rangle$ if and only if there exists a function $g : T_1 \to T_2$ such that $f_2 = g \circ f_1$.
2. Suppose S is finite. If $f_1 \times f_2$ denotes the function $S \to T_1 \times T_2$ defined by $x \mapsto (f_1(x), f_2(x))$, then $\langle f_1 \rangle + \langle f_2 \rangle = \langle f_1 \times f_2 \rangle$.

Proof. A Boolean subalgebra of 2^S can be equivalently described as a collection of subsets of S, closed under union, intersection, and complementation, by equating a $\{0,1\}$-valued function b with the set $b^{-1}(1)$. In this proof we adopt the "collection of subsets" definition of a Boolean subalgebra of 2^S, since it is more convenient. Note that under this interpretation, if $f : S \to T$ is any function then $\langle f \rangle$ consists of all subsets of the form $f^{-1}(U)$, $U \subseteq T$.

If $f_2 = g \circ f_1$ for some g, then every set of the form $f_2^{-1}(U)$ can be expressed as $f_1^{-1}(g^{-1}(U))$ which shows that $\langle f_2 \rangle \subseteq \langle f_1 \rangle$. Conversely, if $\langle f_2 \rangle \subseteq \langle f_1 \rangle$ then for every $u \in T_2$ the set $f_2^{-1}(\{u\}) \in \langle f_2 \rangle$ belongs to $\langle f_1 \rangle$, i.e. it can be expressed as $f_1^{-1}(V_u)$ for some set $V_u \subseteq T_1$. The sets $f_1^{-1}(V_u)$ are disjoint as u ranges over the elements of T_2 so the sets V_u themselves must be disjoint. Define $g(v) = u$ if $v \in V_u$ for some $u \in T_2$, and define $g(v)$ to be an arbitrary element of T_2 otherwise. For any $x \in S$, if $u = f_2(x)$ then $x \in f_2^{-1}(u) = f_1^{-1}(V_u)$, which implies that $g(f_1(x)) = u$. Hence $f_2 = g \circ f_1$ as desired.

To prove (2) we argue as follows. Clearly $\langle f_1 \rangle, \langle f_2 \rangle \subseteq \langle f_1 \times f_2 \rangle$, so $\langle f_1 \rangle + \langle f_2 \rangle \subseteq \langle f_1 \times f_2 \rangle$ as well. For the reverse inclusion, note that every element of $\langle f_1 \times f_2 \rangle$ can be expressed as a finite union of sets of the form $(f_1 \times f_2)^{-1}(t_1, t_2)$. Every such set can be expressed as $f_1^{-1}(t_1) \cap f_2^{-1}(t_2)$, which proves that it belongs to $\langle f_1 \rangle + \langle f_2 \rangle$. □

Algorithm 2 Greedy algorithm for general network codes

GenSerialize(Φ)

/* $\Phi = (G, \mathfrak{M}, \{\Sigma_e\}, \{f_e\}), G = (V, E, S)$ *is a network code.* */
/* *We construct* $\Sigma_e^{(1..k)}$ *and* $f_e^{(1..k)}$. */
$A_e \leftarrow 0$ for all $e \in E$. /* $A_e \subseteq \langle f_e \rangle$ *represents the information we have sent over edge* e */
$A_s \leftarrow \langle f_s \rangle$ for all $s \in S$.
$B_e \leftarrow \langle f_e \rangle \cap \left(+_{s \in \mathsf{In}(e)} A_s\right)$ for all $e \in E$.
/* $B_e \subseteq \langle f_e \rangle$ *represents the information that the tail of* e *knows about* f_e */
$i = 1$
while $\exists\, e = (u, v)$ in G such that $A_e \neq B_e$ **do**
  Let x_e be any binary-valued function in $B_e \setminus A_e$.
  $\Sigma_e^{(i)} \leftarrow \{0, 1\}$, $f_e^{(i)} \leftarrow x_e$
  $A_e \leftarrow A_e + \langle x_e \rangle$
  $\forall\, e' \in E, e' \neq e, \Sigma_{e'}^{(i)} \leftarrow \{0\}$, $f_{e'}^{(i)} \leftarrow 0$
  $\forall\, e' = (v, \cdot) \in E$, $B_{e'} \leftarrow \langle f_{e'} \rangle \cap (B_{e'} + \langle x_e \rangle)$ /* *Node* v *"learns"* x_e */
  $i \leftarrow i + 1$
end while

Lemma B.2 For a given network code Φ, restrictions Φ' of Φ are in one-to-one correspondence with semi-vortices Γ. The correspondence maps Γ to the network code $\Phi'[\Gamma]$ whose alphabets are $\Sigma'_e = \mathrm{At}(A_e)$ and whose coding functions are the functions $f'_e = f_{A_e}$ defined in Definition 4.3.

The inverse correspondence maps Φ' to the unique semi-vortex $\Gamma[\Phi']$ satisfying $A_r = \langle f'_r \rangle$ for all $r \in E \cup S$.

Proof. Suppose $\Gamma = (A_r)$ is a semi-vortex and $\Phi'[\Gamma]$ is defined as stated, with coding functions $f'_e = f_{A_e}$. For all $e = (u, v) \in E$, let $\mathsf{In}(e) = \{e_1, \ldots, e_k\}$ and let $f_i = f'_{e_i}$. The relation

$$A_e = \left(+_{i=1}^{k} A_{e_i}\right) \cap \langle f_e \rangle$$

implies that

$$A_e \subseteq +_{i=1}^{k} A_{e_i} = +_{i=1}^{k} \langle f_i \rangle = \langle (f_1, \ldots, f_k) \rangle,$$

where the last equation follows from Lemma B.1. Since $\langle f_{A_e} \rangle = A_e \subseteq \langle (f_1, \ldots, f_k) \rangle$, we can apply Lemma B.1 again to conclude that $f_{A_e} = g \circ (f_1, \ldots, f_k)$ for some g. Thus $\Phi'[\Gamma]$ is a network code. To prove that it is a restriction of Φ, we use the containment $A_e \subseteq \langle f_e \rangle$ for every edge $e \in E \cup S$, together with Lemma B.1, to construct the functions $g_e : \Sigma_e \to \Sigma'_e$ required by the definition of a restriction of Φ. □

Lemma B.3 If Φ' is a restriction of Φ, and Φ' is serializable, then $\Gamma[\Phi']$ is contained in every GIV of Φ.

Proof. Suppose Φ' is a serializable restriction of Φ, with serialization consisting of alphabets $\Sigma_e^{(i)}$ and coding functions $f_e^{(i)}$.

Suppose now that $\Gamma = \{A_e\}_{e \in E \cup S}$ is any GIV of Φ. First, we claim $\langle f_e^{(i)} \rangle \subseteq A_e$ for every edge e. To prove the claim we use induction on i. The claim is clearly true when $i = 0$. Otherwise, let $e_1, \ldots, e_r$ be the edges in $\mathsf{In}(e)$. By Lemma B.1, the existence of a function $h_e^{(i)}$ such that $f_e^{(i)}(m) = h_e^{(i)}\left(\prod_{j=1}^{r} f_{e_j}^{1..i-1}\right)$ implies the first of the following containments:

$$\langle f_e^{(i)} \rangle \subseteq +_{j=1}^{r} +_{\ell=1}^{i-1} \langle f_{e_j}^{(\ell)} \rangle \subseteq +_{j=1}^{r} A_{e_j}. \tag{1}$$

The second containment in (1) follows from our induction hypothesis. Now, property 2 of a serialization implies that $\langle f_e^{(i)} \rangle \subseteq \langle f'_e \rangle$. Combining this with (1) we obtain

$$\langle f_e^{(i)} \rangle \subseteq \left(+_{j=1}^{r} A_{e_j}\right) \cap \langle f'_e \rangle = A_e, \tag{2}$$

as desired.

If $\Gamma[\Phi']$ is not contained in Γ, then there exists an edge e of G such that

$$\langle f'_e \rangle \not\subseteq A_e. \tag{3}$$

Property 2 of a serialization implies the existence of a function $H : \Sigma'_e \to \prod_{i=1}^{k} \Sigma_e^{(i)}$ such that $H(f'_e(m)) = (f_e^{(1)}(m), \ldots, f_e^{(k)}(m))$ for all $m \in \mathfrak{M}$. Property 3 implies that H is one-to-one, hence it has a left inverse: a function $G : \prod_{i=1}^{k} \Sigma_e^{(i)} \to \Sigma'_e$ such that $G \circ H$ is the identity. Letting $F = \prod_{i=1}^{k} f_e^{(i)}$, the definition of H implies that $F = H \circ f'_e$, whence $f'_e = G \circ F$. Applying Lemma B.1 once more,

$$\langle f'_e \rangle \subseteq \langle F \rangle = +_{i=1}^{k} \langle f_e^{(i)} \rangle,$$

and the right side is contained in A_e by (2). This contradicts (3), which completes the argument. □

Lemma B.4 At the start of any iteration of the main loop of **GenSerialize**(Φ) the following invariants hold.

1. $A_e = \langle f_e^{(1)}, \ldots, f_e^{(i-1)} \rangle$ for all $e \in E$.
2. $B_e = \langle f_e \rangle \cap \left(+_{\hat{e} \in \mathsf{In}(u)} A_{\hat{e}} \right)$ for all $e \in E$.
3. The collection of subalgebras $\Gamma = \{A_e\}_{e \in E \cup S}$ constitutes a semi-vortex.
4. $\Phi'[\Gamma]$ is a serializable restriction of Φ.

Proof. The first three invariants can be verified by a trivial induction on the number of loop iterations. We claim that $\Phi'[\Gamma]$ is serializable, and in fact that the coding functions $\{f_e^{(j)}\}$ constructed in the preceding iterations of the main loop constitute a serialization of $\Phi'[\Gamma]$. For property 1 of a serialization, there is nothing to check. To prove property 2, observe that $f_e^{(j)} \in A_e = \langle f'_e \rangle$, which implies by Lemma B.1 that $f_e^{(j)} = b \circ f'_e$ for some binary-valued function b on Σ'_e. If $f'_e(m_1) = f'_e(m_2)$ then $b(f'_e(m_1)) = b(f'_e(m_2))$, which establishes property 2. To prove property 3, observe that $A_e = \langle f'_e \rangle$ is generated by the functions $f_e^{(1..i-1)}$, so if $f'_e(m_1) \neq f'_e(m_2)$ then there is some $j \leq i-1$ such that $f_e^{(j)}(m_1) \neq f_e^{(j)}(m_2)$. Finally, property 4 follows from the structure of the algorithm itself. Either $f_e^{(j)}$ is the constant function 0, in which case there is nothing to prove, or $f_e^{(j)}$ is equal to the function x_e chosen in line 2 of the j^{th} loop iteration of **GenSerialize**(Φ). In that case x_e belonged to the Boolean algebra B_e at the start of that loop iteration, which means

$$x_e \in \langle f_e \rangle \cap \left(+_{\hat{e} \in \mathsf{In}(u)} A_{\hat{e}} \right) \subseteq +_{\hat{e} \in \mathsf{In}(u)} A_{\hat{e}} = +_{\hat{e} \in \mathsf{In}(u)} \left(+_{1 \leq \ell < j} \langle f_{\hat{e}}^{(\ell)} \rangle \right)$$

and another application of Lemma B.1 implies the existence of the function $h_e^{(j)}$ required by the definition of serialization. □

Lemma B.5 When **GenSerialize**(Φ) terminates, $\Gamma = \{A_e\}_{e \in E \cup S}$ is a GIV.

Proof. Lemma B.4 ensures that Γ is a semi-vortex, and the algorithm's termination condition ensures that there is no edge e such that $A_e \neq B_e$. In light of the fact that $B_e = \langle f_e \rangle \cap \left(+_{\hat{e} \in \mathsf{In}(u)} A_{\hat{e}} \right)$, this implies that $\Gamma = \{A_e\}_{e \in E \cup S}$ is a GIV. □

Theorem B.6 (Restatement of Theorem 4.5) In the ordering of semi-vortices by containment, the ones corresponding to partial serializations have a maximal element and the GIV's have a minimal element. These maximal and minimal elements coincide, and they are both equal to the semi-vortex $\Gamma = \{A_e\}_{e \in E \cup S}$ computed by **GenSerialize**(Φ).

Proof. By Lemmas B.4 and B.5, Γ is a GIV and $\Phi' = \Phi'[\Gamma]$ is a serializable restriction of Φ. If Φ'' is any other serializable restriction of Φ, then Lemma B.3 implies that $\Gamma[\Phi''] \subseteq \Gamma$ because Γ is a GIV. If Υ is any GIV, then Lemma B.3 implies that $\Upsilon \supseteq \Gamma[\Phi'] = \Gamma$ becase Φ' is a serializable restriction of Φ. □

Theorem B.7 (Restatement of Theorem 4.6) For a network code Φ with finite alphabets, the following are equivalent.

1. Φ is serializable.
2. **GenSerialize**(Φ) outputs $\{A_e\}_{e \in E}$ s.t. $\forall\, e, A_e = \langle f_e \rangle$.
3. Φ has no nontrivial GIV.

Proof. In the proof of Theorem 4.5, we saw that the subalgebras $\{A_e\}$ at the time **GenSerialize**(Φ) terminates constitute a GIV Γ such that:

- Γ is contained in every other GIV;

- $\Phi'[\Gamma]$ is a serializable restriction of Φ;
- Γ contains $\Gamma[\Phi'']$ for every serializable restriction Φ'' of Φ.

Let $\Phi' = \Phi'[\Gamma]$. We now distinguish two cases.

Case 1: Φ' is isomorphic to Φ. In this case, we show that all three equivalent conditions hold. First, Φ is serializable because Φ' is. Second, the fact that Φ' is isomorphic to Φ means that $\langle f_e \rangle = \langle f'_e \rangle = A_e$ for every edge e. Finally, we know that every GIV in Φ contains Γ. But $\Gamma = \Gamma[\Phi'] = \Gamma[\Phi]$, which is the trivial GIV. By Definition 4.4, any GIV containing the trivial GIV is trivial. Hence Φ contains no nontrivial GIV.

Case 2: Φ' is not isomorphic to Φ. In this case, Φ' is a proper restriction of Φ, hence the semi-vortex $\Gamma = \Gamma[\Phi']$ constitutes a nontrivial GIV. Any serializable restriction Φ'' of Φ satisfies $\Gamma[\Phi''] \subseteq \Gamma \subsetneq \Gamma[\Phi]$. In particular this means that Φ is not a serializable restriction of itself, i.e. Φ is not serializable. Finally, the statement that Φ' is not isomorphic to Φ means that for some $e \in E$, $\langle f'_e \rangle \neq \langle f_e \rangle$. Recalling that $\langle f'_e \rangle = A_e$, this means that **GenSerialize**(Φ) does not output $\{A_e\}_{e \in E}$ such that $\forall\, e, A_e = \langle f_e \rangle$. □

C Analysis of the hitting set reduction

Reduction C.1 (Hitting Set to Minimum $\mathsf{LSD}(\Phi)$) Given a hitting set instance (N, S) with universe $N = \{1, ..., n\}$ and a family $S \subseteq 2^N$ of subsets of N, we let $S(i) = \{A_{i(1)}, A_{i(2)}, ..., A_{i(n_i)}\} \subseteq S$ where $A \in S(i)$ iff $i \in A$, and the ordering of As is arbitrary. We make the network coding instance $\Phi = (G, \mathfrak{M}, \{\Sigma_e\}, \{f_e\})$ where G is a directed, sourced graph with vertex set: $\{s\} \cup V \cup U \cup W \cup P$, where $V = \{v_1, v_2, ..., v_n\}$, and similarly for U, W and P; and source edges: $\{(\bullet, s)\} \cup \{(\bullet, w_i) | i \in N\}$ with messages $f_{(\bullet,s)} = \prod_{i \in N} \prod_{A \in S(i)} X_i^A$ and $f_{(\bullet,w_i)} = \prod_{A \in S(i)} Y_i^A$. Where all X, Y are uniform random variables over $\mathbb{F}_{2^\ell}$ for some sufficiently large $\ell > 0$ to be chosen later. Rather than enumerate our edge set E and the coding functions on each edge, we will just specify the coding functions for each edge in E. If a function f_e is not specified, then e is not in E. We also show the network coding instance pictorially in Figure 3. We use $\prod$ denote the n-fold cartesien product, so $\prod_{i=1}^n X_i$ is synonomous with the ordered n-tuple $(X_1, X_2, ...X_n)$. The coding functions are as follows:

$$f_{(s,u_i)} = \prod_{k=1}^{n_i} X_i^{A_{i(k)}},\ \forall i \in N \qquad f_{(s,v_i)} = \prod_{j \in N : j \neq i} \prod_{k=1}^{n_j} X_j^{A_{j(k)}}\ \forall i \in N$$

$$f_{(s,p_i)} = \sum_{k=2}^{n_i} X_i^{A_{i(k)}},\ \forall i \in N \qquad f_{(v_i,v_j)} = \prod_{A \in S(i) \cap S(j)} \sum_{k \in A} X_k^A,\ \forall i, j \in N$$

$$f_{(w_i,v_i)} = \prod_{k=1}^{n_i} X_i^{A_{i(k)}},\ \forall i \in N \qquad f_{(v_i,p_i)} = \sum_{k=1}^{n_i} X_i^{A_{i(k)}},\ \forall i \in N$$

$$f_{(p_i,w_i)} = X_i^{A_{i(1)}},\ \forall i \in N \qquad f_{(w_i,u_i)} = \prod_{k=1}^{n_i} X_i^{A_{i(k)}} + Y_i^{A_{i(k)}},\ \forall i \in N$$

$$f_{(u_i,w_i)} = \prod_{k=1}^{n_i} X_i^{A_{i(k+1 \mod n_i)}} + Y_i^{A_{i(k)}},\ \forall i \in N$$

Proof of Part 1 of Lemma 5.3. Given a hitting set instance (N, S) we create the network code Φ using the Reduction C.1.

We show that (N, S) has a hitting set of size k if and only if $\mathsf{LSD}(\Phi) \leq k$.

($\Rightarrow$) Suppose (N, S) has a hitting set of size k. We show that $\mathsf{LSD}(\Phi) \leq k$.

Let C be a hitting set of size k. Consider adding bit $X_c^{A_{c(1)}}$ for $c \in C$ to edge (u_c, w_c). This allows us to serialize all bits in the following stages, implicitly we are defining $f_e^{(1..k)}$ and $\Sigma_e^{(1..k)}$ for all $e \in E$:

1. For all $c \in C$, we can serialize all bits on edges (w_c, u_c) and (u_c, w_c): w_c learns $X_c^{A_{c(1)}}$, so it can send bit $X_c^{A_{c(1)}} + Y_c^{A_{c(1)}}$ to u_c. Now, this allows $X_c^{A_{c(2)}} + Y_c^{A_{c(1)}}$ to be sent on (u_c, w_c), and we continue in this way until all bits serialized on these two edges.
2. For all $c \in C$, send $f_{(w_c,v_c)} = \prod_{k=1}^{n_c} X_c^{A_{c(k)}}$ on (w_c, v_c).
3. For all $i \in N$, send $f_{(s,v_i)} \prod_{j \in N: j \neq i} \prod_{k=1}^{n_j} X_j^{A_{j(k)}}$ on (s, v_i)
4. For every set $A \in S$ there is an element $c \in A \cap C$ because C is a hitting set. Thus, there is a vertex in V, v_c that knows X_c^A. v_c can therefore send bit $t(A) = \sum_{a \in A} X_a^A$ to all $v_a, a \in A, a \neq c$. Now every v_a knows $t(A)$ and can send it along $(v_a, v_{a'})$ for all $a' \in A, a' \neq a$. This serializes all bits on edges between vertices in V.
5. Now every vertex v_i knows every bit X: it received all but $\prod_{k=1}^{n_i} X_i^{A_{i(k)}}$ in step 3, and determined $\prod_{k=1}^{n_i} X_i^{A_{i(k)}}$ in step 4. So, we can send $f_{(v_i,p_i)} = \sum_{k=1}^{n_i} X_i^{A_{i(k)}}$ on edge (v_i, p_i).
6. At p_i we can add the code $\sum_{k=2}^{n_i} X_i^{A_{i(k)}}$ from (s, p_i) to $f_{(v_i,p_i)}$ to obtain $X_i^{A_{i(1)}}$ and send it on (p_i, w_i).
7. Now, for all $i \in N - C$, we can serialize (w_i, u_i) and (u_i, w_i) as we did in step 1.

($\Leftarrow$) Suppose that $\mathsf{LSD}(\Phi) \leq k$. We show that (N, S) has a hitting set of size k.

Consider the partition of E into sets $E(i) \forall i \in N$ such that $E(i) = \{(\cdot, \cdot_i) | i \in N\}$, that is $e \in E(i)$ if and only if the tail of e is indexed by i.

Lemma C.2 For a set $A \in S$ suppose no bits are added to any edge in $\bigcup_{i \in A} E(i)$, then the bit $t(A) = \sum_{a \in A} X_a^A$ on $(v_a, v_{a'})$, $\forall a, a' \in A$ cannot be serialized.

Lemma C.2 implies that for every set $A \in S$, at least one bit must be added on an edge in $\bigcup_{i \in A} E(i)$ to serialize Φ. In particular, if Φ' is a minimal serializable extension of Φ and let $C = \{i | \Phi' \text{ sends at least one additional than } \Phi \text{ on some edge in } E(i)\}$ then C is a hitting set for (N, S). And $|C| \leq \mathsf{LSD}(\Phi) \leq k$.

Proof of Lemma C.2. Let $V(A) = \bigcup_{a \in A} v_a$. Any edge e going into any node in $V(A)$ must send f_e because no bits are added on any of these edges. In any serialization, it must be that for some $v_a \in V(A)$, bit $t(A)$ is sent on $(v_a, v_{a'})$ for some a' before v_a receives $t(A)$ from any $v_{a''} \in V(A)$. Without loss of generality, suppose this vertex is $v_i, i \in A$. Consider the subgraph induced by vertices indexed by i. There is an information vortex on this subgraph with $W_{(s,v_i)} = T_{(s,v_i)}, W_{(s,u_i)} = T_{(s,u_i)}$, and $W_e = 0$ for all other edges in the subgraph. To check this one simply has to verify that for edges e out of v_i $T_e \cap T_{(s,v_i)} = 0$, and similarly for u_i. This implies that this subgraph is not serializable. We don't add any bits to the subgraph by hypothesis, so to "destroy" this IV, and serialize the subgraph we need $W_{(v_j,v_i)} + W_{(s,v_i)}$ to have a non-zero intersection with $T_{(v_i,\cdot)}$. But this is a contradiction to our choice of i. $\square$

□

Proof of Part 2 of Lemma 5.3. Given a hitting set instance (N, S) we create the network code Φ using the Reduction C.1. If $|\mathbb{F}|$ is large enough then one can show using facts from linear algebra (or matroid theory) that the subset $T = \left\{ \begin{pmatrix} 1 \\ x_1^1 \\ \vdots \\ x_1^{q-1} \end{pmatrix}, \begin{pmatrix} 1 \\ x_2^1 \\ \vdots \\ x_2^{q-1} \end{pmatrix}, \ldots, \begin{pmatrix} 1 \\ x_p^1 \\ \vdots \\ x_p^{q-1} \end{pmatrix} \right\} \subset \mathbb{F}^q$ for $\{x_1, \ldots, x_p\}$ distinct elements in $\mathbb{F}$ has the property that any q element subset forms a basis for $\mathbb{F}^q$; in other words, T is a realization of the uniform matroid $U_{p,q}$ over $\mathbb{F}$. Partition the p vectors of T into $|N|$ subsets $T_1, \ldots, T_{|N|}$ such that T_i contains z_i vectors, note that $\sum_{i \in N} z_i = p$ makes this valid.

We now consider Φ^q. Here, we will regard the edge alphabet for edge e as a vector in $\Sigma_e \otimes \mathbb{F}^q$. The tensor product space allows us to consider q copies of Σ_e on each edge e without fixing a basis. We claim that the extenstion of Φ^q in which we transmit the extra bits $\prod_{t \in T_i} X_i^{A_{i(1)}} \otimes t$ on edge (u_i, w_i) for all $i \in N$ is serializable.

Observation C.3 Transmitting $\prod_{t \in T_i} X_i^{A_{i(1)}} \otimes t$ along edge (u_i, w_i) allows node w_i to learn $\prod_{t \in T_i} X_i^{A_{i(k)}} \otimes t$ for all $k \in \{1 \ldots n_i\}$.

Proof. We saw in the proof of the forward direction of the part 1 of Lemma 5.3 that transmitting $X_i^{A_{i(1)}}$ along edge (u_i, w_i) in Φ implies node w_i can learn $X_i^{A_{i(k)}}$ for all $k \in \{1 \ldots n_i\}$. This implies that in the q-fold repetition, transmitting $X_i^{A_{i(1)}} \otimes t$ along edge (u_i, w_i) allows node w_i to learn $X_i^{A_{i(k)}} \otimes t$ for all $k \in \{1 \ldots n_i\}$. □

Observation C.4 If $\prod_{t \in T_a} X_a^A \otimes t$ can be transmitted along edge (w_a, v_a) for all $a \in A$ then we can transmit $\sigma(A) = \sum_{a \in A} X_a^A \otimes \mathbb{F}^q$ on all edges $(v_a, v_{a'})$, $a, a' \in A$.

Proof. $\prod_{j \in N : j \neq a} \prod_{k=1}^{n_j} X_j^{A_{j(k)}} \otimes \mathbb{F}^q$ can be transmitted along (s_a, v_a) for all $a \in A$, and that, together with $\prod_{t \in T_a} X_a^A \otimes t$ transmitted along (w_a, v_a), allows node v_a, for all $a \in A$, to compute $\alpha(a) = \prod_{t \in T_a} \sum_{a \in A} X_a^A \otimes t$.

Now, fix $i \in A$. Send $\alpha(a)$ along (v_a, v_i) for all $a \in A, a \neq i$. Each message in the tuple $\alpha(a)$ is a linear combination of $\sum_{a \in A} X_a^A \otimes \mathbb{F}^q$ and is hence a legal message on all edges (v_a, v_i). After sending these messages, node v_i knows $\sum_{a \in A} X_a^A \otimes t$ for $\sum_{a \in A} z_a$ distinct vectors t. Our z_a's form a feasible fractional hitting set, thus $\sum_{a \in A} \frac{z_a}{q} \geq 1$, and $\sum_{a \in A} z_a \geq q$. Any q-element subset of T forms a basis of $\mathbb{F}^q$, and so v_i can determine $\sigma(A)$. We can then send $\sigma(A)$ on edges $(v_i, v_a), a \in A$ and then $(v_a, v_{a'})$ for all $a' \in A, a' \neq a$. □

Observation C.3 and C.4 together imply that for all sets $A \in S$ we can send $\sigma(A)$ on the clique formed by $v_a, a \in A$. A simple argument identitical to the last steps in the forward direction of the proof of part 1 of Lemma 5.3 imply that we can serialize the rest of the coding functions. □

D Proofs omitted from Section 6

The proofs in this section rely on knowledge of tensor products. We include a brief tutorial here for convenience.

D.1 Tensor products

If V, W are any two vector spaces with bases $\{\mathbf{e}_i^V\}_{i\in\mathcal{I}}$ and $\{\mathbf{e}_j^W\}_{j\in\mathcal{J}}$, respectively, their tensor product is a vector space $V \otimes W$ with a basis indexed by $\mathcal{I} \times \mathcal{J}$. The basis vector corresponding to an element (i, j) in the index set will be denoted by $\mathbf{e}_i^V \otimes \mathbf{e}_j^W$. For any two vectors $v = \sum_i a_i \mathbf{e}_i^V$ in V and $w = \sum_j b_j \mathbf{e}_j^W$ in W, their *tensor product* is the vector

$$v \otimes w = \sum_i \sum_j a_i b_j \mathbf{e}_i^V \otimes \mathbf{e}_j^W$$

in $V \otimes W$.

Lemma D.1 If $\{\mathbf{x}_i\}_{i\in\mathcal{I}}$ and $\{\mathbf{y}_j\}_{j\in\mathcal{J}}$ are bases of V, W, respectively, then $\{\mathbf{x}_i \otimes \mathbf{y}_j\}_{(i,j)\in\mathcal{I}\times\mathcal{J}}$ is a basis of $V \otimes W$.

Proof. It suffices to prove the lemma when $\mathbf{y}_j = \mathbf{e}_j^W$ for all $j \in \mathcal{J}$. If the lemma holds in this case, then by symmetry it also holds when $\mathbf{x}_i = \mathbf{e}_i^V$ for all $i \in \mathcal{I}$, and then the general case of the lemma follows by applying these two special cases in succession: first changing the basis of V, then changing the basis of W.

To prove that $B = \{\mathbf{x}_i \otimes \mathbf{e}_j^W\}_{(i,j)\in\mathcal{I}\times\mathcal{J}}$ is a basis of $V \otimes W$, it suffices to prove that every vector of the form $\mathbf{e}_i^V \otimes \mathbf{e}_j^W$ can be written as a linear combination of elements of B. By the assumption that $\{\mathbf{x}_i\}_{i\in\mathcal{I}}$ is a basis of V, we know that $\mathbf{e}_i^V = \sum_{k\in K} a_k \mathbf{x}_k$ for some finite subset $K \subseteq \mathcal{I}$ and scalars $(a_k)_{k\in K}$. Now it follows that $\mathbf{e}_i^V \otimes \mathbf{e}_j^W = \sum_{k\in K} a_k \left(\mathbf{x}_k \otimes \mathbf{e}_j^W\right)$, as desired. □

D.2 Basis and Rank

Definition D.2 If V is a vector space with basis B, and $\mathcal{W} = \{W_i\}_{i\in\mathcal{I}}$ is a collection of linear subspaces, we say that $\mathcal{W}$ is *B-compatible* if $W_i \cap B$ is a basis of W_i, for all $i \in \mathcal{I}$. We say that $\mathcal{W}$ is *basis-compatible* if there exists a basis B for V such that $\mathcal{W}$ is B-compatible.

Lemma D.3 If V is a vector space and $\mathcal{W}$ is a basis-compatible collection of linear subspaces, then $\mathcal{W}$ can be enlarged to a basis-compatible collection of linear subspaces $\bar{\mathcal{W}}$ that forms a Boolean algebra under $+$ and $\cap$. In particular, any three subspaces $X, Y, Z \in \mathcal{W}$ satisfy:

$$(X \cap Y) + Z = (X + Z) \cap (Y + Z)$$
$$(X + Y) \cap Z = (X \cap Z) + (Y \cap Z).$$

Proof. Simply let $\bar{\mathcal{W}}$ be the set of all linear subspaces of $W \subseteq V$ such that $W \cap B$ is a basis of W. □

Lemma D.4 If V is a vector space and $\mathcal{W}$ is a basis-compatible collection of linear subspaces, then $\mathcal{W} \cup \{V\} \cup \{0\}$ is also a basis-compatible collection of linear subspaces.

Proof. The proof is a trivial consequence of the definition of basis-compatible. □

Lemma D.5 If V is a vector space and X, Y are any two linear subspaces, then $\mathcal{W} = \{X, Y\}$ is basis-compatible.

Proof. Let B_{XY} be any basis of $X \cap Y$, let B_X be any basis of X containing B_{XY}, and let B_Y be any basis of Y containing B_{XY}. All the vectors in $B_X \cup B_Y$ are linearly independent, because if v is any vector that can be expressed as a linear combination of elements of $B_Y \setminus B_X$ and as a linear combination of elements of B_X, then v must belong to both X and Y, hence $v \in X \cap Y$. But the

only element of $X \cap Y$ that can be expressed as a linear combination of elements of $B_Y \setminus B_X$ is the zero vector, because B_{XY} is disjoint from $B_Y \setminus B_X$, and these two sets together constitute a basis of Y. Hence $B_X \cup B_Y$ can be extended to a basis B of V, and then $\mathcal{W} = \{X, Y\}$ is B-compatible. □

If A, B are subspaces of vector spaces V, W, respectively, then $A \otimes B$ is defined to be the linear subspace of $V \otimes W$ consisting of all linear combinations of vectors in the set $\{a \otimes b : a \in A,\, b \in B\}$. If $\mathcal{V}$ is a collection of linear subspaces of V and $\mathcal{W}$ is a collection of linear subspaces of W, then $\mathcal{V} \otimes \mathcal{W}$ denotes the collection of all linear subspaces $A \otimes B \subseteq V \otimes W$ such that $A \in \mathcal{V}$ and $B \in \mathcal{W}$.

Lemma D.6 If $\mathcal{V}$ is a basis-compatible collection of linear subspaces of V and $\mathcal{W}$ is a basis-compatible collection of linear subspaces of W then $\mathcal{V} \otimes \mathcal{W}$ is a basis-compatible collection of linear subspaces of $V \otimes W$.

Proof. If B, B' are bases of V, W, respectively, such that $\mathcal{V}$ is B-compatible and $\mathcal{W}$ is B'-compatible, then $\mathcal{V} \otimes \mathcal{W}$ is $(B \times B')$-compatible. □

Corollary D.7 If X, Y are subspaces of a vector space V and Z is a subspace of another vector space W, then

$$[(X \otimes W) + (Y \otimes W)] \cap (V \otimes Z) = [(X \otimes W) + (V \otimes Z)] \cap [(Y \otimes W) + (V \otimes Z)]\,.$$

Proof. By Lemmas D.4 and D.5, we know that $\mathcal{V} = \{X, Y, V\}$ is basis-compatible in V and $\mathcal{W} = \{Z, W\}$ is basis-compatible in W. Hence $\mathcal{V} \otimes \mathcal{W}$ is basis-compatible in $V \otimes W$. The corollary now follows by applying Lemma D.3. □

Definition D.8 If V, W are any vector spaces, a *rank-one* element of $V \otimes W$ is an element that can be expressed in the form $v \otimes w$ for some $v \in V$, $w \in W$. The *rank* of an element $x \in V \otimes W$ is the minimum value of r such that x can be expressed as a linear combination of r rank-one elements of $V \otimes W$. (If $x = 0$ then its rank is defined to be 0.)

Lemma D.9 If V, W are finite-dimensional vector spaces and $x \in V \otimes W$ then the rank of x is bounded above by $\min\{\dim(V), \dim(W)\}$.

Proof. Let $n = \dim(V), m = \dim(W)$. We will assume without loss of generality that $n \le m$ and prove that the rank of x is at most n. Let $\{\mathbf{e}_i^V\}$ and $\{\mathbf{e}_j^W\}$ be bases of V, W, respectively. We may express x as a linear combination

$$x = \sum_{i=1}^{n} \sum_{j=1}^{m} a_{ij} \mathbf{e}_i^V \otimes \mathbf{e}_j^W.$$

For $1 \le i \le n$ let $y_i = \sum_{j=1}^{m} a_{ij} \mathbf{e}_j^W$. Then

$$x = \sum_{i=1}^{n} \mathbf{e}_i^V \otimes y_i,$$

and this expresses x as a sum of n rank-one elements, implying that the rank of x is at most n. □

Lemma D.10 If V, W are finite-dimensional vector spaces of dimension n, m, and $P \subseteq V \otimes W$ is a linear subspace of dimension p, then there exists a pn-dimensional linear subspace $Q \subseteq W$ such that $P \subseteq V \otimes Q$.

Proof. Let $\{x_1, \ldots, x_p\}$ be a basis of P. By Lemma D.9, we can write each x_k in the form

$$x_k = \sum_{i=1}^{n} v_{ki} \otimes w_{ki},$$

for some vectors $v_{ki} (1 \leq k \leq p,\ 1 \leq i \leq n)$ in V and $w_{ki} (1 \leq k \leq p,\ 1 \leq i \leq n)$ in W. The vectors $\{w_{ki}\}$ span a subspace of W of dimension at most pn. Taking Q to be any pn-dimensional subspace of W containing $\{w_{ki}\}$, we see that $P \subseteq V \otimes Q$ as desired. □

D.3 Non-serializability

Theorem D.11 (Theorem 6.2 restated) For a linear network code $\Phi = (G, \mathfrak{M}, \{\Sigma_e\}, \{f_e\})$ over a field $\mathbb{F}$, then $\mathsf{LSD}(\Phi^n) \geq cn$ where c is a constant dependent on Φ.

Proof. If a linear network code has an information vortex $\{W_e\}_{e \in E}$ then $\{W_e^n\}_{e \in E}$ constitutes an information vortex in the product of n copies of the network code. Using the fact that $V^n = V \otimes \mathbb{F}^n$ for every vector space V, we may rewrite the information vortex as $\{W_e \otimes \mathbb{F}^n\}$.

Now suppose that T is a linear subspace of $\mathfrak{M}^* \otimes \mathbb{F}^n$ of dimension p. (Think of T as a set of bits that are added to Φ to get an extention of ϕ.) Using Lemma D.10, there is a subspace $Q \subseteq \mathbb{F}^n$ of dimension pd (where $d = \dim(\mathfrak{M}^*)$) such that $T \subseteq \mathfrak{M}^* \otimes Q$. Define a new family of subspaces

$$X_e = (W_e \otimes \mathbb{F}^n) + (\mathfrak{M}^* \otimes Q).$$

These subspaces constitute an information vortex. To verify this, we must check that

$$(W_e \otimes \mathbb{F}^n) + (\mathfrak{M}^* \otimes Q) = [(T_e \otimes \mathbb{F}^n) + (\mathfrak{M}^* \otimes Q)] \cap \left[+_{e' \in \mathrm{In}(e)} \left((W_{e'} \otimes \mathbb{F}^n) + (\mathfrak{M}^* \otimes Q)\right)\right]. \quad (4)$$

Because $\{W_e \otimes \mathbb{F}^n\}$ is an information vortex, we have

$$W_e \otimes \mathbb{F}^n = (T_e \otimes \mathbb{F}^n) \cap \left(+_{e' \in \mathrm{In}(e)} W_{e'} \otimes \mathbb{F}^n\right).$$

Equation (4) now follows by applying Corollary D.7 with $X = T_e, Y = +_{e' \in \mathrm{In}(e)} W_{e'}, Z = Q$. Thus the collection of subspaces $\{X_e\}$ constitutes an information vortex in the setting where T added to every edge of the network code. If the network code is serializable in this setting, then $\{X_e\}$ must be a trivial information vortex, implying that $X_e = T_e \otimes \mathbb{F}^n$ for every edge e. Because $\{W_e\}$ is non-trivial, we know there is at least one edge e such that $\dim(W_e) < \dim(T_e)$. Let $m = \dim(T_e)$, $k = \dim(W_e)$. The dimension of $W_e \otimes \mathbb{F}^n$ is kn. The dimension of $\mathfrak{M}^* \otimes Q$ is pd^2. Hence the dimension of X_e is bounded above by $kn + pd^2$. On the other hand, if the information vortex $\{X_e\}$ is trivial, then $X_e = T_e \otimes \mathbb{F}^n$ implying that $\dim(X_e) = mn$. Thus

$$\mathsf{LSD}(\Phi) \geq p \geq \frac{m-k}{d^2} \cdot n.$$

□