On the connection between network coding, network flow, and matroid theory

Jeffrey Kam

10-06-2021

Contents

1 Introduction

1.1 Background and motivation

When information is transmitted through the internet, packets of information are transferred between a global distribution of servers before reaching one's devices. Typically, the packets of data are purely relayed from one server to another throughout this chain of transmission, without modification to the original information in the packet. This mode of information transmission is known as network routing, and a lot of progress has been made on developing efficient routing methods during the 1980s. [2]

However, as shown in the example in Section 2.1, the butterfly network demonstrates the theoretical limits of network routing. It has been shown that by combining (or coding) the incoming packets before transmitting, we are able to achieve a higher information transmission rate. It was not until Ahlswede et al. published their paper "Network Information flow" in 2000 [1] that established foundation for recent developments in network coding, the study of how we can algebraically combine messages to achieve higher information throughput rate.

In addition to theoretical results, there has been research on applying network coding to potential real-world situations. For instance, researchers have leveraged network coding in designing distributed storage [3], distributed information sharing system [8], and peer-to-peer streaming mobile architecture [7]. However, the field is still relatively new and it has not seen widespread adoption.

Matroids, on the other hand, arise from a purely theoretical setting. There were introduced by Hassler Whitney in the 1930s, and many, such as William Tutte and Paul Seymour, made significant contributions to it since then. Matroids can be considered as a generalization of linear algebra and graph theory and a lot of the results in graph theory can be transferred to matroid theory. For instance, results such as Kuratowski's theorem, Tutte's matching theorem, and Turán's theorem, all have analogs in the matroid settings. It is an active and fascinating research area with unexpected connections to other fields, such as algebraic geometry and combinatorial optimization [9].

Network coding allows us to extend the capacity of the network transmission without using more resources. It allows us to achieve higher information rate that would otherwise not be possible using simple network routing ideas. In order to study and optimize coding-based solution, we would first have to understand the underlying structures of the networks. One natural area of study we can hope to extend results from is network flow. For instance, theorem such as max-flow min-cut has found analogs in the network coding setting [19]. Furthermore, there are a lot of structural similarities between matroids and network coding, as demonstrated by Dougherty et al. [5]. Thus, it would be highly beneficial if we are able to use tools from both network flow and matroid theory to better understand the problems in network coding.

1.2 Organization

In this project, we aim to explore the connections between network coding and matroid theory, as well as investigating some connections between network coding and network flow. As such, we will organize the project as follows and state the main references for each section.

Section 1 Introduction.

Section 2 An introduction to network coding with some illustrative examples, as well as some results on network capacities.

- Section 2.1 and 2.3: [5]
- Section 2.2: [5], [12]

Section 3 Connections between network solvability and network coding. In particular, we explore the solvability of multicast network and the complexity of linear coding.

- Section 3.1: [19]
- Section 3.2: [11]

Section 4 Some definitions and results of matroids with ample examples.

- Section 4.1, 4.2.1, and 4.2.2: [9]
- Section 4.2.3: [18]
- Section 4.2.4: [10]

Section 5 Matroidal network and its algorithmic construction.

- Section 5.1: [5], [6]
- Section 5.2 and 5.3: [5]

Section 6 Some results on the capacity of networks using the matroidal structures.

- Section 6.1, 6.2, and 6.3: [5]

Section 7 Conclusion.

2 Network coding fundamentals

2.1 Definitions

Definition 2.1 (Network Construction)
Let A be a finite alphabet set. Fix $k, m \in \mathbb{Z}$, a *message* and a *packet* are defined to be vectors in A of length k and m respectively. A *network* is a directed acyclic multigraph with an assigned message set. Then, each message is assigned to some node in the graph, called the *source nodes*, and each of the source message is demanded by one or more nodes, called the *demand nodes*. Nodes that are not source or demand nodes are called *intermediate nodes*. A node transmits a packet of information to another node if an edge exists between the two nodes. In other words, each out-edge of a non-demand node carries a packet. Lastly, in any single unit of time, only one packet can be sent through an edge and we further assume transmission of information between intermediate nodes takes negligible amount of time.

Remark 2.2 The alphabet set A is usually defined to be a finite field.

This definition of network in the network coding context is quite different from that of graph theory, in which it is usually defined as a (directed) graph. Furthermore, for notation matters, we define the following terms.

Notation 2.3 Given a network N. Let v be a node in N. We define $In(v)$ to be the set of packets arrived or originated at v, and $Out(v)$ to be the set of packets emitted from v. Furthermore, define $E_{in}(v)$ and $E_{out}(v)$ to be the in-edges and out-edges of v respectively.

In network routing, packets arrived or originated at a node can only be rerouted through its out-edges to other nodes. The node is not permitted to perform any other operations on the packets. More concretely, for all nodes v in the network that are not demand nodes, network routing has the restriction that the unique elements in $Out(v)$ is a subset of the unique elements in $In(v)$. If there exists a way such that all source messages are relayed to the correct demand nodes, we say the network has a pure routing solution.

However, in network coding, we relax the restriction and allow each node to perform "algebra", or manipulation, on the packets received before sending the packets out. To put it more formally, let v be a non-demand node in the network, then for each out-edge $e \in E_{out}(v)$, the packet being transmitted on e is defined as a function f_e on input $In(v)$. Also, for each demand node, say node d demanding messages $m_1, \cdots, m_k$, we assign k different functions to d with input $In(d)$, such that each function should yield a demanded message. Thus, we have the following definition on the solvability in the context of network coding.

Remark 2.4 We can in fact merge the definition of function assignment for both demand and non-demand nodes. For instance, given a demand node with request messages $m_1, \cdots, m_k$, we can do so by creating k auxiliary demand nodes such that the original node has an out-edge to each of the new demand nodes, where each of the new demand nodes ask for a requested message. This will give us a new directed acyclic graph, and there is only a solution in this new graph if and only if there is one in the original one.

One main class of network that will be analyzed throughout the project is the multicast network. For instance, we will be proving some important theorems about the solvability and computational complexity of multicast networks in section 3.1 and section 3.2 respectively.

Definition 2.5 A *multicast* network is a network with only one source node, where each demand node demands every source message.

2.2 Solvability of networks

Definition 2.6 A network has a *coding solution* if for every node n, there exists valid function assignments to each out-edge $e \in E_{out}(n)$ such that every demand node receives the requested source messages.

There are 2 main types of network coding solution, namely, routing solution and linear solution, where each imposes different restrictions on the assigned functions. Furthermore, each solution can either be a scalar or a vector solution.

Definition 2.7 A *routing solution* is a coding solution where the packets sent from a node, say n, must be strings of the message symbols in $In(n)$. In some ways, it is a deterministic shuffling of the characters in $In(n)$.

Definition 2.8 A *linear solution* is a coding solution where the packets sent from a node, say n, must be the result of some linear operations on $In(n)$. In particular, the underlying alphabet must be a finite field, and we can only use vector addition and vector multiplication by a constant matrix.

Example 2.9 Given a network N with some non-demand node n. In a routing solution, if $In(n) := \{abc, xyz\}$, we can mix and match the message and send axy as a packet. In a linear solution, if $In(n) := \{\begin{bmatrix}1 & 2 & 3\end{bmatrix}^T, \begin{bmatrix}4 & 5 & 6\end{bmatrix}^T\}$ with alphabet $A = \mathbb{F}_7$, then we can potentially send $\begin{bmatrix}1 & 2 & 3\end{bmatrix}^T + \begin{bmatrix}4 & 5 & 6\end{bmatrix}^T = \begin{bmatrix}5 & 0 & 2\end{bmatrix}^T$ as a packet.

Recall that a network has parameters (k, m) that determines the size of the messages and the size of the packets. In the case of scalar or vector solution, we are mainly concerned with the case of $k = m$.

Definition 2.10 Suppose a coding solution exists for a network with parameters (k, m). If $k = m = 1$, then it is called a *scalar solution*. Otherwise, if $k = m \geq 2$, then it is called a *vector solution*. In the case of $k \neq m$, we say it is a *fractional solution*.

Remark 2.11 A network that is scalar routing solvable is equivalent to having a network routing solution, as described in the introduction, where packets are only allowed to be relayed to other nodes without modification.

With the above definitions, we can finally define what it means for a network to be solvable.

Definition 2.12 A network is *solvable* if there is a scalar solution over some finite alphabet.

Remark 2.13 This is slightly misleading because there can be nonlinear coding solution to a network. However, for the rest of the project, we will be consistent with this definition and will address any inconsistency that arises.

Remark 2.14 If a network N has a vector solution, say it is of length k, over some finite alphabet A. Then, it has a scalar solution over the alphabet A^k, so it is solvable.

Note that for a sufficiently large packet size m, with respect to messages size k, we can fit all messages into any packet, so the network will be (k, m) solvable simply with a routing solution.

2.2.1 Some examples

To help illustrate the concepts mentioned above, we will introduce a few networks and show whether they are solvable.

- The butterfly network, as shown in figure 1.
- The M-network, as shown in figure 2.
- The unsolvable network, as shown in figure 3.

Note (Interpretation of the figures)
The label above a node denotes the source messages and indicates the node as a source node. Similarly, the label below a node denotes the demanded messages and indicates the node as a demand node.

Proposition 2.15 The butterfly network (figure 1) is scalar linear solvable, but not scalar routing solvable.

Proof. It is easy to see it is not scalar routing solvable. In order for message a to travel from node 1 to node 6, it must use the edge $e_{3,4}$. Similarly, it must use the edge $e_{3,4}$ for message b. But, by definition, only one packet can pass through an edge in any single unit of time, so a scalar routing solution is not possible.

To show it is linearly solvable, let's first assume A is a finite field. At n_1 and n_2, we have no choice but to emit a and b to their out-edges respectively. In node n_3,

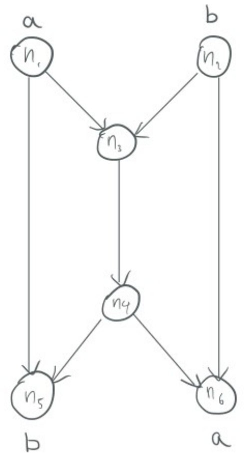

Figure 1: Butterfly network

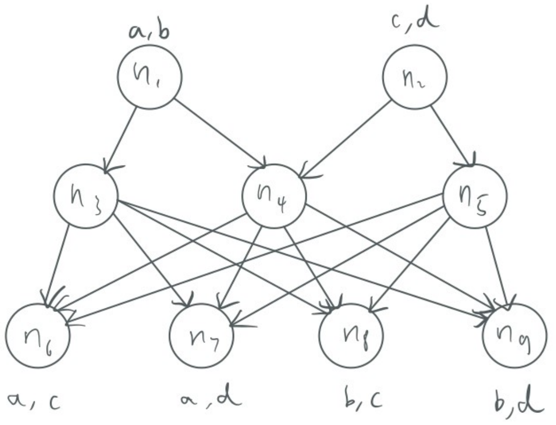

Figure 2: M-network

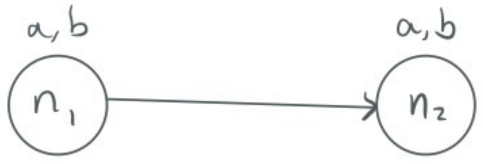

Figure 3: Unsolvable network

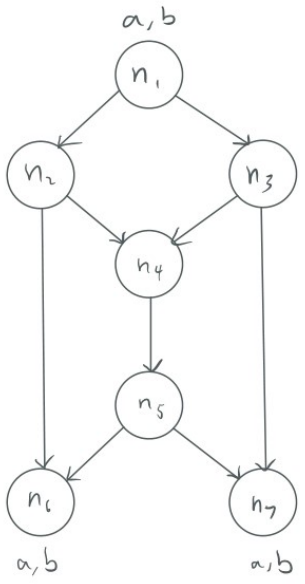

Figure 4: Multicast butterfly network

we have $In(n_3) = \{a, b\}$. Let $f_3 = (a + b)$, so n_3 transmits $a + b$ to n_4. n_4 simply relays the packet $a + b$ to n_5 and n_6. Now, note that $In(n_5) = \{a, a + b\}$, from which it can decode b, the demanded source message, from $-(a) + a + b$. More precisely, $-a + (a + b) = -Ia + (a + b)$ where I is the identity matrix of corresponding size over A. Similarly, we can decode a at node n_6. Thus, the butterfly network is scalar linearly solvable. □

Proposition 2.16 Let the M-network (figure 2) be over some finite alphabet set A. Then, it is solvable. In particular, it has a vector routing solution over A with $k = m = 2$, but it is not scalar solvable over A.

Proof. First, we show that it has a vector routing solution with $k = m = 2$. Recall that a routing solution allows mixing of messages and f_e denotes the function assignment on the edge e. Let the source messages be $a := (a_1, a_2)$, $b := (b_1, b_2)$, $c := (c_1, c_2)$, and $d := (d_1, d_2)$, where each element in the tuples is an element of A. To show it has a vector routing solution, it suffices to give an explicit function assignments

$$f_{e_{1,3}} = (a_1, b_2), f_{e_{1,4}} = (a_2, b_1), f_{e_{2,4}} = (c_1, d_2), f_{e_{2,5}} = (c_2, d_1).$$

Note that n_3 and n_5 receives only one input respectively, so the packets they transmit are simply relay of their inputs. Now, for the remaining n_4, we have the following assignments

$$f_{e_{4,6}} = (a_2, c_1), f_{e_{4,7}} = (a_2, d_2), f_{e_{4,8}} = (b_1, c_1), f_{e_{4,9}} = (b_1, d_2).$$

It is easy to verify from here that each demand node has sufficient information to construct the desired messages.

Now, we have to show that this is not scalar solvable. For the scalar routing case, we suppose without loss of generality that $In(n_4) := \{b, c\}$, $In(n_3) := \{a\}$, and $In(n_5) := \{d\}$. Then, it is clear that the demands for n_8 cannot be satisfied, since only n_4 has b and c as inputs, but n_4 cannot possibly transmit both to n_8 in a single time unit. Hence, it is not scalar routing solvable.

For the scalar linear case, we first assume n_4's output is a linear combination of the inputs, but this leads to a contradiction. Now, if n_4 can only relay input packets without mixing, then at least one demand node will not get the desired message, which is again a contradiction. One can refer to [12] for full details. □

Proposition 2.17 The unsolvable network as shown in figure 3 is not solvable.

Proof. Since we are considering the case $k = m$, only one packet of size same as the message size can be transmitted from n_1 to n_2. Clearly, n_2 cannot reconstruct 2 source messages from just one packet of information, so it is unsolvable. Note that if this is possible, we would essentially have a lossless compression algorithm that guarantees to compress any information to half its size. □

2.3 Capacity of networks

We have so far introduced the concept of network solvability, but we have not yet defined a way for measuring its effectiveness. To evaluate the effectiveness of a network, we need the notion of network capacity based on the coding solutions for the network.

Definition 2.18 If a network N over some alphabet A has a (k, m)-solution, then we say k/m is an *achievable rate* for N.

Definition 2.19 The *coding capacity* of a network over some alphabet A is the supremum over all ratios k/m for which k/m is an achievable rate. Furthermore, we define *routing capacity* and *linear coding capacity* for solutions taken over routing solutions and linear coding solutions respectively.

Remark 2.20 If A is not specified, then the network coding capacity is the supremum of the coding capacity over all possible alphabets.

Remark 2.21 I believe the use of supremum in the definition is because the choice of A, namely all possible finite fields, is infinite. It would help intuitively to just think of it as the maximum capacity over the capacity of all possible choice of A. One can refer to section 6.2 for a potential motivation for the definition.

Proposition 2.22 If a network is solvable, then the network capacity is at least 1.

Proof. A network is solvable if it has a scalar solution over some alphabet A, which means $k = m = 1$. So, the network capacity over A is at least $\frac{k}{m} = 1$, which further implies the network capacity in general is lower bounded by 1. $\square$

Proposition 2.23 Let N be a multicast network with assigned messages M. Then, N has network capacity at least $\frac{1}{|M|}$.

Proof. Let A be some alphabet set and s be the source node generating the messages M. Furthermore, let $M := \{m_1, \cdots, m_n\}$ for some $n \in \mathbb{Z}_{\geq 0}$ and suppose each message has size k. If, for all $e \in E_{out}(s)$, we define $f_e(In(s)) = m_1 \oplus m_2 \oplus \cdots \oplus m_n$, where $\oplus$ is simply string concatenation, and simply relay the same message over all intermediate nodes, then, each demand node will have sufficient information to recover the desired source messages. So, the packet size needed is at most kn, which means, the network is (k, kn) solvable, and thus, the capacity is at least $\frac{1}{n} = \frac{1}{|M|}$. $\square$

We have shown above some fairly simple upper and lower bounds on the network capacity. In section 6, we will discuss more results on network capacities for matroidal networks.

3 Network solvability and network flow

3.1 Solvability of multicast network

Recall from definition 2.5, a multicast network is a network with only one source node, where every demand node demands every source message. Multicast network is an important class of network that has been actively studied. An example of a multicast network is shown in figure 4. There are quite a few important results about the solvability of multicast networks, and one of the most well-known one is due to Li et al. [19], namely, every solvable multicast network is linearly solvable. To show this, we have to first introduce the max-flow min-cut analog for network coding and a concept called linear code multicast (LCM), and then, we can connect both ideas to the solvability of multicast networks.

3.1.1 Max-flow min-cut for network coding

In network flow theory, the *law of commodity flow* states that the in-flow volume of any set of nodes is at least that of the out-flow. It has a direct analog in the network coding settings, but phrased in terms of information. Roughly speaking, the *law of information flow* states that any information flowing out of a set of nodes can be reconstructed from the information flowing into the set of nodes.

Definition 3.1 Let N be a multicast network[1] with source node S. For any non-source node T, an S,T*-cut* is a set of nodes C that includes S but excludes T. The *value* of the S,T-cut is the number of edges with one end in C and another not in C.

Definition 3.2 An S,T*-flow* F is a set of edges E where each edge in E is at maximum capacity (i.e. information is being transmitted over that edge) while satisfying the following conditions,

1. The sub-network induced by E is acyclic.
2. Each node other than S and T has the same number of in-flowing edges and out-flowing edges in E.
3. The number of edges in E out-flowing from S is same as the number of edges in E in-flowing to T.

We say the edges in E are *busy edges*, and the *volume* of F is the number of busy edges with one end from S.

Definition 3.3 Fix a non-source node T. The maximum volume over all S,T-flows is defined to be $maxflow(T)$ and the minimum value over all S,T-cuts is defined to be $mincut(T)$.

[1]Here, we assume $k = m = 1$ over some alphabet A. In other words, each edge carries 1 alphabet.

Now, with all the necessary definitions introduced, we can finally state the theorem.

Theorem 3.4 Let S be a source node and T be a non-source node. Then,

$$maxflow(T) = mincut(T).$$

3.1.2 Linear code multicast

The main idea of a linear multicast is to assign a vector space to each node and a vector to each edge. In particular, the vectors corresponding to the in-edges of a node should span the vector space assigned to the node, which fits in well with the law of information flow.

Definition 3.5 Let Ω be a d dimensional vector space over a sufficiently large field where

$$d = \max\{maxflow(T) : T \text{ is a non-source node}\}.$$

A *linear code multicast* is a function v that assigns a vector space to each node and a d-dimensional vector to each edge, while satisfying the following properties,

1. $v(S) = \Omega$,
2. $v(XY) \in v(X)$ for every edge XY in the network, and
3. For all collections C of non-source nodes,

$$\text{span}\,\{v(T) : T \in C\} = \text{span}\,\{v(XY) : X \in C, Y \notin C\}.$$

We can interpret the third property as the law of information flow for linear coding, which states that the information flowing out of a set of non-source nodes can be derived from the information flowing into the set of nodes (Condition 3), since the out-flowing information depends on the information at the originating node (Condition 2). Furthermore, from this definition, we can see that the vector space assigned to a node is completely determined by its in-edges. Thus, for any non-source node, the function v is completely determined by the vector assignment to each of its in-edges. All these together imply that for any non-source node T, any out-flowing vector is assigned to be a linear combination of the in-flowing vectors.

Then, we have a natural scheme for message transmission over this LCM model. Let S be the source with information m. We encode m by a d-dimensional row vector, which we call the *information vector*. The packet of information being transferred at edge XY is defined to be the cross product $m \times v(XY)$. By above, we know that all out-flow vectors are just linear combinations of in-flow vectors, so this transmission scheme is a reasonable model for linear coding.

From [19], we know that given an LCM for some multicast network, $dim(v(T)) \leq maxflow(T)$ for all non-source node T. In order to get equality here, we need to restrict the class of LCM to generic LCM, as defined below.

Definition 3.6 A *generic LCM* v is an LCM where for any collection of edges,

$$E = \{X_1Y_1, \cdots, X_kY_k\} \text{ and } k \leq d = dim(\Omega),$$

we have $v(X_i) \not\subset \text{span}\{v(X_jY_j) : j \neq i\}$ for all $1 \leq i \leq k$ if and only if $\{v(e) : e \in E\}$ is linearly independent.

One can refer to Example 3.1 in [19] for an example of an LCM that is not generic.

Theorem 3.7 If v is a generic LCM, then $dim(v(T)) = maxflow(T)$ for all non-source node T.

Proof. Let C be a cut. We define $dim(C) = dim(\text{span}\,\{v(XY) : X \in C, Y \notin C\})$. Let N be a multicast network, V be its vertices, T be a non-source node, f be the value of $maxflow(T)$, and v be a generic LCM for N. It suffices to consider the case $dim(v(T)) \geq f$, since the other direction is true for any LCM.

First, suppose for a contradiction, $dim(v(T)) < f$. Then, there exists at least one S, T-cut U where $dim(U) < f$, since $dim(v(T)) < f$ implies $dim(V \setminus \{T\}) < f$. Choose U to be the minimal cut, in the sense that removing any node in the cut will result in it having dimension greater than f. Let C be the edges in the cut U, and B be the boundary nodes. These are the nodes where it is an endpoint of some edges in the cut. Then, for all $W \in B$, we can show that $v(W) \not\subset \text{span}\{v(XY) : XY \in C\}$ by a minimality argument on the choice of U.

Now, $v(W) \not\subset \text{span}\{v(XY) : XY \in C\}$ implies $v(W) \not\subset \text{span}\{v(XY) : XY \in (C \setminus \{WZ\})\}$ for any $WZ \in C$, since $(C \setminus \{WZ\}) \subset C$. This implies, from the definition of a generic LCM, any subset A of $\{v(XY) : XY \in C\}$ with $|A| \leq d$ is linearly independent. Since U is a cut, by max-flow min-cut, $|C| \geq f$ since $f = maxflow(T)$. Furthermore, since $d = maxflow(T)$ over all non-source node T, we have $d \geq f$. Consequently, $dim(U) = dim(\{v(XY) : XY \in C\}) = \min\{d, |C|\} \geq f$, which is a contradiction. So, $dim(v(T)) = maxflow(T)$ as desired. □

Given this property of a generic LCM, it is natural to ask whether there exists a generic LCM for all multicast networks. Yes, there exists, over a sufficiently large field. In fact, we have an algorithm that construct such a generic LCM for a given multicast network. First, we will state a few preliminary lemmas.

Lemma 3.8 Let N be a multicast network. There is a relabelling of nodes to $X_0, X_1, \cdots, X_n$ where every edge in the network goes from a smaller indexed node to a higher indexed node. If a network satisfies this property, we say it is *relabelled*.

Proof (Sketch).
One can use any topological sorting algorithm (i.e. DFS). However, note that this works only because underlying the multicast network is an acyclic graph. □

The following lemma is needed to show the validity of a particular step in the LCM construction algorithm. The goal of this is to show that for a vector space V, the number of lower dimensional subspaces required to cover it is proportional the size of its base field.

Lemma 3.9 Let V be a d-dimensional vector space over a field $\mathbb{F}$. Fix S to be a k-dimensional subspace of V where $k \leq d$. Let $m \in \mathbb{Z}_{\geq 0}$ with $m < |\mathbb{F}|$. Then, if $Y := \{Y_1, \cdots, Y_m\}$ is a set of m distinct subspaces, where $S \not\subset Y_i$ for all $1 \leq i \leq m$, then $S \not\subset \cup_{i=1}^m Y_i$.

Proof. The main idea of the proof is to consider $S \cap Y_i$ for each i. Note that $dim(S \cap Y_i) = dim(S) + dim(Y_i) - dim(S + Y_i) \leq \min\{dim(S), dim(Y_i)\}$. Since $S \not\subset Y_i$, $dim(S+Y_i) > \max\{dim(S), dim(Y_i)\}$, so $dim(S \cap Y_i) < \min\{dim(S), dim(Y_i)\}$, which implies $dim(S \cap Y_i) \leq dim(S) - 1 = k - 1$. Furthermore, note that $S \subset \cup_{i=1}^m Y_i$ if and only if $S = \cup_{i=1}^m (Y_i \cap S)$. Since $dim(S \cap Y_i) \leq k - 1$, we have $|S \cap Y_i| \leq |\mathbb{F}|^{k-1}$. Since S is a k-dimensional subspace, $|S| = |\mathbb{F}|^k$. Then, by a cardinality argument, since $m < |\mathbb{F}|$, $|S| = |\mathbb{F}|^k > m|\mathbb{F}|^{k-1} \geq |\cup_{i=1}^m (Y_i \cap S)|$. Consequently, $S \neq \cup_{i=1}^m (Y_i \cap S)$, so $S \not\subset \cup_{i=1}^m Y_i$. □

```
Data: A relabelled multicast network $N$ with vertices $V := \{X_0, \cdots, X_n\}$
      and edges $E$.
Result: A generic LCM $v$ for $N$.
Initialize $v(XY) = \mathbf{0}$ for all edges $XY \in E$
for ($i = 0$, $i \leq n$, $i++$) do
    Let $S = \{X_iY \in E\}$
    for edge $X_iY \in S$ do
        Let $w \in V(X_i)$ such that $w \notin \text{span}\{v(AB) : AB \in C\}$ for any set $C$ of
          $\leq d - 1$ edges that satisfies $v(X_i) \not\subset \text{span}\{v(AB) : AB \in C\}$
        Set $v(X_iY) = w$
    end
    Let $L = \text{span}\{v(e) : e$ are in-edges to $X_{i+1}\}$
    Set $v(X_{i+1}) = L$
end
```
Algorithm 1: Constructing LCM of a given network

Theorem 3.10 For a sufficiently large field $\mathbb{F}$, algorithm 1 constructs a generic LCM for any multicast network. [2]

Proof (Sketch).
The main idea of the algorithm is to assign a vector space to each node iteratively while maintaining the generic property. Now, one unclear part is how we are able to pick w in the algorithm. To see this, note that for any give node X_i, there are only finitely many, say n, sets C satisfying $v(X_i) \not\subset \text{span}\{v(XY) : XY \in C\}$. By applying lemma 3.9, if $|\mathbb{F}| > f(m)$ where $m = \sum_{i=1}^{d-1} \binom{|E|}{i}$, then $v(X_i) \not\subset \cup_C \text{span}\{v(XY) : XY \in C\}$, since n is upper bounded by m, which counts the number of possible

[2] We should emphasize here that our definition of multicast network is acyclic, which matches the necessary condition in the original paper.

$d-1$ choices of edges in the network. Hence, there exists some vector $w \in V(X_i)$ that is not in any of the linear span. It remains to show that this LCM is generic. One can show that any set of edges $\{X_1Y_1, \cdots, X_mY_m\}$ where $m \leq d$ that satisfies $v(X_i) \not\subset \text{span}\{v(X_jY_j) : j \neq i\}$ for all $1 \leq i \leq m$, is assigned linearly independent vectors under v, which can be shown by induction on m. Remaining details can be found in [19]. □

Theorem 3.11 Given a generic LCM v, every non-source node T can receive the source messages at a rate equal to $maxflow(T)$.

We omit the proof for this. One can refer to example 3.5 in [19] for details. With this theorem, we can finally prove the main result.

Corollary 3.12 Every solvable multicast network, over a sufficiently large base field, is linearly solvable.

Proof. By theorem 3.10 above, every multicast network over a sufficiently large base field has a generic LCM. Then, by theorem 3.11, a generic LCM provides guarantee that each node T can receive the source messages at a rate equal to $maxflow(T)$. Let N be a multicast network with source node S, transmitting an information vector of length k. Then, for any demand node T, a solvable multicast network cannot have an S,T-cut of value less than k, because it must be able to reconstruct k units of information from its in-flow. Then, by max-flow min-cut, the maximum S,T-flow is at least k. So, by the observation above, each node in a generic LCM is able to receive at a rate of at least k, which means every node will be able to receive the messages in a single unit time. Thus, N is linearly solvable. □

3.2 Computational complexity of network solutions

In this section, we will give a brief overview of the complexity of finding linear network coding solutions, following the paper of Lehman et al. [11] Our goal here is to classify the computational complexity of the solvability of various classes of networks. First, we characterize different classes of network using 4 different parameters.

Definition 3.13 (Classification of network structure)
For a given network N, N is in the class of $(\alpha, \beta, \gamma, \delta)$ according to the following classification.

- $\alpha = \begin{cases} S & \text{if } N \text{ has only one source node} \\ M & \text{if } N \text{ has more than one source node} \end{cases}$

- $\beta = \begin{cases} S & \text{if } N \text{ has only one demand node} \\ M & \text{if } N \text{ has more than one demand node} \end{cases}$

- $\gamma = \begin{cases} A & \text{if every source node has ALL the messages} \\ D & \text{if each source node has a DISJOINT set of messages} \\ N & \text{if there is NO GUARANTEE on the source nodes} \end{cases}$

- $\delta = \begin{cases} A & \text{if every demand node demands ALL the messages} \\ D & \text{if each demand node demands a DISJOINT set of messages} \\ N & \text{if there is NO GUARANTEE on the demand nodes} \end{cases}$

We say $(\alpha, \beta, \gamma, \delta)$ is a *network class.*

Then, we separate the potential complexity of solving the network into 3 main types. In particular, for each possible network class, we classify it into one of the three groups below.

Definition 3.14 (Classification of network complexity)
Given a network class C, it is associated with

- *Trivial*: If all solvable networks in C are trivially solvable by relaying messages without encoding.
- *Linear*: If some networks in C cannot be solved trivially and require linear coding.
- *Hard*: If some networks in C have linear coding solutions that are hard to find. Also, some networks are not linearly solvable, but are non-linearly solvable.

The terms *Trivial*, *Linear*, and *Hard* are called the *complexity classes* of C.

We will show an example of network class for each Trivial, Linear, and Hard. It would help illustrate how one could attempt to prove related results for other methods of network classifications, since each of the complexity class employs a different approach. Roughly speaking, the Trivial class utilizes ideas from network flow, the Linear class uses Sander-Li's algorithmic result[16], and the Hard class reduces to a 3-SAT instance to achieve NP-hardness.

3.2.1 Trivial complexity class

Notation 3.15 For a demand node n, we write $M_{demand}(n)$ as the number of messages n demands.

Theorem 3.16 The network class (M, M, A, D) is trivial.

Proof. Let N be a network in the class with n source nodes, m demand nodes, and k messages. If N is solvable, then we want to show that it is trivially solvable. First, we create an auxiliary network N' from N. Then, we create a source node S that generates all the messages, and S has k edges to each of the original source node.

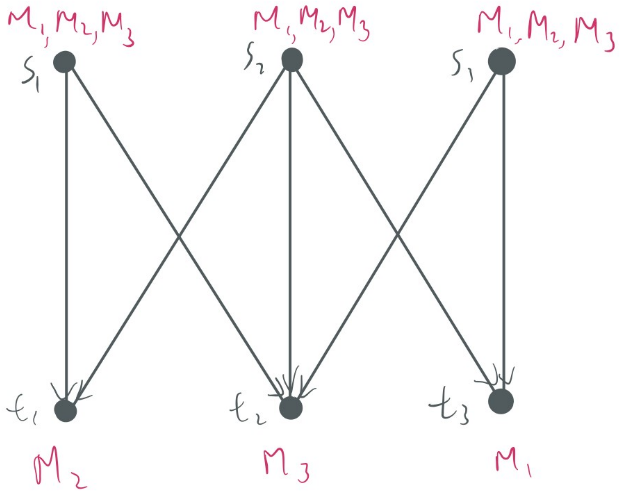

Figure 5: Original network N

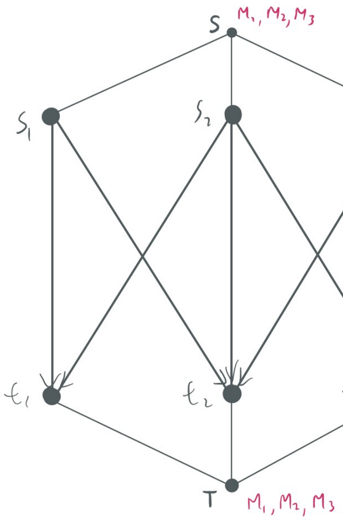

Figure 6: Auxiliary network N'

Also, we create a sink node T that demands all the messages and, for each original demand node t_i, there are $M_{demand}(t_i)$ edges from t_i to T. So, all of the original source and demand nodes are now simply intermediate nodes. (See figure 6 for an example) It is easy to see that N is solvable if and only if N' is solvable.

Now, if N' is solvable, then there must be a flow from S to T that transmits at least k messages. Furthermore, we realize that it implies there is no cut of size less than k. So, by Menger's theorem, we know there exists k-edge disjoint paths from S to T. Furthermore, in order for T to receive k messages, it must uses all of the $D(t_i)$ edges from t_i to T, for each original demand node t_i, which means all t_i has the an in-flow of $D(t_i)$ messages. Since every source node contains all of the messages, the choice of message to send from each of the k edge-disjoint paths can be chosen such that the original demands are satisfied. This completes the proof since that would imply there exists a flow in the original problem that flows from the source nodes to the corresponding demand nodes simply through routing. □

3.2.2 Linear complexity class

The main network class where one can solve a network in polynomial time using linear coding is (M, M, N, A). Sanders et al. [16] found a deterministic polynomial time algorithm to get a linear solution for multicast networks. However, the original result is for multicast networks, which means it only has a single source that generates all messages, whereas, we have multiple source nodes here. This can be easily generalized to the multiple sources case. We can reduce the network to a single source network by creating an auxiliary graph with a single source node that maps to all of the original source nodes. The remaining details can be found in [11].

3.2.3 Hard complexity class

Definition 3.17 Given variables $x_1, \cdots, x_n$, a 3-*CNF formula* is of the form $A_1 \wedge A_2 \wedge \cdots \wedge A_m$ where each $A_i = (l_j \vee l_k \vee l_l)$ and l_i is a literal with variable x_i, for $m \in \mathbb{N}$ and $1 \leq j, k, l \leq n$. A 3-*SAT problem* is one that asks whether a 3-CNF ϕ is satisfiable. In other words, it ask whether there is a truth value assignment to each variable such that ϕ evaluates to true.

Definition 3.18 Given a 3-CNF formula $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ with variables $x_1, \cdots, x_n$, we constructs the *reduction network* as follows:

1. For each variable x_i, we create 2 nodes s_i and r_i, and an edge from s_i to r_i.
2. For each clause $C_l = (l_j \vee l_k \vee l_l)$, where l_j, l_k, l_l are literals for variables x_j, x_k, x_l respectively, we create nodes u_l, v_l, and t_l.
   (a) We connect each of r_j, r_k, and r_l to t_l.
   (b) Similarly, we connect each of s_j, s_k, and s_l to both u_l and v_l.
   (c) Lastly, we connect u_l and v_l to t_l.
3. We let each s_i be a source node, generating messages M_i and $\bar{M}_i$.
4. Similarly, for each clause $C_l = (l_j \vee l_k \vee l_l)$ and assume, without loss of generality, $C_l = (x_j \vee \bar{x_k} \vee x_l)$, we let t_l be a demand node requesting messages M_j, $\bar{M}_k$, and M_l.

Lemma 3.19 Let ϕ be a 3-CNF formula. Then, ϕ is satisfiable if and only if the reduction network is linearly solvable.

Proof. Note that each clause contains 3 variables, so if ϕ is satisfiable, then at least 1 of the variable must be true, and at most 2 can be false. Let π be a satisfiable assignment to ϕ. For any variable x_i, if x_i is true in π, then we send the message M_i from s_i to r_i, and the message $\bar{M}_i$ to all other edges from s_i. Otherwise, if x_i is false, we send $\bar{M}_i$ to r_i, and M_i to all other edges.

Consider a clause in ϕ, say $C_l = (\underbrace{x_i}_{\text{literal 1}} \vee \underbrace{\bar{x_j}}_{\text{literal 2}} \vee \underbrace{x_k}_{\text{literal 3}})$, with corresponding node (t_l, u_l, v_l). Then, we consider the case for each literal in the clause. If the literal, say literal 1, is true under the assignment π, t_l will receive the corresponding message through using the $r_i t_l$ edge. Otherwise, if it is false, t_i will receive the message from one of the u_l or v_l node. By the observation above, since at most 2 variables in a clause can be false, we are always able to get the messages corresponding to the false variables from u_l and v_l. So, we have shown here that if ϕ is satisfiable, the reduction network has a linear solution.

For the other direction, we suppose there exists a linear solution to the reduction network, and then we construct a valid assignment π for the 3-CNF formula ϕ. If the

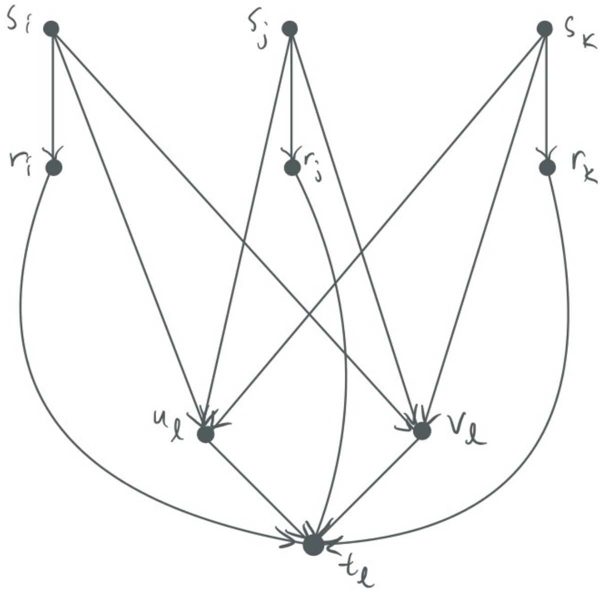

Figure 7: Example of a reduction gadget for the clause $C_l = (x_i \vee x_j \vee x_k)$

output of r_i only depends on the message M_i, we assign x_i to be true. Otherwise, if r_i depends only on message $\bar{M_i}$, we assign it to be false. If r_i is dependent on a linear combination of both M_i and $\bar{M_i}$, we assign the truth value of x_i arbitrarily.

Again, consider a clause $C_l = (x_i \vee x_j \vee x_k)$ with corresponding nodes (t_l, u_l, v_l). Let f_i, f_j, f_k denote the output functions from r_i, r_j, r_k to t_l respectively, and g, h denote the output functions from u_l and v_l to t_l respectively. Then, by a a simple result in [11] (Lemma 4.1), we know that one of f_i, f_j, f_k is a function dependent only on M_i, M_j, M_k respectively.

Note that, by design, the dependency of the f_i's corresponds to the truth value of x_i. For instance, if f_i depends only on $\bar{M_i}$, then we know the truth value assigned to x_i is false. By the construction of the reduction network, if t_l demands $\bar{M_i}, M_j, M_k$, then we know ϕ has the clause $(\bar{x}_i \vee x_j \vee x_k)$, which evaluates to true under π. Since this holds for all clauses in ϕ, ϕ is satisfiable under the assignment π. $\square$

Since the 3-SAT reduction network is in the class of (M, M, D, N), we have the following result as a direct consequence of the above lemma.

Theorem 3.20 Determining whether there is a linear coding solution to instances in the network class (M, M, D, N) is NP-hard.

4 Matroid fundamentals

4.1 Definitions and examples

We can think of a matroid as an abstraction of linear independence of vector spaces in linear algebra. More concretely, a matroid has a ground set S, and an independence relation on S, usually defined by an independent set. However, this independence relation can be defined in various other ways. Here, we provide 2 equivalent definitions of matroids, one using independent sets and the other using the matroid rank function.

Definition 4.1 A *matroid* is a pair $(E, \mathcal{I})$ consisting of a finite set E, called the *ground set*, and a collection $\mathcal{I}$ of subsets of E, called the *independent sets*, satisfying the following 3 axioms.

(I0) The empty set is independent

(I1) Subsets of independent sets are independent, and

(I2) For each $X \subseteq E$, all maximal independent subsets of X have the same size, denoted $r(X)$, which is called the *rank* of X.

Definition 4.2 A *matroid* is a set S equipped with a *rank function* r satisfying the following 3 axioms.

(R0) $r(\emptyset) = 0$

(R1) For $X \subseteq Y \subseteq S$, $r(X) \leq r(Y)$.

(R2) $r(X) + r(Y) \geq r(X \cup Y) + r(X \cap Y)$.

(R2) is also called the submodular inequality.

Remark 4.3 For representable matroids, we can think of the submodular inequality as a reinterpretation of the dimension theorem for linear subspaces, where r is the rank function for a subspace,

$$r(X) + r(Y) \geq r(X \cup Y) + r(X \cap Y).$$

There are many more cryptomorphically equivalent definitions for matroids. For instance, one can define it using bases, circuits, and hyperplanes. However, throughout the remainder of this project, we will be using the independent sets definition for matroid. One of the most important example of matroid is called the column matroid. The example demonstrates a clear relation between linear independence from linear algebra and independence in matroid.

Definition 4.4 Let A be a matrix over some field $\mathbb{F}$. Let $E := \{e_1, \cdots, e_n\}$ be the (multi)-set of columns of A. Then, the *column matroid* of A is $M := (E, \mathcal{I})$ where $\mathcal{I} := \{S \subseteq E : S \text{ is linearly indepedent}\}$.

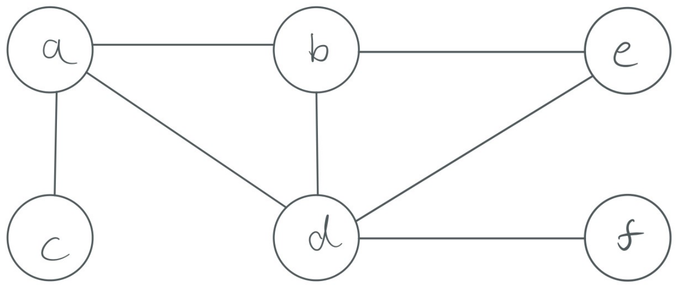

Figure 8: Example of a cycle matroid

Example 4.5 Let A be a matrix over $\mathbb{Z}$ where

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 2 \end{bmatrix}.$$

Then, the column matroid of A is $M := (E, \mathcal{I})$ where

- $E := \{\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 2 \end{bmatrix}\}$, and
- $\mathcal{I} := \{\emptyset, \{\begin{bmatrix} 1 \\ 0 \end{bmatrix}\}, \{\begin{bmatrix} 0 \\ 1 \end{bmatrix}\}, \{\begin{bmatrix} 0 \\ 2 \end{bmatrix}\}, \{\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}\}\{\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 2 \end{bmatrix}\}\}$.

Note The uniqueness of $S \in \mathcal{I}$ is determined by the index of the e_i's in E, not the actual content of the vector.

Another important example comes from graph theory, known as the cycle matroid. Given a graph, one can consider the edges as the ground set and all subsets which do not contain a cycle to be independent. This leads to a class of matroids known as graphic matroid, and many results from graph theory transfers to graphic matroid. In particular, if a result in graph theory can be rephrased in terms of edges only, such as the cycle exchange lemma, then there is a high chance that the result is transferable to the graphic matroid settings.

Definition 4.6 Let G be a graph. Let $M := (E, \mathcal{I})$ where $E := E(G)$ is the edge set of G and $\mathcal{I} := \{S \subseteq E : S \text{ does not contain a cycle}\}$. Then, we say M is a *cycle matroid* for G.

Example 4.7 Given a graph G, as defined in figure 8, we can define the cycle matroid $M = (E, \mathcal{I})$ as follows.

- $E := \{ab, ac, ad, bd, be, de, df\}$, and
- $\mathcal{I} := \{\text{all subsets of } E \text{ that do not induce a cycle}\}$.

Definition 4.8 A matroid is called *graphic* if it is a cycle matroid of some graph.

Some other examples of matroids that arise from graph theory include the tangle matroid, introduced by Robertson and Seymour in Graph Minors X. [15] There are low order decomposition theorem in graphs, such as decomposition into components and blocks for order 1 and 2 respectively, but the case for 4 or higher is not well defined. Roughly speaking, tangle is a dual notion of branch-decomposition in a graph and it generalizes the idea of high-connectivity components. For instance, elements in a tangle of order 1 has a bijective mapping to components in a graph.

Remark 4.9 One can also think of a hypergraph as a matroid without certain restrictions. In particular, a hypergraph $H = (X, E)$ is a matroid if E satisfies the independence axioms.

4.2 Representability of a matroid

4.2.1 Representable matroid

The idea of representable matroid will be used in later sections when discussing the capacity of matroidal networks.

Definition 4.10 A matroid is $\mathbb{F}$-*representable* if it is a column matroid over $\mathbb{F}$. A matroid is *representable* if it is $\mathbb{F}$-representable over some field $\mathbb{F}$.

Trivially, column matroids are representable.

Theorem 4.11 (Ingleton's inequality)
If X_1, X_2, X_3, and X_4 are sets of elements in a representable matroid, then

$$\begin{aligned}&r(X_1) + r(X_2) + r(X_1 \cup X_2 \cup X_3) + r(X_1 \cup X_2 \cup X_4) + r(X_3 \cup X_4) \leq \\ &r(X_1 \cup X_2) + r(X_1 \cup X_3) + r(X_1 \cup X_4) + r(X_2 \cup X_3) + r(X_2 \cup X_4).\end{aligned}$$

Ingleton's inequality is an important property of representable matroids because it can be used to show some matroids are not representable. For instance, we used it to show the Vámos matroid is not representable later. However, this is not a sufficient condition for a matroid to be representable.

Definition 4.12 We say a matroid is *binary* if it is $GF(2)$-representable.

Proposition 4.13 Graphic matroids are binary matroids.

Proof. Let M be a cycle matroid for a graph G. Then, the incidence matrix A for G is a binary representation of M. To show this, consider $v \in GF(2)^{|V|}$ and $v^T A$. The left null-space is the set of vectors v where $v^T A = 0$, which means v is a constant on each component of the graph, because we would get an entry of 1 somewhere in $v^T A$ otherwise. So, the dimension of the left null space of A is the number of components of G, which implies $rank(A) = |V| - \#$ of components of G. This means the rank is

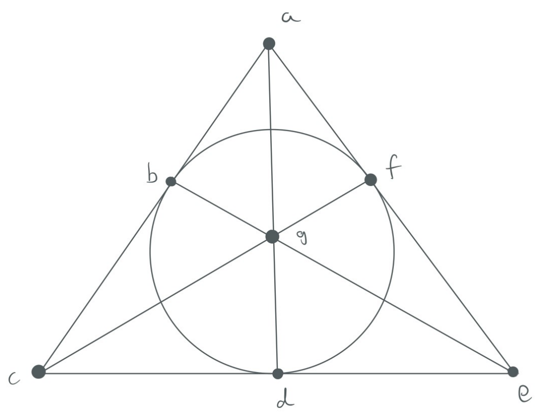

Figure 9: Fano matroid

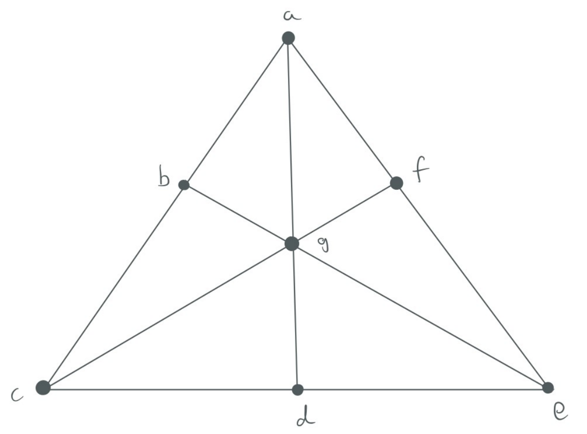

Figure 10: Non-Fano matroid

consistent with the rank of the set $E(G)$ of M, since the maximal spanning forest has $|V| - \#$ of components of G edges. Then, applying this argument to any subgraph of G, we have that the rank function is consistent with the corresponding submatrix of A. So, A is a binary representation of M. $\square$

Definition 4.14 A matroid is *regular* if it is $\mathbb{F}$-representable for all fields $\mathbb{F}$.

Proposition 4.15 All graphic matroids are regular.

Remark 4.16 The regular matroid decomposition theorem shows that nearly all regular matroids are constructed by gluing together graphic matroids in certain ways.

Definition 4.17 The *Fano matroid*, denoted $F_7 := (E, \mathcal{I})$, is defined as

- $E = \{a, \cdots, g\}$, and
- $\mathcal{I} = \{S \subseteq E : |S| \leq 3\} \setminus \{\{a,b,c\}, \{a,f,e\}, \{a,g,d\}, \{b,d,f\}, \{b,e,g\}, \{c,d,e\}, \{c,f,g\}\}$.

Pictorially, we can define $\mathcal{I}$ as the set of all sets of size at most 3 except the colinear triples (including the circle $\{b,d,f\}$) as shown in figure 9.

The *non-Fano matroid*, denoted F_7^-, is defined similarly. It is essentially the same as the Fano matroid, but with the set $\{b,d,f\}$ removed from $\mathcal{I}$, so the original circle linking $\{b,d,f\}$ is removed, as shown in figure 10.

Definition 4.18 The *Pappus matroid* is defined in figure 11 where

- $E := \{a, \cdots, i\}$, and
- $\mathcal{I} := \{$all sets of size at most 3 and are not colinear$\}$.

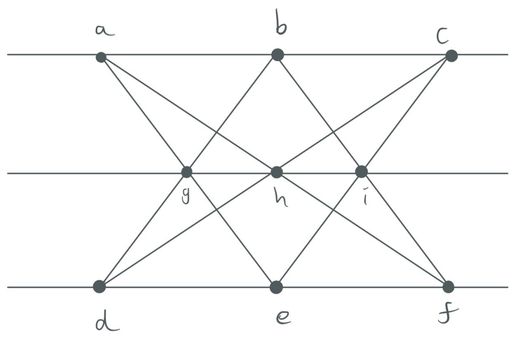

Figure 11: Pappus matroid

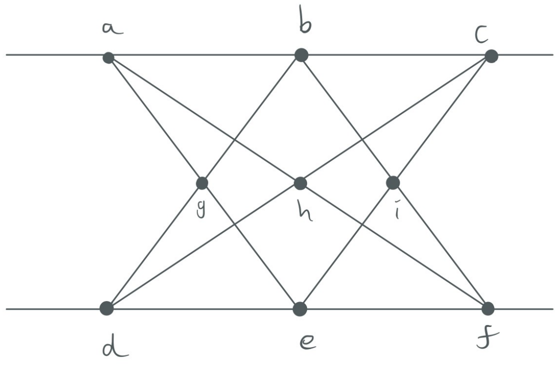

Figure 12: Non-Pappus matroid

The *non-Pappus matroid* is defined similarly, but with the line $\{g, h, i\}$ removed from $\mathcal{I}$, as shown in figure 12.

Proposition 4.19 The Fano matroid is binary.

Proof. One can easily check that the following matrix over $GF(2)$ is a valid column matroid representation

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}. \qquad \square$$

Remark 4.20 In fact, the Fano matroid is $\mathbb{F}$-representable if and only if $\mathbb{F}$ has characteristic 2. Furthermore, the non-Fano matroid is $\mathbb{F}$-representable if and only if $\mathbb{F}$ has characteristic other than 2.

Remark 4.21 We will just state without proof that the Pappus matroid is representable.

4.2.2 Non-representable matroid

Definition 4.22 The *Vámos matroid* is defined in figure 13 where

- $E := \{a, \cdots, h\}$, and
- $\mathcal{I} := \{$all sets of size at most 4 that are not coplanar$\}$. (The triangles, such as $\{a, c, e\}, \{b, d, h\}$, are not considered coplanar.)

Proposition 4.23 The Vámos matroid is not representable.

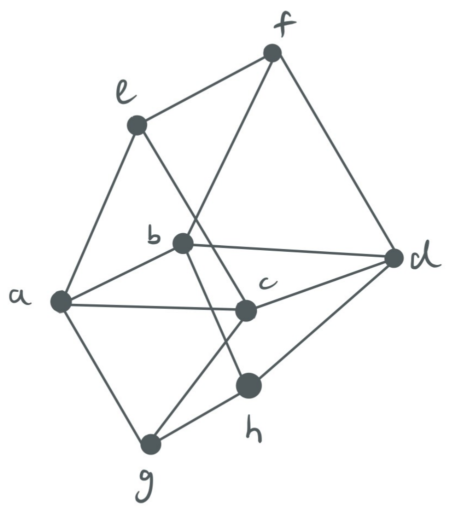

Figure 13: Vámos matroid

Proof ([13]).
Let $X_1 := \{a,b\}$, $X_2 := \{c,d\}$, $X_3 := \{e,f\}$, and $X_4 := \{g,h\}$. These four sets X_1, X_2, X_3, X_4 do not satisfy the Ingleton's inequality. In particular,

$$\begin{aligned}&r(X_1) + r(X_2) + r(X_1 \cup X_2 \cup X_3) + r(X_1 \cup X_2 \cup X_4) + r(X_3 \cup X_4)\\ &= 2 + 2 + 4 + 4 + 4 = 16\\ &\leq\\ &r(X_1 \cup X_2) + r(X_1 \cup X_3) + r(X_1 \cup X_4) + r(X_2 \cup X_3) + r(X_2 \cup X_4)\\ &= 3 + 3 + 3 + 3 + 3 = 15.\end{aligned}$$

Thus, the Vámos matroid is not representable. □

Remark 4.24 Again, without proof, we will just state that the non-Pappus matroid is not representable.

4.2.3 Computational complexity of representability testing

In analyzing matroidal algorithms, researchers often assume the matroid input is given by an independence oracle, where the oracle takes a set of matroid elements as input and returns whether the set is independent. Problems such as whether a matroid is representable over any field, fields of a certain characteristic, or even a given field $\mathbb{F}$ is in general very difficult. In particular, any independence oracle-based algorithm to solve the problems above for a general matroid M with n vertices will require at least $O(2^{n/2}/n^{1/2})$ time [18]. On the bright side, there are some representability problems that are polynomial-time solvable. For instance, there exist efficient algorithms for graphic matroids, due to Seymour [17], and algorithms for regular matroids, due to Truemper [18].

4.2.4 Number of representable matroids

At last, we mention a recent result by Nelson on an upper bound for the number of representable matroids [14].

Theorem 4.25 For sufficiently large n, there are at most $2^{n^3/4}$ representable matroids of size n.

A direct corollary of this is that the number of representable matroids with respect to all possible matroids is negligibly small, since it has been shown by Knuth [10] that there are around 2^{2^n} matroids of size n. So, representable matroids are indeed a really special class of matroids.

5 Linking network coding and matroid theory

We have introduced the two topics that are core to this project: network coding and matroid theory. However, we have yet to show how these two ideas connect. In this section, we will provide an algorithm for constructing networks from a given matroid. Throughout this process, some properties of matroids will be preserved to the constructed network, and thus allowing us to better understand some specific network coding problems.

5.1 Matroidal network

Definition 5.1 Let $\mathcal{N}$ be a network with messages M and packets P. Let $\mathcal{M} = (S, I)$ be a matroid with rank function r. A network is *matroidal* over M if there is a function $f : M \cup P \to S$ that satisfies the following properties.

1. $f|_M$ is injective,
2. $f(M)$ is independent ($f(M) = \{f(m) : m \in M\}$), and
3. $r(f(In(n))) = r(f(In(n) \cup Out(n)))$ for any node n.

The function f is also called the *network-matroid mapping*.

Remark 5.2 The definition here might seem a bit arbitrary. Properties (1) and (2) are some relatively natural definitions to make. We are just mapping messages to the ground sets and making sure these sources messages are independent in the matroid. However, property (3) is a bit more complicated. I think the intuition here is that we are capturing the idea of information with the rank function, and property (3) makes sure the law of information flow is satisfied. In particular, the out-flowing information at node n is dependent on the in-flowing information to n. Also, I believe the definition is related to the definition of polymatroid assignment definition that will be introduced later in section 6.

There is a nice result by Dougherty et al. [6] relating linear solvability and matroidal network, which allows us to potentially construct networks that are not linearly solvable by constructing a matroidal network from a matroid that is not representable.

Theorem 5.3 If a network is linearly solvable, then it is matroidal over a representable matroid.

Proof ([6]).
Let N be a network with source messages $M := \{m_1, \cdots, m_m\}$. Each outedge from node a to b is a function of $In(a)$, satisfying linearity. Let x_i be the set of variables representing source messages and packets (out-edges). Then, each x_i is in fact some linear combination of the messages, namely

$$x_i = c_1 m_1 + \cdots , c_m m_m.$$

In particular, $x_i = m_j$ for some j if x_i is a source message. Without loss of generality, suppose there are n out-edges. Let C_i denotes the vector $\begin{bmatrix} c_1 & \cdots & c_m \end{bmatrix}^T$ for x_i. Then, we construct the matrix C where

$$C = \begin{bmatrix} C_1 & | & \cdots & | & C_{n+m} \end{bmatrix}.$$

Let M be the column matroid for C with rank function r. We claim that N is matroidal over M, under the network mapping $f(x_i) = C_i$. It suffices to check each of the network-matroid mapping is satisfied.

1. It is clear that f is injective.

2. Note that if x_i is a source message, $f(x_i) = C_i = \underbrace{\begin{bmatrix} 0 & \cdots 0 & 1 & 0 & \cdots & 0 \end{bmatrix}^T}_{1 \text{ at the } j^{th}\text{-entry}}$ so $\{f(x_i) : x_i \text{ is a message}\} = \{e_0, \cdots, e_m\}$, where each e_i is the standard basis vector. So, $f(M)$ is clearly independent.

3. By definition, for any n, each $x_i \in Out(n)$ is a linear combination of packets or messages from $In(n)$. This implies $r(f(In(n))) = r(f(In(n) \cup Out(n)))$. □

Corollary 5.4 All solvable multicast network is matroidal.

Proof. In the corollary 3.12 from section 3.1, we show that all solvable multicast network is linearly solvable. Thus, by theorem 5.3 above, all solvable multicast networks are matroidal. □

5.2 Algorithmic construction of matroidal networks

A matroidal network can be constructed from a matroid using algorithm 2, as shown below. It should be noted that the algorithm takes in a matroid $\mathcal{M} := (S, I)$, and output a network $\mathcal{N}$, but the network constructed needs not be unique, since there are many choices one is allowed to make during the construction. Precisely, the output will be a network $\mathcal{N}$, with messages M, nodes N, and packets P, along with the network-matroid mapping f, and an auxiliary function g that maps elements in S from the matroid to the nodes N in the network. The construction of a matroidal network does not imply solvability, or in the very least, linear solvability. In fact, one can construct a matroidal network that has no linear solution using Fano and non-Fano matroids, as remarked in section 6.2.

Remark 5.5 In [5], it states that this algorithm works for most, but not all matroids. However, I currently do not have a concrete matroid to show that this algorithm fails.

```
Data: A matroid $\mathcal{M} := (S, I)$ with rank $r$.
Result: A matroidal network $\mathcal{N} := (N, M, P)$, network-matroid mapping $f$,
        and auxiliary function $g$.
Let $N := \{n_1, \cdots, n_r\}$, $M := \{m_1, \cdots, m_r\}$
Choose a basis $B = \{b_1, \cdots, b_r\}$
Let $f(m_i) = b_i$ and $g(b_i) = n_i$ for all $1 \leq i \leq r$

while there exists a circuit $\{x_0, \cdots, x_j\}$ where $g(x_0)$ is not defined, but
  $g(x_1) \cdots g(x_j)$ are all defined do
    /* For $1 \leq i \leq j$ */
    Add a new node $y$
    Add edges $e_1, \cdots, e_j$ with corresponding packets $p_1, \cdots, p_j$
    Connects $g(x_i)$ to $y$ using edge $e_i$
    Let $f(p_i) = x_i$

    /* For $i = 0$ */
    Add a new node $n_0$
    Add edge $e_0$ with corresponding packet $p_0$
    Connect $y$ to $n_0$ using edge $e_0$
    Let $f(p_0) = x_0$ and $g(x_0) = n_0$
end

/* Optional (As many times as desired) */
if $\{x_0, \cdots, x_j\}$ is a circuit where $g(x_0)$ is a source node then
    Let $m$ be the message associated with $g(x_0)$
    Add a new demand node $y$ that demands message $m$
    Add edges $e_1, \cdots, e_j$ with corresponding packets $p_1, \cdots, p_j$
    Connects $g(x_i)$ to $y$ using edge $e_i$
    Let $f(p_i) = x_i$
end

/* Optional (As many times as desired) */
if $\{x_1, \cdots, x_r\}$ is a basis then
    Add a new demand node $y$ that demands all source messages
    Add edges $e_1, \cdots, e_r$ with corresponding packets $p_1, \cdots, p_r$
    Connects $g(x_i)$ to $y$ using edge $e_i$
    Let $f(p_i) = x_i$
end
```

Algorithm 2: Network construction from a matroid

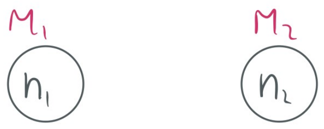

Figure 14: Step 1

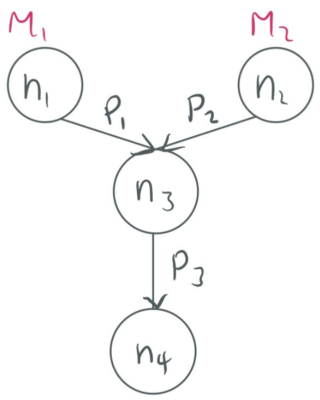

Figure 15: Step 2

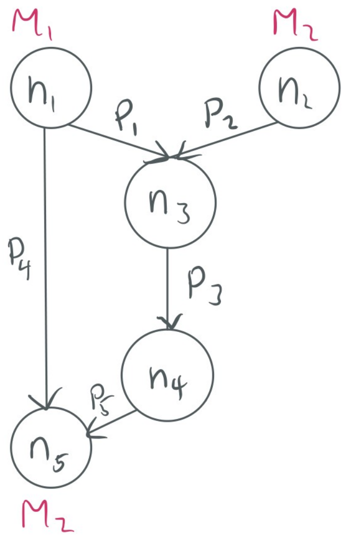

Figure 16: Step 3

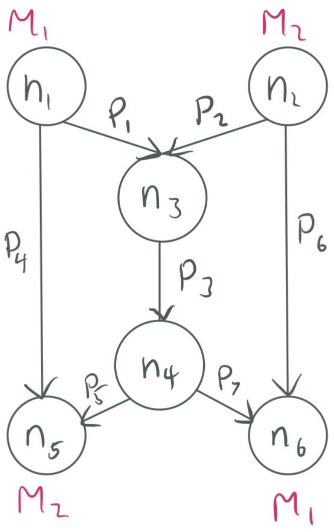

Figure 17: Step 4

5.3 Some examples

We will construct some matroidal networks using the algorithms above. In particular, we will show the construction of the butterfly network from the uniform matroid $U_{2,3}$. (See definition 5.6)

Definition 5.6 The *uniform matroid* $U_{r,n}$ is a matroid $(E, \mathcal{I})$, where $E := \{1, \cdots, n\}$ and $\mathcal{I} := \{\text{all subsets of size at most } r\}$.

We can follow the steps below to construct the butterfly network, as shown in 1, from $U_{2,3}$.

Step 1: First, we create nodes n_1, n_2 and messages m_1, m_2. Then, we choose a basis $B := \{1, 2\}$. For the network-matroid mapping f, we let $f(m_1) = 1$ and $f(m_2) = 2$. Lastly, for the node mapping, we let $g(1) = n_1$ and $g(2) = n_2$.

Step 2: The only circuit in $U_{2,3}$ is the ground set itself, namely $\{1, 2, 3\}$. Note that $g(3)$ is not defined, so we add a new node n_3 with edges n_1n_3 and n_2n_3, carrying packets p_1 and p_2 respectively. Then, we add a new node n_4, which is connected from n_3 and the edge n_3n_4 carries the packet p_3. Furthermore, we update the function f for the new packets so that $f(p_1) = 1$, $f(p_2) = 2$, and $f(p_3) = 3$. Lastly, we update the node mapping so that $g(3) = n_4$.

Step 3: Now, note that $\{2, 3, 1\}$ is a circuit with source node $g(2) = n_2$, which generates the message m_2. We add a new demand node n_5, with edges $g(1)n_5 = n_1n_5$ and $g(3)n_5 = n_3n_5$ carrying packets p_4 and p_5. This new node n_5 demands the message m_2. Finally, we update f where $f(p_4) = 1$ and $f(p_5) = 3$.

Step 4: Similar to step 3, we use the circuit $\{1, 2, 3\}$, which has the source node n_1 generating the message m_1. We add the demand node, new edges, new packets, and demand messages accordingly. At last, we update the function f for the new packets.

To help with understanding the algorithm, we can find a few more examples of matroidal networks constructed from the Fano, non-Fano, and Vámos matroid in [5].

6 Capacity of networks using matroid theory

6.1 Polymatroid upper bound

Definition 6.1 Let X be a discrete random variable. The *Shannon entropy function* (SEF) is defined to be

$$H(x) = \sum_{i=1}^{n} P(x_i) \log(P(x_i)),$$

where x_i are the possible outcomes of X and P is the probability function.

Given a solvable network N, if we consider our messages to be independent random variables and model the capacity of the network by measuring the entropy as the messages get transmitted through N, then it must satisfy the following properties, where H is the Shannon entropy function.

Definition 6.2 (Necessary condition for solvable network using SEF)

1. $H(M) = k|M|$ for any subset of messages M.
2. $H(p) \leq m$ for any packet p.
3. $H(In(n)) = H(In(n) \cup Out(n))$ for any node n.

Furthermore, a few additional *Shannon-type information inequalities* must be satisfied when we use Shannon entropy function on a set of random variables. These inequalities are sometimes known as the polymatroidal axioms. More formally, we have the following definition.

Definition 6.3 Given a finite set S and a function $f : S \to \mathbb{R}_{\geq 0}$. If f satisfies the following axioms,

N1: $f(\emptyset) = 0$,

N2: For $A \subseteq B \subseteq S$, $f(A) \leq f(B)$, and

N3: For $A, B \subseteq S$, $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$,

then, we say f is a *polymatroid*.

The main reason for this definition is to provide a reasonable capacity upper bound using polymatroid.

Definition 6.4 Let S be the set of all messages and packets over a network N. If a polymatroid f satisfies the necessary conditions stated in definition 6.2, where H is replaced with f, then, we say f is a *polymatroidal assignment* to the network N, with parameters (k, m). Then, the *polymatroid upper bound* is the supremum k/m over all pairs of (k, m) for which there exists a polymatroid assignment.

The upper bound makes sense because if there is a (k, m) solution to the network N, then the entropy function on the set of messages and packets is a (k, m)-polymatroid assignment, so the polymatroid upper bound on the capacity is at least the actual capacity of N. This provides the best upper bound on capacity that is "derivable from the Shannon-type information inequalities"[5], however, it is possible that one uses non-Shannon-type inequalities to reach a better upper bound, which will be briefly explained in section 6.3.

Theorem 6.5 For any matroidal network N, N has a polymatroid upper bound on the capacity of at least 1.

Proof. Let N be a matroidal network over matroid M. Let r be the rank function of the matroid M and f be the network-matroid mapping. Then, we can easily verify that $r \circ f$ is a polymatroid that also satisfies the necessary conditions. Since $f|_M$ is injective and $f(M)$ is independent, by N1, we have $(r \circ f)(M) = |M|$, which implies $k = 1$. By the rank axiom of matroid, $(r \circ f)(p) \leq 1$ since rank of one element is at most 1. Thus, $r \circ f$ is a $(1, 1)$-polymatroidal assignment, and we have the upper bound of at least 1 as required. □

6.2 Unachievable network capacity

From remark 4.20, we know that the Fano matroid and non-Fano are representable only for fields of characteristic 2 and not 2 respectively. Thus, it is tempting to see if this incompatibility result transfers over to the matroidal networks that are constructed from them. The following proposition from [5] shows that this is indeed true.

Proposition 6.6 The Fano and non-Fano network are not solvable using the same alphabet.

In particular, the Fano network is only solvable using alphabets A where $|A| = 2^k$ for some k, whereas, the non-Fano network is solvable when $|A| = 2k + 1$ for some k. It turns out, as shown in [4], this incompatibility result translates to unachievable capacity when considering the union of the networks.

Proposition 6.7 If N and M are two disjoint networks that cannot be solved using the same alphabets, then $N \cup M$ has coding capacity 1, but the capacity is not achievable.

Proof (Sketch).
Since N and M are solvable, each has capacity at least 1 respectively. Let A and B be some alphabets where N and M are solvable respectively, with $|A| \neq |B|$. By lemma 1 in [4], coding capacity of a network is independent of its alphabet size, so the coding capacity of M over A is at least 1 as well. Furthermore, let there be a (k, n) solution of M over A, then $\frac{k}{n} > 1 - \epsilon$ for any $\epsilon > 0$, since the capacity is at least

1. Note that $k < n$ is not possible. With some work, we can show that the coding capacity of $N \cup M$ is at least 1, since we have a $(k, \max\{k, n\})$ solution for M and a (k, k) solution for N over the alphabet A.

Now, suppose for a contradiction, there is a (k, n) solution for $N \cup M$ with $k \geq n$ over some alphabet C. Then, this implies there is a (k, k) solution for each N and M using C, contradicting our assumption that they cannot be solved using the same alphabet, namely C^k. Finally, lemma 2 in [4] states that every rational coding rate below the coding capacity is achievable. Thus, the coding capacity of $N \cup M$ is at most 1, since any (k, n) solution for it must satisfy $k < n$, meaning the supremum of all possible k/n is at most 1. Consequently, the coding capacity is exactly 1, but it is not achievable. □

As a result, the union of the Fano and non-Fano network has an unachievable network capacity of 1. Furthermore, we have fractional (k, m)-solutions to this network where k/m can be arbitrarily close to 1, but can never achieve 1. [3]

Remark 6.8 One can use a similar network construction to show that linear coding solution is insufficient. In particular, there is no (k, m)-fractional linear solution where k/m is arbitrarily close to 1. However, this network is shown to be non-linearly solvable.

6.3 Some remarks on the capacity of networks

There are various types of capacity bounds one can define for a network as shown in section 2.3. Here, we are mainly interested in routing capacity, linear coding capacity, and general coding capacity. Routing problem can be solved using multi-commodity flow techniques, and while it is possible to compute the capacity, there is no efficient algorithm to compute it in general. For both general and linear coding capacity, there is currently no known algorithm to compute them for an arbitrary network, regardless of efficiency.

For matroidal networks, we showed that the polymatroid upper bound on the capacity is at least 1. This is insufficient, however. For instance, in the case of Vámos network, the capacity is at most 1, which can be seen from a bottleneck edge in the network. This implies using Shannon-type information inequalities will not yield us a better bound. However, one can use non-Shannon type inequalities, such as the Zhang-Yeung inequality [20], to get a tighter capacity upper bound of $\frac{10}{11}$. Coincidentally, the Ingleton's inequality for matroid representability has also found its use case in linear coding. The implication of this is that the capacity region of linear coding can be strictly smaller than that of general network coding. In the Vámos network case, we are able to push the linear capacity bound down to $\frac{5}{6}$ using Ingleton's inequality, which is known to be tight since one is able to construct a $(5, 6)$-fractional linear solution.

[3] The definition of supremum in network capacity was not immediately clear to me initially, but I think cases like the one above motivate this definition.

7 Conclusion

In this project, we have shown above some unexpected connections between network coding, network flow, and matroid theory. It is a fascinating field of study with interesting interplay with other theoretical areas. Like many other fields, there are still a lot we do not know about. So, we would like to conclude by listing some interesting problems[4] in network coding and related areas.

1. How to find the general and linear coding capacity for an arbitrary network?
2. Does the Vámos network have a solution better than the linear solution?
3. What are the classes of networks that can be generated from algorithm 2? What are some non-trivial upper-bounds on the number of networks of size k that can be constructed from the algorithm? What if we restrict to only graphic matroids, since we can count the number of bases of a graphic matroid efficiently?
4. Are there some specific properties of graphic matroids that transfer nicely to matroidal networks, other than results that directly follow from representable matroids?

Acknowledgments

Thanks to prof. Alfred Menezes for introducing me to the topics covered in this project and for various discussions during office hours. Also, thanks to Steven Wong for helpful discussions on a few results in section 3.1.

All of the content is written the way I understand it from the corresponding references. Furthermore, proof for proposition 2.23, lemma 3.9, and corollary 3.12 are original, in the sense that I have not seen the proofs anywhere in the literature. Everything else without explicit reference should be credited to the main references stated in the introduction for each individual subsection.

I hope you enjoy reading this as much as I had fun learning and writing about it :)

[4] open problems, to my knowledge at least

References

[1] R. Ahlswede, Ning Cai, S.-Y.R. Li, and R.W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, 2000. doi: 10.1109/18.850663.

[2] Deepankar (Deep) Medhi and Karthikeyan (Karthik) Ramasamy. Preface (1st edition). In Deep Medhi and Karthik Ramasamy, editors, *Network Routing (Second Edition)*, The Morgan Kaufmann Series in Networking, pages xxix–xxxiv. Morgan Kaufmann, Boston, second edition edition, 2018. ISBN 978-0-12-800737-2. doi: https://doi.org/10.1016/B978-0-12-800737-2.00038-7. URL `https://www.sciencedirect.com/science/article/pii/B9780128007372000387`.

[3] Alexandros G. Dimakis, P. Brighten Godfrey, Yunnan Wu, Martin J. Wainwright, and Kannan Ramchandran. Network coding for distributed storage systems. *IEEE Transactions on Information Theory*, 56(9):4539–4551, 2010. doi: 10.1109/TIT.2010.2054295.

[4] R. Dougherty, C. Freiling, and K. Zeger. Unachievability of network coding capacity. *IEEE Transactions on Information Theory*, 52(6):2365–2372, 2006. doi: 10.1109/TIT.2006.874405.

[5] R. Dougherty, C. Freiling, and K. Zeger. Network coding and matroid theory. *Proceedings of the IEEE*, 99:388–405, 2011.

[6] Randall Dougherty, Chris Freiling, and Kenneth Zeger. Networks, matroids, and non-shannon information inequalities. *IEEE Transactions on Information Theory*, 53(6):1949–1969, 2007. doi: 10.1109/TIT.2007.896862.

[7] Attilio Fiandrotti, Valerio Bioglio, Marco Grangetto, Rossano Gaeta, and Enrico Magli. Band codes for energy-efficient network coding with application to p2p mobile streaming. *IEEE Transactions on Multimedia*, 16(2):521–532, Feb 2014. ISSN 1941-0077. doi: 10.1109/TMM.2013.2285518.

[8] Mohammad Hamed Firooz and Sumit Roy. Data dissemination in wireless networks with network coding. *IEEE Communications Letters*, 17(5):944–947, 2013. doi: 10.1109/LCOMM.2013.031313.121994.

[9] Jim Geelen. Lectures in matroid theory, 2021.

[10] Donald E Knuth. The asymptotic number of geometries. *Journal of Combinatorial Theory, Series A*, 16(3):398–400, 1974. ISSN 0097-3165. doi: https://doi.org/10.1016/0097-3165(74)90063-6. URL `https://www.sciencedirect.com/science/article/pii/0097316574900636`.

[11] April Rasala Lehman and Eric Lehman. Complexity classification of network information flow problems. In *Proceedings of the Fifteenth Annual ACM-SIAM*

Symposium on Discrete Algorithms, SODA '04, page 142–150, USA, 2004. Society for Industrial and Applied Mathematics. ISBN 089871558X.

[12] Muriel Medard, Michelle Effros, David Karger, and Tracey Ho. On coding for non-multicast networks. In *IN PROC. 41ST ANNUAL ALLERTON CONFERENCE ON COMMUNICATION, CONTROL AND COMPUTING*, 2003.

[13] Peter Nelson. Lots of ingleton matroids, Oct 2017. URL `http://matroidunion.org/?p=2285`.

[14] Peter Nelson. Almost all matroids are nonrepresentable. *Bulletin of the London Mathematical Society*, 50(2):245–248, 2018. doi: https://doi.org/10.1112/blms.12141. URL `https://londmathsoc.onlinelibrary.wiley.com/doi/abs/10.1112/blms.12141`.

[15] Neil Robertson and P.D Seymour. Graph minors. x. obstructions to tree-decomposition. *Journal of Combinatorial Theory, Series B*, 52(2):153–190, 1991. ISSN 0095-8956. doi: https://doi.org/10.1016/0095-8956(91)90061-N. URL `https://www.sciencedirect.com/science/article/pii/009589569190061N`.

[16] Peter Sanders, Sebastian Egner, and Ludo Tolhuizen. Polynomial time algorithms for network information flow. In *Proceedings of the Fifteenth Annual ACM Symposium on Parallel Algorithms and Architectures*, SPAA '03, page 286–294, New York, NY, USA, 2003. Association for Computing Machinery. ISBN 1581136617. doi: 10.1145/777412.777464. URL `https://doi.org/10.1145/777412.777464`.

[17] P. D. Seymour. Recognizing graphic matroids. *Combinatorica*, 1(1):75–78, March 1981. doi: 10.1007/bf02579179. URL `https://doi.org/10.1007/bf02579179`.

[18] K. Truemper. On the efficiency of representability tests for matroids. *European Journal of Combinatorics*, 3(3):275–291, 1982. ISSN 0195-6698. doi: https://doi.org/10.1016/S0195-6698(82)80039-5. URL `https://www.sciencedirect.com/science/article/pii/S0195669882800395`.

[19] Shuo yen Robert Li, Raymond W. Yeung, and Ning Cai. Linear network coding. *IEEE TRANSACTIONS ON INFORMATION THEORY*, 49(2):371–381, 2003.

[20] Zhen Zhang and R.W. Yeung. On characterization of entropy function via information inequalities. In *Proceedings. 1998 IEEE International Symposium on Information Theory (Cat. No.98CH36252)*, pages 375–, 1998. doi: 10.1109/ISIT.1998.708980.