Network Coding

by

April Rasala Lehman

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2005

Author .....
Department of Electrical Engineering and Computer Science
January 31, 2005

Certified by .....
Madhu Sudan
Associate Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by.....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Network Coding

by

April Rasala Lehman

Submitted to the Department of Electrical Engineering and Computer Science
on January 31, 2005, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

In the *network coding problem*, there are k commodities each with an associated message M_i, a set of sources that know M_i and a set of sinks that request M_i. Each edge in the graph may transmit any function of the messages. These functions define a network coding *solution*.

We explore three topics related to network coding. First, for a model in which the messages and the symbols transmitted on edges are all from the same alphabet Σ, we prove lower bounds on $|\Sigma|$. In one case, we prove $|\Sigma|$ needs to be doubly-exponential in the size of the network. We also show that it is NP-hard to determine the smallest alphabet size admitting a solution.

We then explore the types of functions that admit solutions. In a *linear solution* over a finite field $\mathbb{F}$ the symbol transmitted over each edge is a linear combination of the messages. We show that determining if there exists a linear solution is NP-hard for many classes of network coding problems. As a corollary, we obtain a solvable instance of the network coding problem that does not admit a linear solution over any field $\mathbb{F}$.

We then define a model of network coding in which messages are chosen from one alphabet, Γ, and edges transmit symbols from another alphabet, Σ. In this model, we define the *rate* of a solution as $\log |\Gamma| / \log |\Sigma|$. We then explore techniques to upper bound the maximum achievable rate for instances defined on directed and undirected graphs. We present a network coding instance in an undirected graph in which the maximum achievable rate is strictly smaller than the sparsity of the graph.

Thesis Supervisor: Madhu Sudan
Title: Associate Professor of Electrical Engineering and Computer Science

Acknowledgments

I wish to thank my advisor Madhu Sudan. His help with technical matters shows its mark clearly on my work but it was the quiet encouragement and support that were most important to me. My thesis readers, Dan Spielman and Muriel Médard, provided very useful feedback on drafts of this work. Over the last six years, Cliff Stein, Anna Karlin and Gordon Wilfong all went well beyond their official roles as mentors. The work presented in this thesis is the result of collaborative efforts. Nick Harvey and Robert Kleinberg have made the closing stretches of graduate school the most enjoyable. Other fellow students, past and present, haven been the people who drew me back to work each day. In particular, I would like to thank David Liben-Nowell and Adam Smith for their friendship. Be Blackburn, Joanne Talbot and Kathleen Dickey always seemed to be looking out for me. Abby Marsh, Alison Cohen and Taylor Schildgen provided many excellent diversions from work. I have been extremely lucky to have my family close by. I wish to thank my father for his encouragement along the way and interest in my work. Finally, Eric Lehman is an official collaborator on part of the work in this thesis and an unofficial collaborator in all that I do. His support, understanding and love are extraordinary.

Contents

Chapter 1

Introduction

How much information can be transmitted through a network? This fundamental question has recently received renewed attention.

Information in a network is often viewed as an immutable commodity. Data packets in a network are modeled in the same way as cars in a highway system or water in a system of pipes. However, information is fundamentally different from traffic or a fluid in two respects. A car reaching an intersection can proceed along only one road. In contrast, information can be duplicated and transmitted across many channels in a network simultaneously. In addition, information can also be manipulated in ways with no physical analogue. For example, if a network router receives bits x and y, then it can transmit a function, say $x \oplus y$, of the two bits.

Does *coding* information in this way provide any advantage? The example in Figure 1-1 shows that it can [1]. In this example, the node a has two bits of information, x and y. Nodes f and g both want these two bits. Each directed edge represents a channel that can transmit one bit of information.

Let's first look for a naive solution. Suppose each edge transmits one of the two bits. Then, in particular, edge (d, e) transmits either x or y. In both cases, one of the two sinks receives one bit twice and the other bit not at all. Apparently we are stuck.

However, Figure 1-1 shows a way around this difficulty. The label on each edge specifies the function of x and y transmitted over that edge. Node a sends bit x on one out-edge and bit y on the other. Nodes b and c copy the bit they receive to both out-edges. The key is that edge (d, e) transmits the function $x \oplus y$ rather than either bit x or y alone. Finally, node e copies the bit it receives to both out-edges. As a result, node f receives both x and $x \oplus y$ and so it can *compute* the value of y as well. Similarly, node g can compute x from y and $x \oplus y$ and thus also obtain both bits.

This example reinvigorated the field of network information theory. The generalization of this technique is now called *network coding*. Evidently, allowing edges to transmit functions can increase the amount of information that can be sent through a network. This

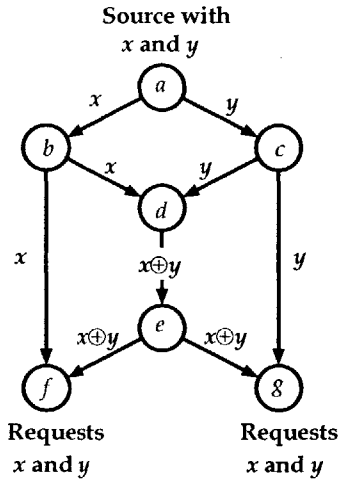


Figure 1-1: The node a is a source with two bits, x and y, of information. The two bottom nodes, f and g, need to each receive both bits. The shown solution sends the exclusive or of the two bits down the middle edge from d to e.

observation raises some interesting questions. We give preliminary answers here and provide more detail throughout the thesis.

How useful is network coding? When is it useful to transmit functions on the edges of a network? How much more information can be sent in this way? The benefit can be huge. In some instances network coding can increase the amount of information transmitted through a network by a factor of $\Omega(n)$ where n is the number of nodes [11, 25].

Do we need to use more complicated functions? In the example, each edge transmits a single bit, and the only nontrivial function is xor. Could more complex schemes be required for more complicated networks? As it turns out, working with bits is not sufficient for some problems; we must use an alphabet larger than $\{0, 1\}$. Furthermore, functions more complex than xor may be required.

How hard is finding suitable edge functions? Many traditional routing problems are computationally hard [2, 16, 20]. In contrast, for some interesting classes of problems, appropriate edge functions can be found efficiently [19, 15, 10]. In these cases, network coding not only increases the throughput, but also makes the problem efficiently solvable. However, determining whether suitable edge functions exist for the general problem could be anywhere from easy (in P) to undecidable [22].

What are the node requirements? How computationally powerful must nodes be in order to implement network coding? What are the memory requirements? In our example, node d must be able to compute $x \oplus y$, whereas a traditional router only copies and forwards data. Furthermore, node f must be able to store both x and $x \oplus y$ in order to compute y. For some restricted versions of the problem, upper bounds on computational requirements are known but no bounds have been found for the general problem. We present evidence that computational requirements may be large.

What about more general networks? The network in Figure 1-1 is directed and acyclic, but most real networks are not. Is network coding useful if the graph contains directed cycles? What if the network is undirected? Even formulating these questions rigorously requires considerable care. For directed graphs with cycles, the answer is yes. Network coding in undirected graphs is less understood and a major topic of this thesis.

In the remainder of this chapter, we define a model for network coding and discuss some of its aspects. Next, we define two restricted problems of particular interest. We then consider the history of network coding and conclude by outlining the main contributions of this thesis.

1.1 Definitions

In this section we define a model for network coding in directed acyclic graphs. There are three parts to our model. First, we define the type of communication problem that we consider. Then we define the concept of a network code. Lastly, we define what it means for a network code to be a solution to a communication problem. The model presented in this section is relatively simple but adequate for Chapters 2 and 3. In Section 1.2 we discuss the limitations of this model, and in Chapters 4 and 5 we introduce more general models.

Definition 1 (Instance of the Network Coding Problem) *An instance of the network coding problem is specified by the following:*

- *A directed acyclic graph* $G = (V, E)$.
- *A nonnegative integer capacity* $c(e)$ *for each edge* e.
- *A set* $\mathcal{I}$ *consisting of* k *commodities.*
- *For each commodity* $i \in \mathcal{I}$, *a set of sources* $\mathbf{S}(i) \subseteq V$ *and a set of sinks* $\mathbf{T}(i) \subseteq V$.

In the example in Figure 1-1, each edge has capacity 1. There are two commodities and node a is the source for both. Therefore, $\mathrm{S}(1) = \mathrm{S}(2) = \{a\}$. Nodes f and g are the sinks for both commodities. Thus, $\mathrm{T}(1) = \mathrm{T}(2) = \{f, g\}$.

Let's take a minute to discuss the model in terms of an actual communication problem. The graph G represents the network with routers or computers represented by nodes in V and communication channels between some computers, represented by edges. Each commodity models a file that must be transmitted through the network. Node a represents a computer that has a copy of two files, one file for each commodity. Nodes f and g represent two other computers that want copies of both files.

Definition 2 (Network Code) *A network code is defined by:*

- *An alphabet* Σ.
- *For each edge* $e \in E$, *a function* $f_e : \Sigma^k \rightarrow \Sigma^{c(e)}$.

Note that a network code is just an alphabet and a set of edge functions. In a moment we will define a *network coding solution*, which incorporates the requirements that a network code must satisfy if it "solves" the given instance.

In Figure 1-1 the alphabet is $\Sigma = \{0, 1\}$ and the label next to an edge specifies the function associated with that edge. For example, the function $f_{(b,d)} : \Sigma^2 \rightarrow \Sigma$ associated with edge (b, d) is defined by $f_{(b,d)}(x, y) = x$.

We return to our physical interpretation of this example. The files associated with each commodity are represented as a sequence of symbols from Σ. It is important to note that in designing a network code, we choose the alphabet Σ. In this way, the problem of transferring files is reduced to the problem of transferring a single symbol from each file. We call these single symbols *messages*.

Formally, if the network code uses an alphabet Σ, then associated with each commodity i is a symbol $M_i \in \Sigma$ which must be transmitted from the sources to the sinks. We refer to M_i as the message for commodity i, and let $M = (M_1, M_2 \ldots M_k)$ be the k-tuple of messages associated with the k commodities. For convenience, we say a source for commodity i is a *source for message* M_i and a sink for commodity i *requests message* M_i.

The domain for each edge function is the set of all possible k-tuples of messages. The function f_e specifies, for each possible combination of messages M, the $c(e)$ symbols from Σ sent over edge e.

Note that the definition of a network code does not require that the code be implementable or that sinks receive the requested information. For example, Figure 1-2 depicts a different network code for the same instance. The code in Figure 1-2 has two flaws. First, node b cannot compute the function $x \wedge y$, which it must transmit on edge (b, d), from the information it receives. Second, the sinks do not receive enough information to determine uniquely both messages. For example, if $x = 0$ then node f cannot determine the value of

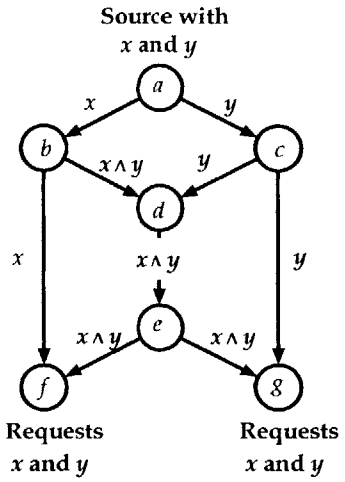


Figure 1-2: The source a has two bits of information, x and y, and the nodes f and g need to receive both bits. The edges are labeled with functions which determine a network code which is not a network coding solution.

y from x and $x \wedge y$. We define a network code to be a solution if it possesses neither of these flaws.

Definition 3 (Network Coding Solution) *A network code is a* solution *to an instance of the network coding problem if for every k-tuple of messages* $M = (M_1, M_2 \ldots M_k)$*:*

- *For every edge* $(u, v) \in E$*, the function* $f_{(u,v)}$ *is computable from the functions on in-edges to* u *and messages for which* u *is a source.*
- *For every sink* v *for commodity* i*, the functions on in-edges to* v *together with the messages for which* v *is a source are sufficient to determine the value of* M_i*.*

Our intuition is that the code in Figure 1-1 is a solution. To verify this, we must check that every edge function $f_{(u,v)}$ is computable from the information on the in-edges to u and the messages for which u is a source. Node a is the only source and it is a source for both messages. Therefore the edges (a, b) and (a, c) can be assigned any function of x and y. Nodes b, c and e copy the message received on the single in-edge to both out-edges. Node d receives both x and y on in-edges and therefore can transmit any function of x and y on edge (d, e). Next we check that each sink can compute the requested information. Node f receives x unencoded on edge (b, f) and the function $x \oplus y$ on edge (e, f). From this information it can compute y. Similarly for node g. Therefore this network code meets the definition of a network coding solution.

1.2 Discussion of Model

This model has been used in almost all work to date on network coding and is the model we use for the next two chapters. However, it has two drawbacks. First, the model requires that the communication network be directed and acyclic. Second, for the general network coding problem there is no notion of the rate of a solution. These drawbacks are discussed more extensively below. A contribution of this thesis is a clarification of these issues and more general models that address them.

1.2.1 Cycles

We model a communication network as a directed acyclic graph. Why is this necessary? Without this restriction, a solution, under our definition, may be meaningless in practice.

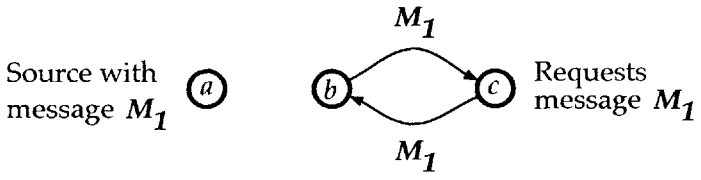


Figure 1-3: The functions on the edges in the graph meet the definition for a network coding solution. However, there is actually no way to transmit a message from node a to node c.

As an example, consider the instance in Figure 1-3. Message M_1 must be sent from node a to node c. A "solution" is to send M_1 along both edge (b, c) and edge (c, b). This network code meets the definition of a network coding solution. Nodes b and c can each compute the function on their out-edge from the function on their in-edge. However, there isn't even a path from the source a to the sink c. Therefore, there is no practical way to implement this so-called solution.

There is a natural suggestion to address this issue. Suppose we impose a total order on the edges in G and require that the function assigned to edge (u, v) be computable from the in-edges to u that come before edge (u, v) in the ordering. This restriction does remove the issue of cyclic dependencies between edge functions. However, this is too restrictive a condition because it also rules out obvious solutions in other instances. For example, the instance in Figure 1-4 has four commodities each with a single source and single sink. Each edge has capacity two which means that two symbols can be transmitted on each edge. As a practical matter, this problem can be solved. As indicated in the diagram, there is enough capacity for each message to be sent two hops clockwise to its destination. However, in this scheme, every edge function depends on the preceding edge in the cycle. Therefore there is no total order of edges under which this scheme is allowed.

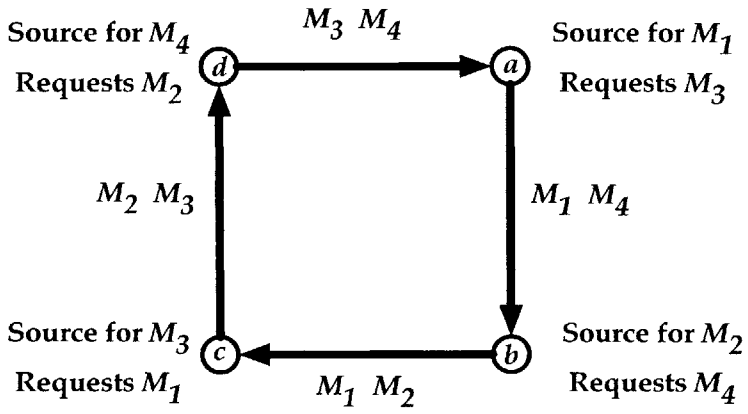


Figure 1-4: In the example, each edge has capacity two. There are four commodities. For each commodity there is a single source and a single sink two hops clockwise from the source. There is no ordering on the edges which allows one to define a solution for this instance.

The examples in Figures 1-3 and 1-4 suggest some of the difficulties in extending our simple definition for a network code and network coding solution in a directed acyclic graph to graphs with cycles. One solution is to associate a non-zero delay with each edge and consider the behavior of the network over time. We address these issues more extensively in Chapter 5.

1.2.2 Rate

The questions posed within this model are feasibility questions: does an instance have a solution? To clarify the issue, consider the instance in Figure 1-5, which is similar to the example in Figure 1-1. Now there are four commodities, but each edge still has capacity 1.

Is there a solution? In our model, no. There are two out-edges from the node a. Therefore, a network coding solution would need to encode the value of four messages using just two symbols. Clearly this is impossible.

On the other hand, in practice it is possible. Let's reconsider our physical interpretation of network coding. Node a represents a computer which receives four files, and nodes f and g want copies of these files. We know node a can send *two* files to f and g using the solution in Figure 1-1. Therefore, in practice, it can send four files, it just takes twice as long.

In Chapter 4, we extend our model to define the rate of a solution. For example, under the extended model, there exists a rate $1/2$ solution to the instance in Figure 1-5. In the

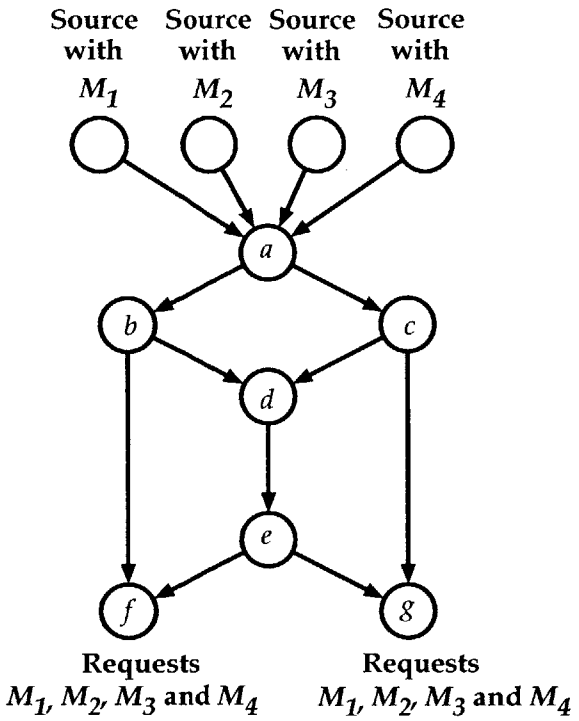


Figure 1-5: In the example, each edge has capacity one. There are four commodities each with their own source. Nodes f and g are the sinks.

next section we discuss the one situation in which the current model includes (implicitly) a notion of rate of a solution.

1.3 Two Restricted Network Coding Problems

In this section we define two restricted versions of the network coding problem. Both are fundamental problems in network communication. The first, the multicast problem, has been the focus of much of the work in network coding. All positive network coding results, such as efficient network coding algorithms, apply only to the multicast problem and small generalizations. The second problem we define is the k-pairs communication problem, which is closely related to the classical multicommodity flow problem. The applicability of network coding to the k-pairs communication problem is not well understood and will be the focus of Chapters 4 and 6.

1.3.1 Multicast Problem

In an instance of the *multicast* network coding problem, there is a single source for all messages and a set of sinks which request all messages. Figure 1-6 is an example of the multicast problem. Node a is a source for four messages and nodes b, c and d are the sinks,

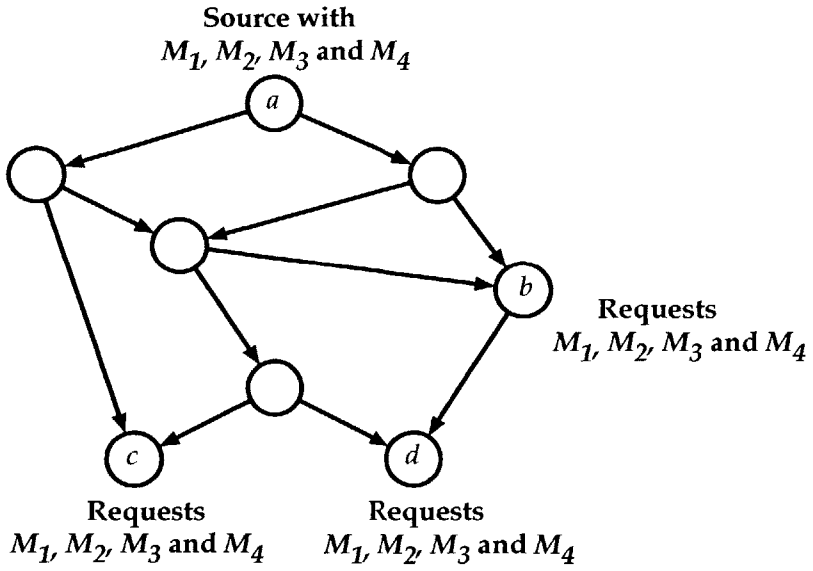


Figure 1-6: This is an example of the multicast problem. Node a is the source for all four messages. Nodes b, c and d are the sinks, which request all the messages.

which request all messages. Since all messages have the same source and same set of sinks, one can rephrase the feasibility of a multicast problem as an optimization problem in which the source is trying to send as many messages as possible to all sinks. This is important because rephrasing the multicast problem in these terms means that one can consider the rate of a multicast solution within the model presented in this chapter. However, in general when commodities have different sets of sources and different sets of sinks, the model needs to be adapted to consider the rate of a solution.

In a small generalization of the multicast problem, each sink still requests all messages but there is no restriction on the sources. This generalization is interesting because essentially all results that hold for the multicast problem also hold for this generalization.

1.3.2 k-Pairs Communication Problem

Another important problem in network communication is supporting k different point-to-point connections. In an instance of the k-pairs communication problem, there are k com-

modities, each with a single source and single sink for each commodity. In the example in

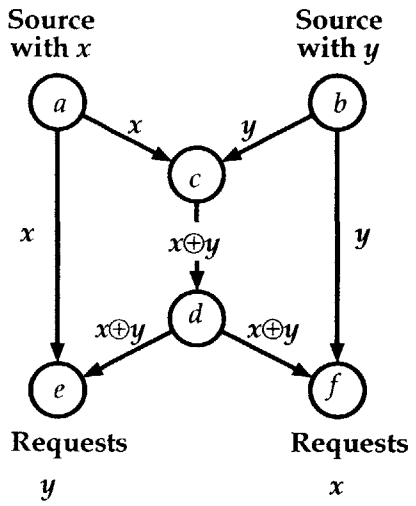


Figure 1-7: This is an example of the k-pairs communication problem. For message x, node a is the source and node f is the sink. For message y, node b is the source and node e is the sink. A solution over the alphabet $\{0, 1\}$ is given in the labels next to the edges.

Figure 1-7, there are two commodities. For the one commodity, node a is the source and node f is the sink. For the other, node b is the source and node e is the sink.

The k-pairs communication problem is interesting because it is closely related to the classical multicommodity flow problem. On an abstract level, instances of the two problems are defined in exactly the same way. However, commodities in a multicommodity flow problem can only "flow". In contrast, commodities in the k-pairs communication problem are information and thus can be copied and encoded. A fundamental question is the extent to which network coding is beneficial for the k-pairs communication problem. This question is the focus of Chapters 4 and 6.

1.4 Characteristics of a Solution

In this section we discuss some characteristics of a network coding solution of particular importance in this thesis: the size of the alphabet and the types of edge functions.

1.4.1 Alphabet Size

The first network coding problem we considered in Figure 1-1 has a solution over a binary alphabet. But sometimes a larger alphabet is required. For example, Figure 1-8 shows an

instance and a solution using the finite field $\mathbb{F}_3$ as the alphabet. However, there is no solution with an alphabet of size 2. Roughly, every pair of out-edges from the source needs to carry enough information to determine both messages. Over a binary alphabet there are only a small number of different choices of edge functions. Therefore, using a binary alphabet, some pair of out-edges from the source is forced to transmit redundant information and the corresponding sink is unable to determine both messages. We generalize this idea and provide a formal proof in Chapter 2.

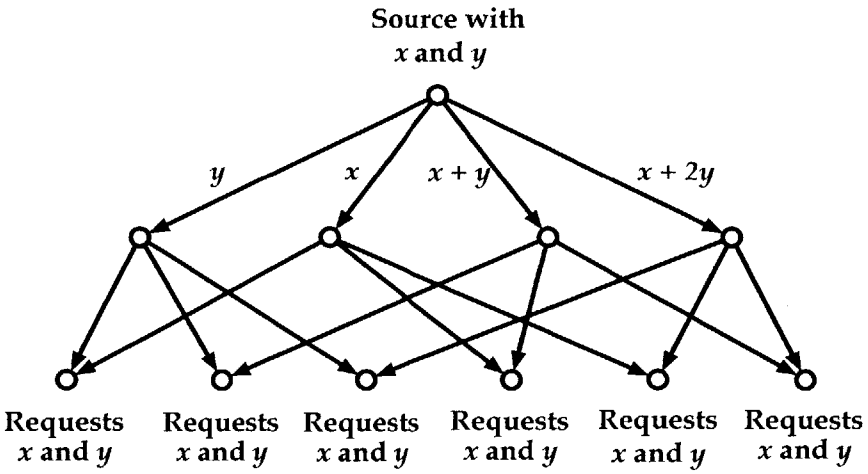


Figure 1-8: An instance in which there is a solution over $\mathbb{F}_3$ but not over a binary alphabet. The top node is a source for x and y. For each pair of nodes in the middle layer there is a sink in the bottom layer requesting both messages.

A large alphabet is undesirable for three reasons:

Latency. In practice, transmitting a symbol from a large alphabet takes longer than transmitting a symbol from a smaller alphabet. If a network coding solution uses a large alphabet, then a sink may need to wait a long time before it even begins decoding a message. Therefore, the network latency could be large.

Memory. Second, if nodes within the network need to compute functions over a large alphabet, then they may need a significant amount of storage even to remember the last alphabet symbol received on each edge.

Function Complexity. A small alphabet limits the complexity of edge functions. With a large alphabet, we lose this guarantee.

Fault Tolerance. A network failure during the transmission of a symbol from a large alphabet would cause a significant loss of data.

Although none of these issues is critical for the alphabet sizes we've seen so far (2 or 3), some instances require an extremely large alphabet (doubly exponential in the size of the network and number of commodities). We discuss this further in Chapter 2.

1.4.2 Linearity of a solution

How complicated do edge functions need to be? Ideally, a network code would use functions that are easy to represent and implement. Three restricted classes of codes using such simple functions have received special attention in the literature.

- A network code is *trivial* if edges transmit only unencoded messages. Traditional routing solutions are trivial codes.
- A network code in *linear* if the alphabet is a finite field and edges transmit only linear combinations of the messages. The code in Figure 1-1 is linear over $\mathbb{F}_2$ and the code in Figure 1-8 is linear over $\mathbb{F}_3$.
- A network code is *vector linear* if the alphabet is a finite vector space over a finite field. Furthermore, every component of every transmitted vector is a linear combination of the components of the message vectors. Every linear code is vector linear.

1.5 History of Network Coding

Network coding is a relatively new field that has strong connections to some classical problems in information theory and graph theory. In this section we describe the history of network coding, beginning with a review of these connections. We then describe the most influential early research in network coding. At the end of each chapter, we discuss the previous work related to its topic.

1.5.1 Prehistory of Network Coding

The field of network coding has roots in information theory as well as connections to two classic graph problems: Steiner tree packing and multicommodity flow.

Information Theory

Information theorists have concentrated on communication over a single (possibly noisy) channel. In the closest approach to network coding, a set of senders wishes to transmit information across a channel to a set of receivers. (See [5].) This work has led to a thorough understanding of data compression and error-tolerance.

However, a network cannot be accurately modeled as a single channel. The physical structure of a network introduces tremendous complexities not present in the single channel case. For example, in a network communication problem, different receivers may have access to different channels in a network. Characterizing the capacity of an information network has remained an open problem for decades.

Steiner Tree Packing

In a traditional view of communication in a network (predating network coding) data can be replicated at nodes but not encoded together with other data. The problem of packing fractional Steiner trees in a graph can be used to model this type of communication.

We begin by defining a Steiner tree and then discuss the fractional Steiner tree packing problem. Given a directed graph G, a specified node r and a subset S of the vertices, a *Steiner tree* is a tree rooted at r containing S. A fractional Steiner tree is a Steiner tree along with a weight between 0 and 1. Given a directed capacitated graph G, a node r and a subset of vertices S, a *fractional Steiner tree packing* is a set of fractional Steiner trees such that for every edge in G the total weight of trees containing that edge is no more than the capacity of the edge. The objective of the fractional Steiner tree packing problem is to maximize the total weight of the set of Steiner trees.

The problem of multicasting from a source to a set of sinks has traditionally been studied as a fractional Steiner tree packing problem. Suppose a node r needs to transmit data to a set of nodes S. Each tree in a fractional Steiner packing can be used to send a fraction of the data, given by its weight.

Generally, optimal fractional Steiner packing in undirected graphs is hard. Jain, Mahdian and Salavatipour [16] showed that the problem in undirected graphs is APX-hard [16]. This implies that the optimal value cannot be found efficiently unless P = NP. However, they gave a polynomial time algorithm to find a 1.598-approximation.

Multicommodity Flow

The communication problem in which there is a single source and single sink for each commodity has traditionally been viewed as a *multicommodity flow problem*. In this perspective data is modeled as a fluid: the amount of data leaving a node must exactly equal the amount entering except at sources and sinks.

In an instance of the multicommodity flow problem, there is a directed capacitated graph G and k commodities. For each commodity i, there is a single source $s(i)$, a single sink $t(i)$, and a demand d_i. In a multicommodity flow, the total flow of all commodities across an edge in G must be no more than the capacity of that edge. The objective is to find the largest fraction r such that for every commodity i, at least rd_i units of commodity i flow from source $s(i)$ to sink $t(i)$. This problem can be solved optimally using linear programming [34]. For further details see [34, 35, 23, 4].

The multicommodity flow problem is most closely related to the k-pairs communication problem. Understanding the relationship between these two problems is the focus of Chapters 4 and 6.

1.5.2 The Dawn of Network Coding

The field of network coding is only five years old; it was introduced by Ahlswede et al. [1] in 2000. Their work focused on the multicast problem and proved that a source can multicast k messages to a set of sinks provided the min-cut between the source and each sink has capacity k. This was the first major progress in understanding the capacity of a communication network.

Li, Yeung and Cai showed that the maximum achievable rate for the multicast problem can always be achieved using a linear code [24]. This focused attention on linear codes and in particular raised the question of whether they can be used to solve a wider array of network coding problems. Their proof of the existence of a linear solution can be viewed as the first deterministic algorithm for network coding. However, its running time is exponential in the size of the network.

Koetter and Médard devised an algebraic framework for network coding [19]. This reduced the problem of finding a linear solution for a general network coding problem to finding a non-zero point for a multivariate polynomial. Using their algebraic framework, Koetter and Médard where able to extend the study of linear network coding to directed graphs with cycles. For the special case of multicasting, Ho et al. [14] used the additional structure of the problem to construct an efficient randomized algorithm. Their algorithm has the nice property that it can be implemented in a distributed fashion without coordination between nodes in the network.

Jaggi et al. devised a polynomial-time implementation of the Li, Yeung and Cai multicast algorithm [15]. We refer to this procedure as the Jaggi-Li algorithm. It always finds a linear solution over a field $\mathbb{F}$ such that $|\mathbb{F}| = O(\#sinks)$. This raised the question of whether an even smaller alphabet would suffice. A second important contribution of Jaggi et al. was an instance on n nodes in which network coding achieves a rate $\Omega(\log n)$ times larger than the best rate achievable with fractional Steiner tree packing [15].

Up to this point, all research in network coding assumed the network was directed. In 2004, Li et al. [27] considered network coding in undirected graphs. They showed that fractional Steiner tree packing techniques can achieve at least $1/2$ the maximum possible multicast rate. This put an upper bound on the usefulness of network coding in undirected graphs for the multicast problem. Furthermore, it showed a drastic difference between network coding in the directed and undirected cases. It has been conjectured that for the k-pairs communication problem in an undirected graph the optimal rate can be achieved without network coding [27, 25, 11]. This open question is one of the motivations for the work in Chapter 6.

The applications of network coding now extend well beyond maximizing capacity to areas such as error-correction, wire-tapping protection and detection, and distributed network management. The Network Coding Home Page [18] provides a bibliography for the topics not covered in this thesis. Ho's dissertation [13] is also an excellent reference for many of these topics.

1.6 Our Results

We now outline the contributions of this thesis to the area of network coding chapter by chapter. Network coding is a young field with many exciting open questions. We discuss these at the end of each chapter.

1.6.1 Alphabet Size

An important measure of a network code is the size of the alphabet; a smaller alphabet is better. In Chapter 2 we show:

- Determining the smallest alphabet size for which there is a solution is NP-hard. This work initiated the study of complexity questions related to network coding.
- There are multicast instances requiring an alphabet of size $\Omega\left(\sqrt{\#sinks}\right)$. (Recently, this was shown to be tight by Fragouli and Soljanin [8].)
- For the general network coding problem, we show a periodicity effect. There are instances which only admit a solution if the alphabet size is a perfect k^{th} power.
- Building on this, we show a doubly-exponential lower bound on alphabet size for certain instances.
- For these instances, slightly increasing the capacity of a single edge exponentially reduces the required alphabet size. This is a motivation for refining the model in Section 1.1 to consider the rate of a solution. We do this in Chapter 4.

The results in this chapter appeared in [21, 22].

1.6.2 Complexity of Linear Network Coding

Positive results for the multicast problem suggested that solvable instances might always admit a linear solution. Furthermore, there was hope that the multicast algorithms could be extended to general network coding problems.

In Chapter 3 we classify network coding problems based on the configuration of sources and sinks. We show that

- For some classes of problems, every solvable instance admits a trivial solution, which can be found efficiently using traditional flow techniques.

- The multicast problem and a generalization with multiple sources are the only restricted network coding problems for which network coding is necessary and linear codes suffice. The known multicast algorithms can be extended to solve instances of the generalization [15, 10, 14].

- We show that it is NP-hard to determine if a linear solution exists if sinks are allowed to request different subsets of the messages.

- As a corollary, we obtain an instance of the general network coding problem that admits a vector linear solution but not a linear solution.

The work in Chapter 3 appeared in [21].

1.6.3 The Capacity of an Information Network

In Chapter 4 we define a model for network coding in a directed acyclic graph in which the rate of the solution is a parameter. Our goal is to find necessary and sufficient conditions for the existence of a solution of rate r for an instance of the general network coding problem. This remains an open question; however, we make the following advances:

- We define the sparsity of an edge-cut, a natural condition to consider. We show this is neither necessary nor sufficient.

- We then define an improved cut condition called *meagerness*. We show that the capacity of the most meager edge-cut is an upper bound on the achievable rate.

- We present an example in which the maximum achievable rate is $2/3$, but the most meager cut has capacity 1. This shows that meagerness is a necessary but not sufficient condition.

- The proof of this gap leads to a tighter entropy-based upper bound on the maximum achievable rate. (This condition was independently discovered by Song, Yeung and Cai [36] and is discussed in [39] as well.)

Preliminary results appeared in [11]. The work in this chapter is joint with Nicholas Harvey and Robert Kleinberg.

1.6.4 Models for Graphs with Cycles

Our objective in Chapter 5 is to extend the techniques from the preceding chapter to graphs with cycles and to undirected graphs. We consider three possible models for network coding in directed graphs (which may have cycles) and prove their equivalence. Unlike the work of Koetter and Médard [19], we do not restrict our attention to linear coding solution.

We model an undirected graph, roughly, with a directed graph by replacing each undirected edge with two oppositely-directed edges. Under this model, we let a network code split the capacity of an undirected edge in any possible way between the two oppositely-directed edges.

With these models in hand, we then extend our entropy-based upper bounds on the maximum rate of a network coding solution to this new setting. It is important to note that undirected graphs present considerable challenges beyond directed graphs. A contribution of this chapter is the introduction of a framework for studying network coding in undirected graphs.

The work in this chapter is joint with Nicholas Harvey, Robert Kleinberg and Eric Lehman.

1.6.5 The k-Pairs Communication Problem in Undirected Graphs

Recall that in the k-pairs communication problem there are k commodities, each with a single source and single sink. This problem is closely related to the multicommodity flow problem and understanding this relationship in undirected graphs is the focus of Chapter 6.

The target of this investigation is an open conjecture: for any instance in an undirected graph, the maximum rate of a network coding solution can be achieved using multicommodity flow techniques. In undirected graphs, the value of the sparsest cut is an upper bound on the maximum achievable rate with and without network coding. However, it is well known that the maximum multicommodity flow in a network can be much smaller than the value of the sparsest cut [32, 23]. In a limited sense, we extend this result to network coding. In particular, we prove the first gap between the maximum rate of a network coding solution and the value of the sparsest cut. In fact, we show such a gap exists for an infinite class of instances using a new entropy-based necessary condition for the existence of a network coding solution. For all of these instances, we prove the conjecture is true; namely, the maximum rate of a network coding solution can be achieved using multicommodity flow techniques.

The work in this chapter is joint with Nicholas Harvey and Robert Kleinberg.

Chapter 2

Alphabet Size

In this chapter we consider the size q of the alphabet Σ. There are a number of reasons why alphabet size is interesting. First, a small alphabet implies that the edge functions are simple and efficiently computable. Second, the alphabet size is related to the network latency. In a network implementing a network code each symbol from the alphabet for the code is mapped to a sequence of symbols from the alphabet the network is using. The encoding of a symbol from a large alphabet is larger and therefore takes longer to pass through a network than the encoding of a symbol from a small alphabet. Third, a naive implementation of a network code requires nodes to store alphabet symbols in order to compute edge functions. Thus a larger alphabet might require more storage at nodes. Finally, there is a connection between understanding the alphabet size and understanding the computational complexity of network coding. Currently, the general network coding problem is not known to be decidable. An upper bound on the size of the alphabet admitting a solution would imply decidability.

In this chapter we present two lower bounds on the alphabet size required for there to exist a solution. The first lower bound is for the multicast problem. We show that the size of the alphabet for some instances must be $\Omega(\sqrt{m})$ where m is the number of sinks. We then prove that determining the minimum alphabet size that admits a solution for an instance of the multicast problem is NP-hard. This hardness result initiated the study of complexity issues related to network coding.

For the general network coding problem we demonstrate a periodicity effect. Specifically, we show instances of the network coding problem with $2k$ messages and $O(k^2)$ nodes for which there only exists a solution if the alphabet size is a perfect k^{th} power. Combining these instances we create an instance of the network coding problem with k messages that only admits a solution if the alphabet size is doubly-exponential in k. Lastly, we consider solutions that use slightly larger alphabets for the edges than for the sources. We show that these solutions avoid the periodicity effect for the previously considered instances. This introduces the idea of solutions which operate at rates other than 1. In Chapter 4 we consider

this concept in far greater depth.

2.1 Lower Bound for Multicast

We show that there exist solvable multicast problems requiring an alphabet of size $\Omega(\sqrt{m})$ where m is the number of sinks. This has recently been shown to be a tight bound [8].

We begin with a technical lemma that gives some insight into why a larger alphabet may be necessary to solve a network coding problem. Let f_i and f_j be functions mapping Σ^2 to Σ. Then we can form a function $g_{ij} : \Sigma^2 \rightarrow \Sigma^2$ defined by:

$$g_{ij}(\alpha, \beta) \;=\; (\, f_i(\alpha, \beta)\,,\; f_j(\alpha, \beta)\,)$$

If g_{ij} is invertible, then we say that functions f_i and f_j are *independent*. Equivalently, f_i and f_j are independent if and only if there do not exist distinct points (α_1, β_1) and (α_2, β_2) in Σ^2 such that $f_i(\alpha_1, \beta_1) = f_i(\alpha_2, \beta_2)$ and $f_j(\alpha_1, \beta_1) = f_j(\alpha_2, \beta_2)$.

In short, the messages α and β can be determined from $f_i(\alpha, \beta)$ and $f_j(\alpha, \beta)$ if and only if the functions f_i and f_j are independent. But the following lemma says we can not construct a large set of pairwise independent functions over a small alphabet. The main idea of our subsequent theorems is that some information flow problems are not solvable with a small alphabet because one "runs out" of independent functions.

Lemma 1 *If $f_1, \ldots, f_n$ are pairwise independent functions of the form $f_i : \Sigma^2 \rightarrow \Sigma$, then $n \leq q + 1$.*

Proof. First, we show that each function f_i must be a q-to-1 mapping. Suppose not. Then f_i must take on some value $\gamma \in \Sigma$ at more than q points $(\alpha, \beta) \in \Sigma^2$. By the pigeonhole principle, the function f_j (where $j \neq i$) must take on some value $\delta \in \Sigma$ for at least two of those points; call them (α_1, β_1) and (α_2, β_2). Thus, we have $f_i(\alpha_1, \beta_1) = f_i(\alpha_2, \beta_2)$ and $f_j(\alpha_1, \beta_1) = f_j(\alpha_2, \beta_2)$, contradicting the assumption that f_i and f_j are independent.

Now define an *agreement* of the function f_i to be a pair of distinct points (α_1, β_1) and (α_2, β_2) in Σ^2 such that $f_i(\alpha_1, \beta_1) = f_i(\alpha_2, \beta_2)$. Each function f_i has $q^2(q-1)/2$ agreements; for each of the q elements $\gamma \in \Sigma$, we can choose $q(q-1)/2$ pairs of distinct points from among the q points in $(\alpha, \beta) \in \Sigma^2$ such that $f_i(\alpha, \beta) = \gamma$. In all, there are $q^2(q^2-1)/2$ pairs of distinct points in Σ^2. Therefore, again by the pigeonhole principle, there must exist two different functions f_i and f_j that share an agreement if:

$$\begin{aligned} n \cdot q^2(q-1)/2 &> q^2(q^2-1)/2 \\ n &> q+1 \end{aligned}$$

But if f_i and f_j share an agreement (α_1, β_1) and (α_2, β_2), then we have $f_i(\alpha_1, \beta_1) = f_i(\alpha_2, \beta_2)$ and $f_j(\alpha_1, \beta_1) = f_j(\alpha_2, \beta_2)$, contradicting the assumption that f_i and f_j are independent. Therefore, we must have $n \leq q + 1$ as claimed. □

Conversely, constructing a set of $q+1$ pairwise independent functions is a simple matter when q is a prime power. Regard Σ as a finite field, and take all functions $f(x, y)$ of the form $x + \alpha y$ where $\alpha \in \Sigma$ together with the function y.

Theorem 2 *There exist solvable instances of the multicast network coding problem that require an alphabet of size* $\Omega(\sqrt{m})$.

Proof. Consider an instance of the multicast network coding problem defined as follows. There is a single source s and intermediate nodes $v_1, \ldots, v_p$. There is a directed edge (s, v_i) from the source to each intermediate node. For each pair of distinct intermediate nodes v_i and v_j, create a sink t_{ij} and add directed edges (v_i, t_{ij}) and (v_j, t_{ij}). Note that the number of intermediate nodes p is $\Theta(\sqrt{m})$, where m is the number of sinks. There are two messages, M_1 and M_2 available at the source, and these two messages are demanded by every sink.

First, we show that this problem is solvable. Let M_1 and M_2 represent the two messages which must be multicast from the source to all sinks. If the alphabet size is a prime power greater than the number of intermediate nodes, then the edges $(s, v_1), \ldots, (s, v_p)$ may carry functions of M_1 and M_2 that are pairwise independent. If each intermediate node then relays its input directly to its outputs, then each sink receives pairwise independent functions of M_1 and M_2 and can therefore reconstruct messages M_1 and M_2 as desired.

Now, we show that the problem is not solvable if the alphabet is too small. Suppose that the number of intermediate nodes is greater than $q + 1$. Then by Lemma 1 the edges (s, v_i) can not carry functions of M_1 and M_2 that are all pairwise independent. In particular, suppose that intermediate nodes v_i and v_j receive functions that are not independent. Then messages M_1 and M_2 can not be determined from the values of these functions. Therefore, these messages can not be computed at sink t_{ij}, which receives no other information. Thus, in general, an alphabet of size $\Omega(\sqrt{m})$ is required to solve some instances of the multicast network coding problem. □

2.2 Hardness for Multicast

We have seen that some multicast network coding problems can only be solved by using a large alphabet. In this section, we show that it is computationally hard to determine exactly how large an alphabet is required.

The following two reductions rely on the hardness of graph coloring and a generalization of graph coloring called H-coloring. In both cases, we map an undirected graph $G' = (V', E')$ to an instance of the multicast network coding problem on a graph G as follows. The nodes of G consist of a single source s, an intermediate node v_i for each

vertex $v_i' \in V'$, and a sink t_{ij} for each edge $\{v_i', v_j'\} \in E'$. There is an edge $(s, v_i) \in E$ for each vertex $v_i' \in V'$, and there are edges $(v_i, t_{ij}) \in E$ and $(v_j, t_{ij}) \in E$ for each edge $\{v_i', v_j'\} \in E'$. Two messages, M_1 and M_2 are available at the source, and these two messages are demanded by every sink.

Theorem 3 *Deciding whether there exists a linear network code with alphabet size q for a multicast network coding instance is NP-hard when q is a prime power.*

Proof. We use a reduction from vertex coloring on undirected graphs.

Let $G' = (V', E')$ be an undirected graph. Construct an instance of the network coding problem defined on a graph G as described above.

We show that G' is $q + 1$ colorable if and only if the instance is solved by a linear network code with an alphabet of size q. First, suppose that G' is $(q+1)$-colorable in order to show that this implies the existence of a linear network code that solves the instance of the network coding problem. Let $c(i) \in \{1, \ldots, q+1\}$ denote the color of vertex v_i'. As noted after the proof of Lemma 1, there exist pairwise independent functions $f_1, \ldots, f_{q+1}$ of the form $f_i : \Sigma^2 \to \Sigma$. Along each edge (s, v_i) and all edges (v_i, t_{ij}), send the symbol $f_{c_i}(M_1, M_2)$. Then each sink t_{ij} receives $f_{c(i)}(M_1, M_2)$ and $f_{c(j)}(M_1, M_2)$. Since colors on adjacent vertices are distinct, $c(i) \neq c(j)$, and so the functions $f_{c(i)}$ and $f_{c(j)}$ are independent. Thus, each sink can reconstruct messages M_1 and M_2 as desired.

Next, suppose that there exists a linear solution to the network coding instance with an alphabet of size q. We show that this implies that there exists a $q + 1$ coloring of G'. Each edge (s, v_i) then carries a nonzero linear combination $\alpha M_1 + \beta M_2$. We can partition the set of all such linear combinations into $q + 1$ equivalence classes; the nonzero multiples of $M_1 + \alpha M_2$ form one class for each $\alpha \in \Sigma$ and the nonzero multiples of M_2 form the remaining class. This places every pair of independent linear combinations into different classes. Assign each class a distinct color. Now assign vertex $v_i' \in V'$ the color of the class containing the function associated with edge (s, v_i). The endpoints of each edge $\{v_i', v_j'\} \in E'$ are then colored differently, because the functions for edges (s, v_i) and (s, v_j) must be independent so that sink t_{ij} can reconstruct messages M_1 and M_2. Therefore, this gives a valid $q + 1$ coloring of G'. □

2.2.1 General Codes

We now consider general codes and show that minimizing the alphabet size remains hard. We use a reduction from H-coloring. An H-coloring of an undirected graph G is a homomorphism $h : G \to H$ such that $h(v)$ and $h(u)$ are adjacent vertices of H if v and u are adjacent vertices of G. Hell and Nešetřil showed that H-coloring is NP-hard whenever H is not bipartite and is solvable in polynomial time if H is bipartite[12].

Theorem 4 *Deciding if there exists a network code with alphabet size q for a instance of the multicast network coding problem is NP-hard when q is a prime power.*

Proof. Define a graph H as follows. The vertices of H are all the functions $f : \Sigma^2 \to \Sigma$. There is an edge between vertices f and g if they are independent functions. Note that H is not bipartite for all prime powers q, since there exists a set of three pairwise independent functions.

Let $G' = (V', E')$ be an arbitrary undirected graph. Construct an instance of the multicast network coding problem on G as described above. We show that G' is H-colorable if and only if there exists a solution to the network coding instance over an alphabet of size q.

Suppose that G' has an H-coloring, h. Then we can solve the instance of the network coding problem by sending each vertex v_i the symbol $f(M_1, M_2)$, where $f = h(v_i')$. Each sink receives inputs from two vertices v_i and v_j such that v_i' and v_j' are adjacent in G'. This means that $h(v_i')$ and $h(v_j')$ are adjacent in H and are therefore independent functions. Thus, the sink can reconstruct messages M_1 and M_2.

On the other hand, suppose that there is a solution to the network coding instance. Then we can construct an H-coloring h of the graph G' as follows. For each vertex v_i in G, let $h(v_i')$ be the function of M_1 and M_2 transmitted on the out-edge of v_i. If vertices v_i' and v_j' are adjacent in G', then the corresponding vertices v_i and v_j in G share a sink. Since the sink can reconstruct M_1 and M_2, the functions $h(v_i')$ and $h(v_j')$ must be independent and thus adjacent in H. Therefore, h is a valid H-coloring of G'. □

Interestingly, some multicast problems can be solved with a smaller alphabet by using *nonlinear* codes. For example, let Σ be an alphabet of size 3. Let G' be a graph whose vertices are all functions $f : \Sigma^2 \to \Sigma$ and whose edges are all pairs of independent functions. Then consider the instance of the network coding problem on a graph G derived as before. This problem can be solved using alphabet Σ by sending $f(M_1, M_2)$ to the vertex corresponding to the function f. On the other hand, suppose that we want a linear solution. This requires coloring the vertices of G' using independent linear functions. A computation shows that the chromatic number of G' is 6, which implies an alphabet of size at least 5. We conjecture that there can be a very large gap between the absolute smallest alphabet size for a multicast problem and the smallest possible alphabet size using linear coding.

2.3 Lower Bound for General Problem

We now turn our attention to the general network coding problem. In this section, we show a *periodicity effect*: for every integer $k \geq 2$, there exists an instance of the network coding problem that admits a solution if and only if the alphabet size is a perfect k^{th} power. Building on this result, we construct an instance with $O(k)$ messages and $O(k)$ nodes that admits a solution if and only if the alphabet size is an enormous $2^{\exp(\Omega(k^{1/3}))}$. In other words, if we regard each message as a length-d vector over the binary field, then d must be *exponential* in the size of the network. For this same instance, we show that if edge capacities are slightly larger than the message size, then there is a solution with a modest

alphabet size of $O(2^k)$. In light of these results, we suggest that a more appropriate model would assume that the network operates at slightly under capacity.

We present the following results in this section:

- The power of network codes does not strictly increase with alphabet size, but rather increases as the size of the set of perfect roots of the alphabet size increases. Thus, an alphabet size of 2^6, which is a perfect square, a perfect cube, and a 6^{th} power is strictly better than a size of 2^2or 2^3, but an alphabet of size 2^7, which is only a 7^{th} power is not. In practice, one might be tempted to use an alphabet size of 2^{32} or 2^{64} so that a single alphabet symbol fits into a machine word. However, our construction suggests that these would actually be poor choices, since 32 and 64 have so few divisors.

- When linear coding is used to solve the multicast problem, an alphabet of size $O(m)$ suffices if there are m sinks. The situation with the general network coding problem is dramatically different. By placing many of our constructions in parallel, we obtain an instance of the general network coding problem with $O(k)$ nodes, including sinks, that requires an alphabet of size $2^{\exp(\Omega(k^{1/3}))}$. Thus, there exist instances of the network coding problem that are solvable, but require extremely large alphabets. Naively, even describing the solution takes space exponential in the problem size.

- We show that our lower bound on the alphabet size does not hold if we slightly increase the capacity of the edges. In particular, the instance described above admits a vector-linear solution where messages are vectors of length q provided each edge can transmit a vector of length $n\left(1+\frac{1}{q^{1/3}}\right)$.

In light of these results, we suggest that a better model for the study of network coding problems would allow the network to operate at slightly under capacity, since this may avoid an exponential blowup in the solution complexity. In Chapter 4 we consider a more general definition of a network coding solution which allows each edge and message to use a different alphabet.

2.3.1 Construction

First we describe the construction of an instance of the network coding problem that we denote I_k. We then prove that instance I_k admits a solution if and only if the alphabet size is a k^{th} power. The construction is shown for $k = 3$ in Figure 2-1.

There is a single source with $2k$ messages $M_1, M_2 \ldots M_{2k}$ and a single middle-layer node. There is an edge C of capacity 2 from the source to the middle layer node. There are $O(k^2)$ sinks. There is an edge of capacity 2 from the middle-layer node to each sink. One sink t^* requests all $2k$ messages and all other sinks request k messages. Let $P =$

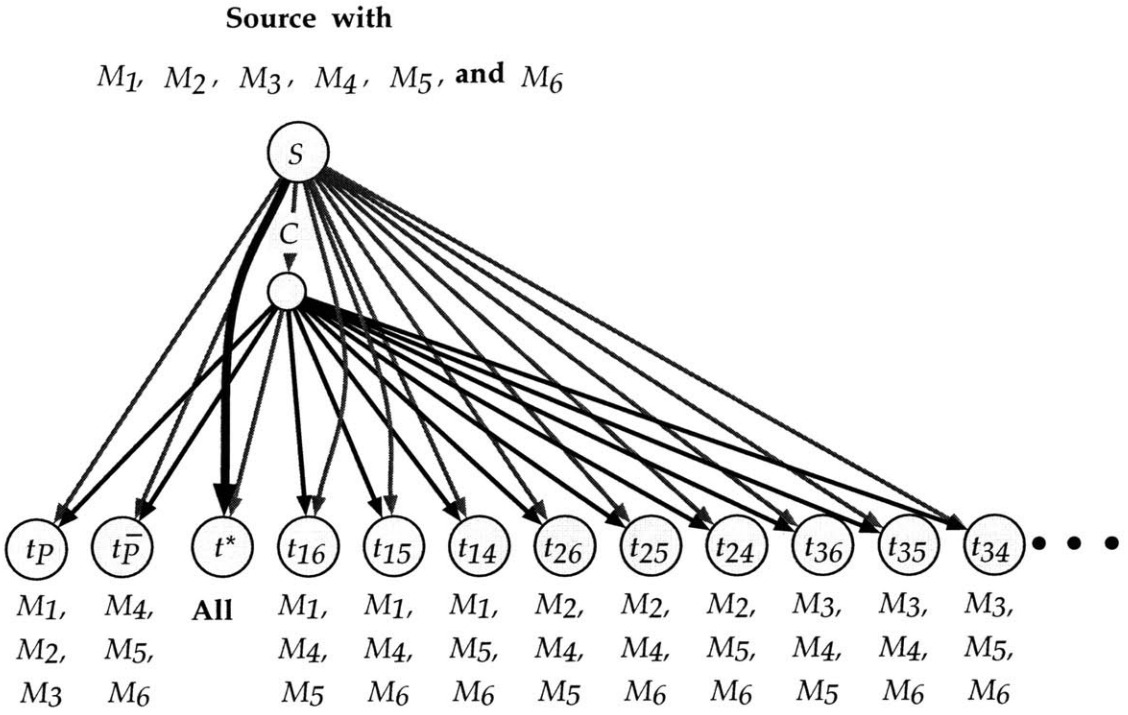


Figure 2-1: In this instance of the network coding problem, all edges are directed downward. All edges have capacity 2, except for the thick, curved edge, which has capacity 4. There are six messages, $M_1, M_2, \ldots, M_6$. The top node is the only source and has every message. The bottom layer of nodes is the sinks, whose requests are listed below each node. The complementary sinks $t_{\overline{ij}}$ are not shown.

$\{M_1, M_2 \ldots M_k\}$ and $\overline{P} = \{M_{k+1}, M_{k+2} \ldots M_{2k}\}$. The sink t_P requests all messages in P, and the sink $t_{\overline{P}}$ requests all messages in $\overline{P}$. For all i and j such that $1 \leq i, j \leq k$ there is a sink t_{ij} that requests the k messages $(\overline{P} \cup \{M_i\}) - \{M_{k+j}\}$ and a complementary sink $t_{\overline{ij}}$ that requests the k messages $(P \cup \{M_{k+j}\}) - \{M_i\}$. From the source to t^* there is an edge of capacity $2k-2$, and from the source to every other sink there is an edge of capacity $k-1$.

2.3.2 Analysis

The analysis of the network I_k relies on understanding the function $F : \Sigma^{2k} \rightarrow \Sigma^2$ that describes the two symbols transmitted over edge C. We first prove a preliminary fact about the function F.

Lemma 5 *The function $F : \Sigma^{2k} \rightarrow \Sigma^2$, which determines the symbol sent over edge C, is q^{2k-2}-to-1.*

Proof. Sink t^* must recover all the messages from the symbol sent on edge C and the $2k-2$ other alphabet symbols it receives on the direct edge from the source. Let $g : \Sigma^{2k} \rightarrow \Sigma^{2k-2}$ be the function that determines the $2k-2$ symbols sent along the direct edge. If F is not q^{2k-2}-to-1, then F must map more than q^{2k-2} points in Σ^{2k} to some pair of symbols in Σ^2. Then g necessarily maps two of these points to the same value in Σ^{2k-2}. Thus, sink t^* receives identical symbols for two different sets of sent messages and can not distinguish them. □

We now consider restrictions placed on the function F by a sink requesting a set of k messages. In particular, we consider the set of assignments to the $2k$ messages that are mapped by F to the same value. For the remainder of this section we let $\beta \in \Sigma^2$ be a fixed value sent by F down edge C and $B \subseteq \Sigma^{2k}$ denote the subset of size q^{2k-2} that F maps to β. Thus, B is the set of assignments to messages such that the edge C carries the value β. Consider a subset of the $2k$ messages $A = \{M_{s_1}, M_{s_2} \ldots M_{s_r}\}$. The set of assignments to the messages in A such that there exists an assignment to the remaining messages on which F takes on the value β is the projection of each point in B onto the coordinates $s_1, s_2 \ldots s_r$. We denote this set as $\pi_{s_1,s_2 \ldots s_r}(B)$ or equivalently $\pi_A(B)$. For example, if $A = \{M_1, M_2, M_3\}$

$$\begin{aligned} \pi_A(B) &= \{(x_1, x_2, x_3) \mid (x_1, x_2, x_3, y_4, \ldots y_{2k}) \in B \\ &\qquad \text{for some } (y_4, y_5 \ldots y_{2k})\} \end{aligned}$$

Lemma 6 *Let t be a sink requesting the set A of k messages. Then $|\pi_A(B)| = q^{k-1}$.*

Proof. In addition to the point in Σ^2 sent to the middle-layer node, the sink t also receives $k-1$ symbols on a direct edge from the source. If the sink t receives the value $\beta \in \Sigma^2$ from the middle-layer node, then the assignment to the k messages in A must be according to one of the points in $\pi_A(B)$. Each point in $\pi_A(B)$ represents a different assignment to the messages in A. Therefore, the sink t must receive a different set of $k-1$ symbols along the direct edge from the source for each of the points in $\pi_A(B)$. Since there are only q^{k-1} different assignments to the $k-1$ symbols sent down the direct edge from the source, we must have $|\pi_A(B)| \leq q^{k-1}$. Otherwise, the messages in A can not be uniquely determined from the information received at the sink.

By construction, there is a complementary sink $\bar{t}$ requesting the subset $\overline{A}$ consisting of the other k messages. By the same argument as above, $|\pi_{\overline{A}}(B)| \leq q^{k-1}$. The number of different points in Σ^{2k} on which F takes on the value β is at most $|\pi_A(B)| \cdot |\pi_{\overline{A}}(B)|$. By Lemma 5, F is a q^{2k-2}-to-1 function. Therefore,

$$\begin{aligned} q^{2k-2} &\leq |\pi_A(B)| \cdot |\pi_{\overline{A}}(B)| \\ &\leq |\pi_A(B)| \cdot q^{k-1} \\ q^{k-1} &\leq |\pi_A(B)| \end{aligned}$$

Therefore, $|\pi_A(B)| = q^{k-1}$. □

We learn more about the structure of the set of points B on which F takes on the value β by applying the above lemma to sink t requesting the set A of k messages and its complementary sink $\overline{t}$ requesting the other k messages.

Lemma 7 *Let t be a sink requesting the set A of k messages, and let $\overline{t}$ be the sink requesting the other k messages $\overline{A}$. Then*

$$B = \pi_A(B) \times \pi_{\overline{A}}(B)$$

Proof. Consider an assignment $z \in B$ to the messages. Suppose z assigns the k messages in A according to an assignment z_A and assigns the messages in $\overline{A}$ according to an assignment $z_{\overline{A}}$. Then, $z_A \in \pi_A(B)$ and $z_{\overline{A}} \in \pi_{\overline{A}}(B)$ by the definition of projection. Therefore, $B \subseteq \pi_A(B) \times \pi_{\overline{A}}(B)$. By Lemma 6, $|\pi_A(B) \times \pi_{\overline{A}}(B)| = |\pi_A(B)| \cdot |\pi_{\overline{A}}(B)| = q^{2k-2}$. Since $|B| = q^{2k-2}$ by Lemma 5, B has the same size as the set containing it. Therefore, $B = \pi_A(B) \times \pi_{\overline{A}}(B)$. □

The next lemma shows that for at least one sink, the projection of the set B onto the messages requested by that sink is "large". The proof makes use of the discrete Loomis-Whitney inequality relating the size of a set to the product of the sizes of projections of the set [28, 3, 38, 37] . Roughly, the discrete Loomis-Whitney inequality generalizes the intuition that a massive statue must look big from the front, the side, or the top; that is, a big region must have some big projection.

Theorem 8 (Discrete Loomis-Whitney Inequality) *Let $Q \subseteq \Sigma^h$ and $r \leq h$,*

$$|Q| \leq \prod_{1 \leq s_1 < \ldots < s_r \leq h} |\pi_{s_1,\ldots,s_r}(Q)|^{hr^{-1}\binom{h}{r}^{-1}}$$

Lemma 9 *There exists a set of k messages $A = (\overline{P} \cup \{M_i\}) - \{M_{k+j}\}$ such that $|\pi_A(B)| \geq \left\lceil q^{\frac{k-1}{k}} \right\rceil \left\lceil q^{\frac{(k-1)^2}{k}} \right\rceil$.*

Proof. We prove this in three steps. First we show that there exists a message $M_i \in P$ such that $|\pi_i(B)| \geq \left\lceil q^{\frac{k-1}{k}} \right\rceil$. Then we show that there exists a set of $k-1$ messages,

$\overline{P} - \{M_{k+j}\} = \{M_{s_1}, M_{s_2} \ldots M_{s_{k-1}}\}$ such that $|\pi_{s_1,s_2 \ldots s_{k-1}}(B)| \geq \left\lceil q^{\frac{(k-1)^2}{k}} \right\rceil$. To finish the proof, we use Lemma 7 to show that

$$|\pi_{i,s_1,s_2 \ldots s_{k-1}}(B)| \geq |\pi_i(B)| \cdot |\pi_{s_1,s_2 \ldots s_{k-1}}(B)|$$

Using the Loomis-Whitney Inequality for $r = 1$,

$$\begin{aligned} \prod_{1 \leq i \leq k} |\pi_i(B)| &\geq |\pi_P(B)| \\ &= q^{k-1} \end{aligned}$$

Therefore, there exists at least one message $M_i \in P$ for which $|\pi_i(B)| \geq \left\lceil q^{\frac{k-1}{k}} \right\rceil$.

Similarly, by the Loomis-Whitney Inequality for $r = k - 1$,

$$\begin{aligned} \prod_{k+1 \leq s_1 < s_2 < \ldots < s_{k-1} \leq 2k} |\pi_{s_1,s_2 \ldots s_{k-1}}(B)|^{\frac{1}{k-1}} &\geq |\pi_{\overline{P}}(B)| \\ &= q^{k-1} \\ \prod_{k+1 \leq s_1 < s_2 < \ldots < s_{k-1} \leq 2k} |\pi_{s_1,s_2 \ldots s_{k-1}}(B)| &\geq q^{(k-1)^2} \end{aligned}$$

Since there are k terms in the product on the left, there exist a subset of $k-1$ messages $\{M_{s_1}, M_{s_2} \ldots M_{s_{k-1}}\} \subseteq \overline{P}$ such that $|\pi_{s_1,s_2 \ldots s_{k-1}}(B)| \geq \left\lceil q^{\frac{(k-1)^2}{k}} \right\rceil$.

For each $x_i \in \pi_i(B)$, there exists $x \in \pi_P(B)$ that assigns message M_i the value x_i. Similarly, for each $(y_{s_1}, \ldots y_{s_{k-1}}) \in \pi_{s_1,s_2 \ldots s_{k-1}}(B)$ there exists $y \in \pi_{\overline{P}}(B)$ that corresponds to assigning the messages $\{M_{s_1}, M_{s_2}, \ldots M_{s_{k-1}}\}$ the values $(y_{s_1}, y_{s_2} \ldots y_{s_{k-1}})$. By Lemma 7, $B = \pi_P(B) \times \pi_{\overline{P}}(B)$, and so there is a point in B that assigns the value x_i to message M_i and the values $y_{s_1}, y_{s_2}, \ldots, y_{s_{k-1}}$ to messages $M_{s_1}, M_{s_2}, \ldots M_{s_{k-1}}$. Therefore,

$$\begin{aligned} |\pi_{i,s_1,s_2 \ldots s_{k-1}}(B)| &\geq |\pi_i(B)| \cdot |\pi_{s_1,s_2 \ldots s_{k-1}}(B)| \\ &\geq \left\lceil q^{\frac{k-1}{k}} \right\rceil \left\lceil q^{\frac{(k-1)^2}{k}} \right\rceil \end{aligned}$$

□

Theorem 10 *There exists a solution to network I_k if and only if the alphabet size q is a perfect k^{th} power.*

Proof. There are two steps. We first show how to construct a solution with an alphabet of size $q = \ell^k$ for any $\ell \geq 2$. Then we show that the network only admits a solution if the alphabet size is a perfect k^{th} power.

Let Γ be a set of size ℓ. We regard each message as a length-k vector of symbols drawn from Γ. Recall that edge C has capacity 2. Therefore, we can send $2k$ symbols from Γ across edge C. We use these $2k$ symbols to transmit the first coordinate of each of the $2k$ messages. The sink t^*, which requests all $2k$ messages, must receive $2k$ length-k vectors. Via edge C, it receives the first coordinate of each of these $2k$ vectors. Along the direct edge of capacity $2k-2$ from the source to t^*, we send the remaining $k-1$ coordinates of each of the $2k$ messages. Now consider a sink t requesting a subset A of k messages. Sink t receives the first coordinate of each message in A from edge C. The remaining $k-1$ coordinates of each of the messages in A can be transmitted across the direct edge of capacity $k-1$ from the source to t. Thus, each sink receives every coordinate of the messages it requests. The alphabet size is $q = \ell^k$.

Next, we show that the alphabet size must be a k^{th} power. Lemma 9 says that there exists a set $A = (\overline{P} \cup \{M_i\}) - \{M_{k+j}\}$ of k messages with $|\pi_A(B)| \geq \left\lceil q^{\frac{k-1}{k}} \right\rceil \left\lceil q^{\frac{(k-1)^2}{k}} \right\rceil$. On the other hand, since there is a sink t_{ij} for every $1 \leq i, j \leq k$ requesting the set of messages $(\overline{P} \cup \{M_i\}) - \{M_{k+j}\}$, we must have $|\pi_A(B)| = q^{k-1}$ by Lemma 6. These two relationships can hold simultaneously only if q is a k^{th} power. □

2.3.3 A Lower Bound on Alphabet Size

We now construct an instance, J_n, of the network coding problem with $\Theta(n)$ nodes that admits a solution if and only if the alphabet size is $q = 2^{\exp(\Omega(n^{1/3}))}$. The construction is as follows. For each prime number $p \leq n^{1/3}$, we take the instance I_p of the preceding construction, which forces the alphabet size to be a p^{th} power. We place all of these constructions in parallel in order to create instance J_n.

Corollary 11 *Instance J_n with $\Theta(n)$ nodes admits a solution if only if the alphabet size is $2^{\exp(\Omega(n^{1/3}))}$.*

Proof. The number of nodes in J_n is at most:

$$\sum_{i=1}^{n^{1/3}} 2i^2 + 1 \quad = \quad \Theta(n)$$

Instance I_p, is solvable if and only if the alphabet size if a p^{th} power. Thus, instance J_n is solvable if and only if the alphabet size is a p^{th} power for every prime p less than $n^{1/3}$.

The product of primes less than x is $e^{(1+o(1))x}$ (see [9]). Therefore, the minimum alphabet size is $q = 2^{\exp(\Omega(n^{1/3}))}$. □

The fact that J_n is made up of a collection of disjoint networks is not critical to the proof. In fact, one can add some sinks that join the networks and force some degree of coding. More generally, one can imagine problems in which the various instances requiring different vector sizes are embedded in a larger network and may not be easily detectable.

While the instance J_n requires a very large alphabet, not much storage is actually needed at the nodes. Also, the solution presented in Section 2.3.2 can be described concisely without resorting to a particularly powerful description language. An interesting question is whether other instances admit only solutions with not only enormous alphabets, but also comparable storage requirements and description complexity.

2.3.4 Operating Below Capacity

We now consider the effect of allowing the network to operate at slightly below full capacity. We model this using vector linear codes in which the edges are allowed to transmit vectors that are longer than the message vectors. In particular, suppose that each message is a length-k vector, but vectors transmitted over edges have length $(1+\epsilon)k$. We show that for vanishingly small ϵ, the network J_n in Corollary 11 admits a solution over any field with a vector length linear in the size of the network. Using a constant-size field, this corresponds to a vector-linear solution with an alphabet that is only exponential (instead of doubly exponential) in the size of the network.

Theorem 12 *There exists a vector linear solution over the field $\mathbb{F}_2$ to the network J_n on $\Theta(n)$ nodes with message-vector length n and edge-vector length $\left(1+n^{-2/3}\right)n$.*

Proof. Recall that J_n is constructed by placing instances $I_2, I_3, I_5 \ldots I_s$ in parallel, where s is the largest prime less than $n^{1/3}$. Consider prime p and the subnetwork I_p in J_n. In our solution, we send $\left\lceil \frac{n}{p} \right\rceil$ unencoded bits of each message across edge C in I_p. A sink requesting p messages must receive a total of pn message bits. A total of $p\left\lceil \frac{n}{p} \right\rceil$ of these message bits are sent via edge C. The remaining at most

$$pn - p \cdot \left\lceil \frac{n}{p} \right\rceil \leq (p-1)n$$

message bits can be transmitted along the direct edge from the source to the sink. Similarly the sink t^* requesting all the messages receives $2p \cdot \left\lceil \frac{n}{p} \right\rceil$ message bits from edge C and can receive the other at most $(2p-2)n$ message bits via the direct edge with capacity $2p-2$.

We can upper bound the length k' of the two vectors transmitted across edge C as follows. For each of the $2p$ messages, we transmit $\left\lceil \frac{n}{p} \right\rceil$ bits on edge C. Therefore, we have:

$$\begin{aligned} 2k' &= 2p \left\lceil \frac{n}{p} \right\rceil \\ &\leq 2n + 2p \\ &= 2n \left(1 + \frac{p}{n}\right) \\ &\leq 2n \left(1 + n^{-2/3}\right) \end{aligned}$$

Therefore the length of each vector sent across edge C is at most $\left(1 + n^{-2/3}\right) n$. We make no use of the extra capacity along any other edge. □

2.4 Discussion and Open Problems

The work in this chapter raises two interesting open problem.

- **For an instance of the network coding problem, find an integer q such that either there exists a solution to the instance using an alphabet of size at most q or the instance is not solvable.** This is closely related to the question of whether the general network coding problem is decidable. For a specific alphabet size, it is possible to search all network codes over that alphabet and check is any of them are network coding solutions. However, in light of Theorem 10 it is not clear which alphabet sizes should be tried.
- **Does every solvable instance admit a solution with a moderate alphabet size, provided the network operates just below capacity?** Our results suggest that using a network at full capacity may be undesirable; even if a solution exists, an enormous alphabet may be required. On the other hand, slightly increasing the network capacity eliminates this problem, at least for the instance we propose. This points toward an exploration of network coding in a model where the network has a small amount of surplus of capacity. Proving that under such a model, every solvable instance admits a solution with a moderate alphabet size is an interesting open question.

2.5 References

Theorems 2, 3 and 4 appeared in [21] and are joint work with Eric Lehman. Theorem 2, which shows that an alphabet of size $(\sqrt{\#sinks})$ sometimes necessary, was independently

proved by Feder, Ron and Tavory[29]. Recently it was shown to be tight by Fragouli and Soljanin[8]. All three algorithms for the multicast problem, the Jaggi-Li algorithm [15], the randomized algorithm due to Ho et al. [14] and the deterministic algorithm due to Harvey et al. [10], require an alphabet of size at most $O(\#sinks)$. Therefore, in terms of the encoding length of an alphabet symbol, all is within an additive constant of optimal.

Dougherty, Freiling, and Zeger[6] independently showed that a problem solvable with a smaller alphabet (say, size 5), may not be solvable with a larger alphabet (size 6). By relating the network coding problem to orthogonal Latin squares, they created an instance of the multicast problem that has a solution as long as the alphabet size is not 2 or 6.

Theorems 10 and 12 and Corollary 11 appeared in [22] and are joint work with Eric Lehman.

Chapter 3

Complexity of Linear Network Coding

In this chapter, we explore the applicability and limitations of linear network coding to a breadth of network coding problems beyond multicast. Our main contribution is a taxonomy of network coding problems based on the connectivity of sources and sinks to the rest of the graph. We describe a three-way partition of possible network coding problems. For the first class, we prove that network coding adds nothing; if sink demands can be satisfied at all, traditional flow techniques provide a solution in polynomial time that does not involve coding. Then we exhibit a second class of network coding problems for which coding is advantageous. In this case, linear solutions can be obtained in polynomial time by adapting the Jaggi-Li multicast algorithm. For the third class of problems, we show that determining whether there exists a solution using linear codes— as in the Jaggi-Li algorithm [15, 24]— is NP-hard. Finally, the techniques developed to prove hardness also yield solvable instances of the network coding problem that do not admit linear solutions.

3.1 Taxonomy of Network Coding Problems

For our purposes, a network coding problem is defined by four attributes: single or multiple sources, single or multiple sinks, message distribution at sources, and message distribution at sinks. Thus a class of network coding problems is defined by a four-tuple (α, β, γ, δ), which is interpreted as follows:

- α is 1 if there is a single source and n if there are multiple sources.
- β is 1 if there is a single sink and m if there are multiple sinks.
- γ is I if all messages are available at every source, D if the sources have available disjoint sets of messages, and A if there are no specific guarantees about the availability of messages at sources. In the case of a single source γ is I.

- δ is I if every sink demands every message, D if sinks demand disjoint sets of messages, and A if there are no specific demand guarantees. In the case of a single source γ is I.

We show that each of the resulting classes of network coding problems falls into one of the following three categories.

Trivial codes suffice: The simplest problems can be solved with a trivial network code, one in which every edge carries an unencoded message. There are two types of problems in this class. The first type is problems with a single sink regardless of the number of sources and distribution of information at the sources. The second type is problems in which each message is available at every source but requested by exactly one sink.

Linear codes suffice: The next set consists of problems for which nontrivial network coding is sometimes necessary. For this class of problems, linear network coding always suffices, if a solution exists. Furthermore, a solution can be found in polynomial time by adapting the Jaggi-Li algorithm. Problems in this class have the property that every sink requests the entire set of available messages. Thus problems in this class may have multiple sources with no guarantees on the distribution of information between sources.

Hard: Finally, the remaining problems sometimes require nontrivial network coding, but determining if a linear solution exists is NP-hard. For this last class of problems, there are instances that do not permit linear solutions but are solvable with vector linear codes.

In the next three subsections, we justify this classification. The following table summarizes these results.

Problem difficulty	# of sources	# of sinks	Information at sources	Information at sinks
Trivial codes suffice	1 or n	1	I, D or A	I
	1 or n	m	I	D
Linear codes suffice	1 or n	m	I, D or A	I
Hard to find linear codes, may need general codes	1 or n	m	I	A
	n	m	D or A	D or A

3.1.1 Easy Problems

Theorem 13 *An instance of network coding problem, (n, m, I, D), with multiple sources, multiple sinks and each message available at every source but requested by exactly one*

sink has a solution if and only if there is a trivial network coding solution.

Proof. We show that such an instance can be solved by augmenting the associated graph G and finding an appropriate flow. Let k be the number of number of commodities and $s_1, s_2 \ldots s_\ell$ be the ℓ nodes which are the sources. We add a super-source s^* and add k edges from s^* to each source s_i, creating a multigraph G'. We also add a super-sink t^*. For each node v let q be the number of commodities i for which $v \in \mathbf{T}(i)$. Add q edges from v to t^*. In G' the original sources and sinks are now intermediate nodes and all messages are available only at the super source s^* and requested by only the super-sink t^*.

Since the super-source can transmit every message unencoded to each node that was a source, if the original problem was solvable, then so is the new one. If the new problem is solvable, then the maximum flow from s^* to t^* must be at least k units; by a counting argument, we can not transmit k messages across a cut with capacity less than k. This implies that there exist k edge-disjoint flow paths from s^* to t^*. Our construction ensures that every flow path traverses a former-source s_i and that exactly q paths through node v if v was a sink for q commodities. Therefore, in the original problem, each message can be routed from a source to the appropriate sink on a path that is edge-disjoint from the paths taken by all other messages. Consequently, no coding is necessary. □

Next we turn our attention to network coding problems with a single sink and show that regardless of the number of sources there always exists a trivial network coding solution whenever the instance is solvable.

Theorem 14 *An instance of the network coding problem with a single sink has a solution if and only if there is a trivial network coding solution.*

Proof. Let $G = (V, E)$ be the graph representing the underlying network of the instance with the single sink t for all k commodities. Create graph $G' = (V', E')$ by adding to G a super source s^*. In addition, for each commodity i, add a node, μ_i, an edge from s^* to μ_i and add an edge from node μ_i to v if $v \in \mathbf{S}(i)$.

Since a network code for the new problem can transmit message M_i unencoded to each node $v \in \mathbf{S}(i)$, if the original problem was solvable, then so is the new one. Now we show that if the new problem is solvable, then the maximum flow from s^* to t must be of size at least k; by a counting argument, we can not transmit k messages across a cut with capacity less than k. If the maximum flow from s^* to t is of size k, then one unit of flow passes through each node μ_i. Therefore we can use the edge-disjoint paths from the maximum flow to route each message to t without using coding. □

3.1.2 Polynomial Time Solvable Linear Coding Problems

Jaggi et al. [15] presented a deterministic polynomial time algorithm for solving the multicast network coding problem. This algorithm can be easily adapted to also solve the

network coding problem with multiple sources provided that all sinks request to receive all messages. Ho et al. [14] provided an efficient randomized algorithm for finding solutions to this class of problems.

The Jaggi-Li algorithm was initially presented in terms of the multicast problem in which every sink wants to receive all available messages from a single source. The first step of the algorithm is to find a flow of size k from the source to each sink. Label the sinks $t_1, t_2 \ldots t_m$. Let F_i be the flow associated with sink t_i. The edges are then considered in topological order. For each sink t_i, there are a set $\mathcal{E}(i)$ of K edges that are the last edge in each flow path of F_i considered by the algorithm. The Jaggi-Li algorithm maintains the invariant that for each sink the set of symbols sent on edges in $\mathcal{E}(i)$ are linearly independent combinations of the messages.

Theorem 15 *An instance of the network coding problem with every message requested by every sink is polynomial time solvable.*

Proof. Consider an instance of the network coding problem in which every message is requested by all the sinks. Let $G = (V, E)$ represent the associated network. Let $t_1, t_2 \ldots t_m$ be the m sinks requesting every message. Create a graph G' by adding a super source s^*. In addition, for each message M_i, add a node μ_i, an edge from s^* to μ_i. In addition, for a node $v \in \mathrm{S}(i)$, add an edge from μ_i to v.

For each sink t_i find a flow F_i of size k in G'. Using the corresponding portions of these flow paths in G and the Jaggi-Li algorithm yields a linear network coding solution in polynomial time. □

Note that in the special case, (1, m, I, I), where there is only a single source the problem in which every sink requests every message corresponds to the multicast case and therefore the proof that it is polynomial time solvable is due to Jaggi et al. [15].

3.1.3 Hard Linear Network Coding Problems

We now consider the class of network coding problems (n, m, D, A), where there are multiple sources with disjoint information and multiple sinks that may demand arbitrary messages. We show that determining whether there exists a linear network coding solution to such a problem is NP-hard. This contrasts with the network coding problems considered previously, for which linear solutions can be found efficiently, provided they exist. We focus on the class (n, m, D, A) for ease of presentation; similar arguments give hardness results for more restricted problem classes. We discuss these extensions at the end of the section.

We begin with a preliminary lemma.

Lemma 16 *Let f_1, f_2, f_3, h, and k be linear functions over a field. If x_1, x_2, and x_3 are uniquely determined by $f_1(x_1, z_1)$, $f_2(x_2, z_2)$, $f_3(x_3, z_3)$, $g(x_1, x_2, x_3, z_1, z_2, z_3)$ and $h(x_1, x_2, x_3, z_1, z_2, z_3)$ then $f_i(x_i, z_i) = \alpha x_i$ for some $i \in \{1, 2, 3\}$ and $\alpha \neq 0$.*

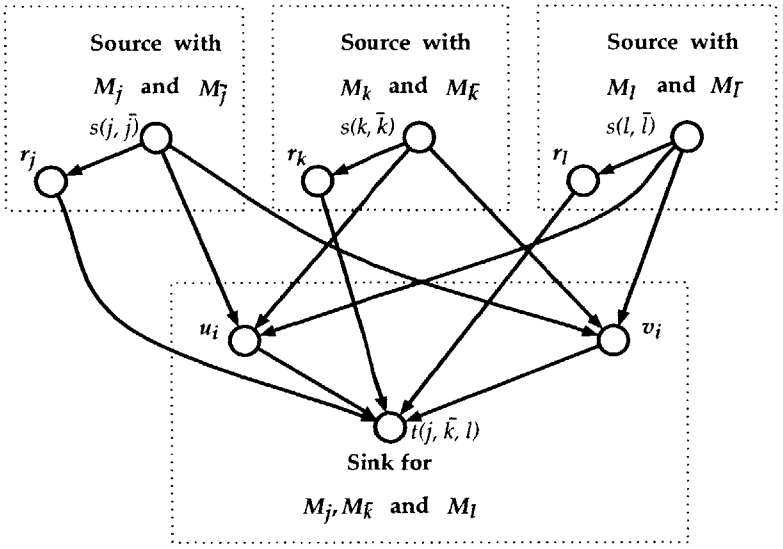


Figure 3-1: Portion of the instance derived from a 3-CNF formula. Each of the top boxes corresponds to a variable gadget. The bottom box corresponds to the clauses gadget for $c_i = (x_j \vee \overline{x_k} \vee x_l)$.

Proof. The values of the five functions can uniquely determine the values of at most five of the variables x_1, x_2, x_3, z_1, z_2 and z_3. If x_i is determined and $f_i(x_i, z_i)$ depends on z_i, then z_i is determined as well. Thus, at least one of the functions f_i does not depend on z_i, and so $f_i(x_i, z_i) = \alpha x_i$ as claimed. □

Reduction: We now describe how to map a 3-CNF formula to an instance of the network coding problem in the class $(n, m$,D,A). Let ϕ be a 3-CNF formula over variables $x_1, x_2, \ldots x_z$. For each variable x_j in ϕ, we make the variable gadget shown in the top three boxes of Figure 3-1. This gadget consists of a source node $s(j, \overline{j})$ which is a source for commodities j and $\overline{j}$. The source node $s(j, \overline{j})$ has an outgoing edge to an intermediate node, r_j. For each clause $c_i = (x_j \vee \overline{x_k} \vee x_l)$, we create the clause gadget shown in the bottom box of Figure 3-1. This consists of a sink $t(i, \overline{k}, l)$, which demands messages M_j, $M_{\overline{k}}$, and M_l, together with two intermediate nodes, u_i and v_i. The variable gadget is connected to the clause gadget as follows. Nodes r_j, r_k, and r_l connect directly to the sink $t(i, \overline{k}, l)$. Nodes s_j, s_k, and s_l all connect to both u_i and v_i. This linkage is illustrated in Figure 3-1. (Note that all three variable gadgets are connected to the clause gadget in the same way, even though variable x_k is negated in the clause. This negation is reflected in the demands at the sink.)

Lemma 17 *A 3-CNF formula ϕ is satisfiable if and only if the corresponding network coding problem has a linear network coding solution.*

Proof. Suppose that ϕ is satisfied by some assignment π. If a variable x_j is true in π, then source $s(j, \overline{j})$ sends message M_j to r_j and sends message $M_{\overline{j}}$ on all other outgoing edges. If x_j is false, then $s(j, \overline{j})$ sends $M_{\overline{j}}$ to r_j and sends M_j on all other edges. Node r_j passes its input to its output. Now consider the gadget associated with a clause such as $c_i = (x_j \vee \overline{x_k} \vee x_l)$. Since π is a satisfying assignment, at least one literal in the clause is true and so at most two literals are false. Each message corresponding to a true literal is sent to the sink from an r-node. Each message corresponding to a false literal is sent from a source node to both u_i and v_i. Since there are at most two such messages, u_i and v_i can relay them both to the sink. (For example, suppose that assignment π makes x_j true and both $\overline{x_k}$ and x_l false. Then the sink receives message M_j via node r_j, message $M_{\overline{k}}$ via node u_i, and message M_l via node v_i.) Thus, all sink demands are satisfied, and the instance of the network coding problem has a linear solution.

In the other direction, suppose that there is a linear solution to the instance of the network coding problem. We construct an assignment π as follows. If the output of r_j is a function of only M_j, then set x_j true. If the output is a function of only $M_{\overline{j}}$, then set x_j false. Otherwise, set x_j arbitrarily. Now consider a clause $c_i = (x_j \vee \overline{x_k} \vee x_l)$. Let f_j, f_k, f_l denote the values output by r_j, r_k, and r_l, and let g and h denote the values output by u_i and v_i. Since the sink can determine messages M_j, $M_{\overline{k}}$, and M_l, Lemma 16 implies that either f_j depends only on M_j, f_k depends only on $M_{\overline{k}}$, or f_l depends only on M_l. Therefore, at least one of the literals x_j, $\overline{x_k}$, or x_l is true, and the clause is satisfied by assignment π. Therefore, π is a satisfying assignment for the 3-CNF ϕ. □

Lemma 17 establishes the hardness of network coding problems in the class (n, m, D, A). We conclude this section by describing minor adjustments to the reduction which yield the following more general result.

Theorem 18 *Determining whether there exist linear network coding solutions to network coding problems in the classes (1, m, I, A), (n, m, D or A, D or A), and (n, m, I, A) is NP-hard.*

We've shown how to map a 3-CNF formula to an instance in the class (n, m, D, A). We now describe extensions of this result to the remaining classes in Theorem 18. We start with the class (1, m, I, A). For this problem we show how to use sinks to fix the way that messages are distributed from a common source. We then turn to the class (n, m, D, D). We only sketch the reduction here. The key elements in both of these reductions are found in the preceding discussion about the class (n, m, D, A). Finally, since (n, m, I, A) contains (1, m, I, A) and (n, m, A, A) contains (n, m, D, A) the theorem follows.

Lemma 19 *A 3-CNF formula ϕ is satisfiable if and only if the corresponding instance of the network coding problem (1, m, I, A) has a linear network coding solution.*

Proof. Instances in the class (1, m, I, A) have a single source with all the messages. There are no restrictions on the demands at the sinks. Let G be the graph obtained from the

reduction given above for a 3-CNF formula ϕ. We augment G with a super-source s^* and an edge of capacity 2 from s^* to $s(j, \overline{j})$ for each node $s(j, \overline{j}) \in G$. In addition, we change node $s(j, \overline{j})$ from being a source for messages M_j and $M_{\overline{j}}$ to being a sink for both messages. Each node $s(j, \overline{j})$ can receive the two messages it now requests unencoded. Therefore, if the original instance had a linear solution, the augmented instance does as well. Now suppose the augmented instance has a linear solution, then $s(j, \overline{j})$ receives two linear functions of the messages and is able to decode M_j and $M_{\overline{j}}$. Therefore the linear functions that node $s(j, \overline{j})$ receives do not depend on any messages other than M_j and $M_{\overline{j}}$. A linear solution for the new instance can be mapped to a linear solution for the original instance. The result follows from Lemma 17. □

Augmenting the reduction to create an instance in the class (1, m, I, A) was straightforward. The next lemma deals with the class (n, m, D, D). While the heart of the reduction is the same as given for the class (n, m, D, A) we need to insure that each message is requested by exactly one sink. This restriction adds some complexity to the structure of the instance produced by the reduction. Figure 3-2 shows the main changes to the reduction.

Lemma 20 *A 3-CNF formula ϕ is satisfiable if and only if the corresponding instance of the network coding problem (n, m, D, D) has a linear network coding solution.*

Proof. Instances in the class (n, m, D, D) have a single source for each message and a single sink requesting each message. In the reduction given for the class (n, m, D, A) for each clause with variable x_j there was a sink requesting message M_j. We alter the reduction given above in order to insure each message is requested by exactly one sink. Figure 3-2 shows most of the additions to the reduction. Let G be the graph obtained according to the reduction given for (n, m, D, A) from 3-CNF formula ϕ. For variable x_j let $C(j)$ be the set of clauses in which the literal x_j appears. Let $C(\overline{j})$ be the clauses in which $\overline{x_j}$ appears. Without loss of generality assume that for all variables x_j, there are no clauses with both x_j and $\overline{x_j}$.

For each variable x_j, we add two sources $s(j)$ and $s(\overline{j})$ and an edge from each of these new sources to node $s(j, \overline{j})$. Node $s(j, \overline{j})$ is no longer a source. We also add two sinks $t(j)$ and $t(\overline{j})$ which are the only sinks requesting M_j and $M_{\overline{j}}$ respectively.

For each clause $c_i = (x_j \vee \overline{x_k}, \vee x_l)$ there are three new messages $Y_{ij}, Y_{i\overline{k}}$ and Y_{il} with new sources $\sigma(ij)$, $\sigma(i\overline{k})$ and $\sigma(il)$. We add a single sink $\tau(i)$ requesting all three messages associated with clause c_i. We create the remainder of the network to insure that $\tau(i)$ must also receive $M(j), M(\overline{k})$ and $M(l)$ in order to decode $Y_{ij}, Y_{i\overline{k}}$ and Y_{il}.

For each variable x_j, we add a path q_j of length three from $s(j)$ to $t(j)$. We also add a single path p_{ij} from $\sigma(ij)$ to $\tau(i)$ of length three. The middle edge on p_{ij} is the middle edge from q_j. This common edge must transmit a linear function which depends on both M_j and Y_{ij}. Therefore, sink $\tau(i)$ will need to receive message M_j in order to recover Y_{ij}. We add an edge of capacity three from the node $t(i, \overline{k}, l)$ to $\tau(i)$. We construct similar paths for the other variables and clauses.

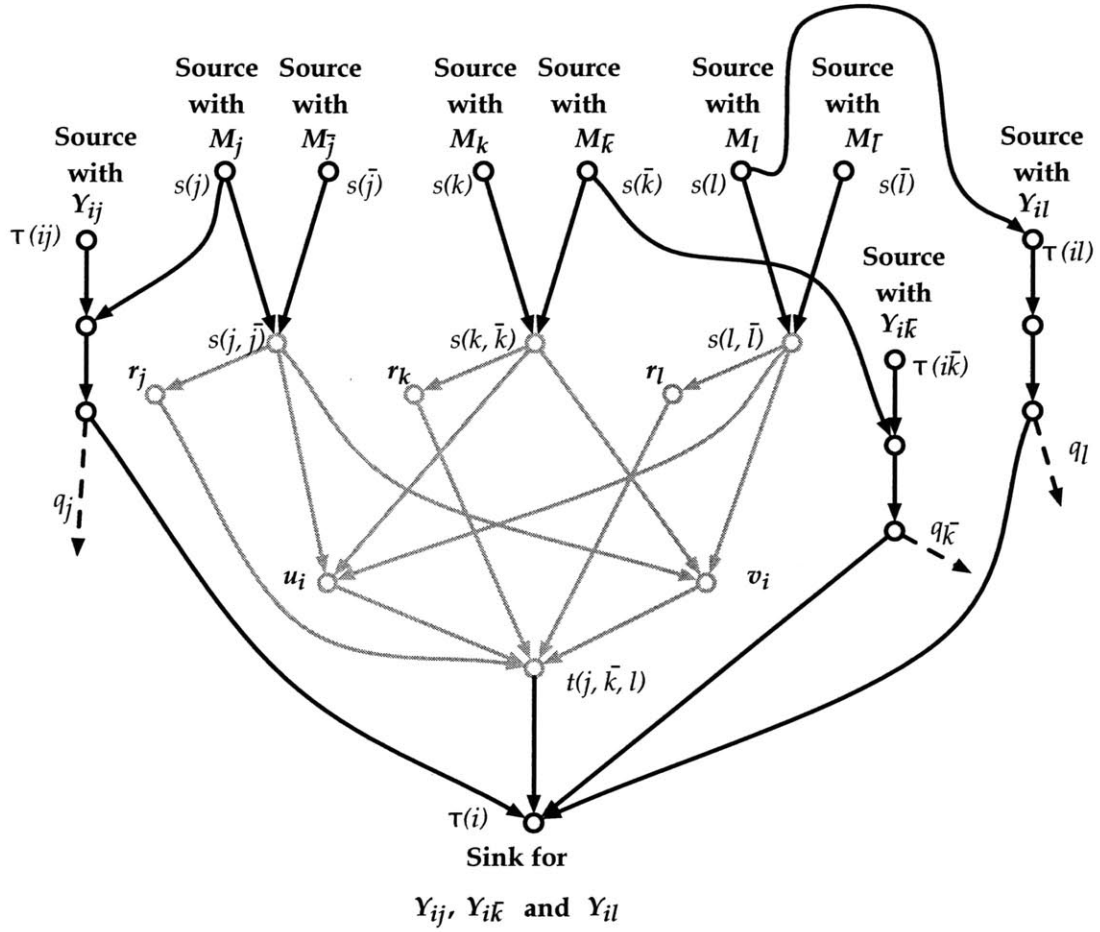


Figure 3-2: The gray portion corresponds with the part of the graph derived according to the reduction for (n, m, D, A). The black nodes and edges are the alterations to the instance which put it in the class (n, m, D, D).

Lastly, we need to add some direct edges from sources to sinks for other messages. The middle edge on the path q_j is the middle edge on all paths $p_{\ell j}$ such that $c_\ell \in C(j)$. Therefore, we add an edge from source $s(\ell j)$ to sink $\tau(i)$ for all $\ell \neq i$ and $c_i, c_\ell \in C(j)$. We also add an edge from source $s(ij)$ to sink $t(j)$ for all $c_i \in C(j)$. We add similar edges for the other messages.

We now show that the augmented instance has a linear solution if an only if the original instance did. By construction, each message associated with a variable can be decoded by its sink. This is because there is a path q_j from $s(j)$ to $t(j)$ and direct edges to $t(j)$ from every source for a message whose only path to a sink uses the middle edge in q_j.

Similarly, for a sink $\tau(i)$ associated with a clause c_i, the messages receive along the path p_{ij} will be a linear combination of Y_{ij}, M_j and messages $Y_{\ell j}$ such that $c_\ell \in C(j)$. Since there is a direct edge from every source $s(\ell j)$ to $\tau(i)$ such that $\ell \neq i$, $\tau(i)$ can recover a linear combination Y_{ij} and M_j. Note that this linear combination must depend on both Y_{ij} and M_j. Likewise, for the other messages which $\tau(i)$ requests. Therefore, there exists a linear solution to this instance if an only if $\tau(i)$ can receive $M_j, M_{\overline{k}}$ and M_l from $t(i, \overline{k}, l)$. Therefore the augmented instance has a linear solution if and only if the original instance admits a linear solution. □

3.2 Insufficiency of Field Linear Codes

If we do not insist on a linear solution, then the coding problems generated by 3-CNF formulas are *always* solvable— even if the 3-CNF formula was not satisfiable. This is in stark contrast to the multicast case where a linear solution exists if any solution exists. At the end of this section we present a simple solvable network with no linear solution.

Theorem 21 *There are solvable network coding problems in the classes (1, m, I, A), (n, m, D or A, D or A), and (n, m, I, A) for which there is no linear network coding solution.*

Proof. [for the class (n, m, D, A)] Let ϕ be an unsatisfiable 3-CNF formula. By Lemma 17, there does not exist a linear solution to the corresponding network coding problem. However, we can construct a nonlinear network coding solution to this network coding problem as follows. Take the alphabet $\Sigma = \Gamma \times \Gamma$, where Γ is an arbitrary alphabet. Thus, each message is a pair of symbols drawn from Γ. Each source $s(j, \overline{j})$ sends the first symbol of messages M_j and $M_{\overline{j}}$ to node r_j and sends the second symbol of these two messages on all other outgoing edges. Node r_j relays its input to its output. Now consider a clause $c_i = (x_j \vee \overline{x_k} \vee x_l)$. The sink $t(j, \overline{k}, l)$ requests the three messages M_j, $M_{\overline{k}}$, and M_l. From nodes r_j, r_k, and r_l, the $t(j, \overline{k}, l)$ receives the first symbol of all six messages M_j, $M_{\overline{j}}$, M_k, $M_{\overline{k}}$, M_l, and $M_{\overline{l}}$. Furthermore, the nodes u_i and v_i receive the second symbol of all six messages. Since each of these nodes can send two symbols from Γ to the $t(j, \overline{k}, l)$, they can together provide the second symbol of the three messages $M_j, M_{\overline{k}}$ and M_l. □

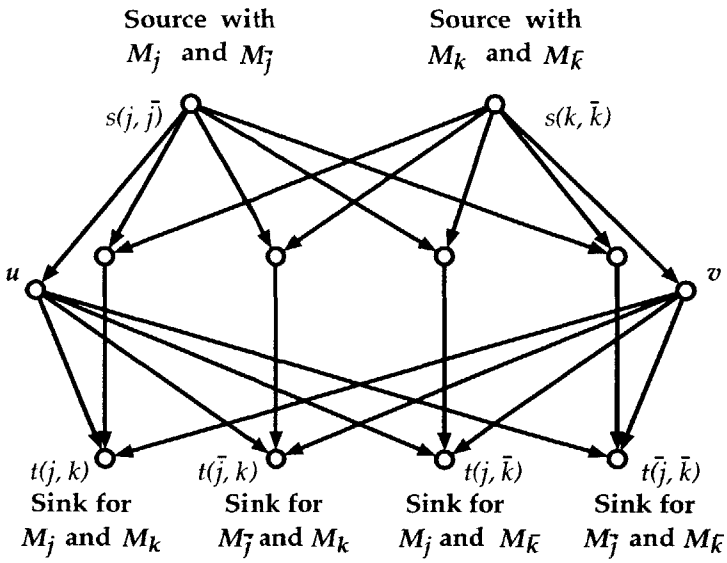


Figure 3-3: A network coding instance with a vector linear solution but no field linear solution.

The above theorem demonstrates that linear codes do not suffice for a large class of general network coding problems. For concreteness, we present a simple network which does not have a linear solution but does have a vector linear solution.

The example network presented in Figure 3-3 is derived by considering the unsatisfiable 2-CNF formula $(x_j \vee x_k) \wedge (\overline{x}_j \vee x_k) \wedge (x_j \vee \overline{x}_k) \wedge (\overline{x}_j \vee \overline{x}_k)$. First consider a vector linear solution over $\Sigma = \Gamma \times \Gamma$. Node $s(j, \bar{j})$ sends the first symbol from M_j and the first symbol from $M_{\bar{j}}$ to node u. Node $s(j, \bar{j})$ sends the second symbol from M_j and $M_{\bar{j}}$ to the four unlabeled intermediate nodes. Similarly for $s(k, \overline{k})$. It is check that all sinks receive both requested messages.

Now suppose their is a linear solution. Let $f_u(M_j, M_{\bar{j}})$ and $f_v(M_k, \overline{k})$ be the linear functions sent to u and v respectively. Node $t(j, k)$ receives three linear functions. Since f_u is a function of M_j and $M_{\bar{j}}$ and f_v is a function of M_k and $M_{\overline{k}}$, one of these functions must depend on only one of its inputs. Without loss of generality, assume $f_u(M_j, M_{\bar{j}}) = M_j$. Then node $t(\bar{j}, \overline{k})$ must receive $M_{\bar{j}}$ unencoded and therefore must also receive $M_{\overline{k}}$ unencoded. This implies that $f_v(M_k, M_{\overline{k}}) = \alpha M_{\overline{k}}$ for some non-zero α. Therefore, regardless of the third linear function transmitted to node $t(\bar{j}, k)$, it cannot possibly reconstruct messages $M_{\bar{j}}$ and M_k. Hence there is no linear solution.

3.3 Discussion

There are two, very important, open questions related to the work in this chapter.

- **What is the computational complexity of the general network coding problem?** It is not even known if the general network coding problem is decidable.
- **Find an algorithm for any class of problems for which linear codes are not sufficient.** For the classes of problems for which linear solutions are not sufficient, there are no known algorithms (regardless of efficiently). This set of problems represents the majority of network communication problems. All known algorithms for the multicast problem use the fact that any solvable instance admits a linear solution. Therefore, a algorithm, regardless of efficiency, for any class of problems for which linear codes are not sufficient would necessarily introduce new techniques.

3.4 References

There are now three efficient algorithms for the multicast problem. Ho et al. [14] presented a randomized algorithm which can be implemented in a distributed fashion[19]. Jaggi et al. found a fast implementation for the algorithm due to Li, Yeung and Cai [24]. In addition, Harvey, Karger and Murota [10], building on the framework of Koetter and Médard [19], used matrix completion techniques to derive an entirely different deterministic algorithm. Each of these algorithms can be used to solve all the network coding problems for which linear codes are sufficient.

Koetter and Médard conjectured that any solvable network coding problem would have a linear solution. Theorem 21 answered this conjecture in the negative. However, the networks used in the proof of the theorem admit vector linear solutions. Effros et al. presented a similar example to the network in Figure 3-3 and conjectured that linearity, in some form, is always sufficient. In [6], Doughtery, Freiling and Zeger showed that every instance with two commodities that admits a solution over the alphabet $\{0, 1\}$ admits a linear solution. However, if the instance has three or more commodities, a solution over $\{0, 1\}$ does not imply the existence of a linear solution.

A series of papers have further explored the uses and limits of codes which satisfy some linearity condition [21, 33, 7, 24, 31].

Chapter 4

Entropy Based Upper Bounds

The k-pairs communication problem represents an important class of problems in terms of both theory and practice. The problem of allowing k point-to-point connections in a communication network arises in many practical situations. In addition, this problem has a strong connection to the classical multicommodity flow problem. The primary motivation for the work in the next three chapters is understanding this fundamental communication problem. For this reason, we present the results in terms of the k-pairs communication problem. However, many of them apply directly or are easily extended to general network coding problems. At the end of each of of this chapter we discuss the straightforward extensions of these ideas to the general network coding problem.

In this chapter we consider a definition of a network code that allows us to consider questions about the maximum rate of a network coding solution. In the previous two chapters, network codes used the same alphabet for all messages and all edges. Suppose that we used a much larger alphabet on the edges. Some instances that previously did not have a solution, now would have solutions. However, these solutions seem "worse" than a solution which uses the same size alphabet for all sources and all edges. We quantify this by defining the rate of a network coding solution.

We then consider the problem of determining the maximum rate at which every source can simultaneously transmit information to its sink. For the multicast problem there is a min-cut condition which determines this rate[24]. For the k-pairs communication problem, we consider two possible cut conditions. The first is the value of the sparsest edge cut in the graph. For directed graphs network codes can achieve much higher rates than suggested by the value of the sparsest edge cut.

We define a quantity called the *meagerness* of a cut. We show that while meagerness is an upper bound on the maximum rate achievable in a directed graph, it is not a lower bound. To prove this, we present an instance of the k-pairs communication problem in which the most meager cut has value 1 but the maximum coding rate is $2/3$. The proof of this result relies on information theoretic arguments and leads to general upper bounds on

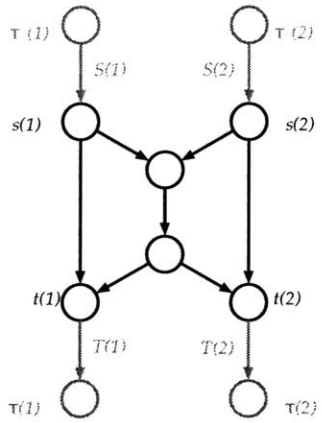


Figure 4-1: In black is an instance of the k-pairs communication problem. The gray edges and nodes are added to create the augmented graph.

the maximum achievable rate.

4.1 Definitions

We redefine a network code and a network coding solution. These definitions rely on an augmented graph $\hat{G}$. By augmented the graph, we can treats source messages and symbols transmitted on edges in a similar way. Figure 4-1 shows the underlying graph and augmented graph for an instance of the k-pairs communication problem.

Definition 4 (Augmented Graph $\hat{G}$) *Given a instance of the k-pairs communication problem on underlying directed graph G, the* augmented graph $\hat{G} = (\hat{V}, \hat{E})$ *is obtained by applying the following transformation to G.*

- *For each commodity i, we add a new vertex $\sigma(i)$ with one outgoing edge $S(i) = (\sigma(i), s(i))$ and no incoming edges. The set of all edges $S(i)$ is denoted by $\mathcal{S}$.*
- *For each commodity i, we add a new vertex $\tau(i)$ with one incoming edge $T(i) = (t(i), \tau(i))$ and no outgoing edges. The set of all edges $T(i)$ is denoted by $\mathcal{T}$.*

A generalized edge *of G is an edge of $\hat{G}$. If $e = (u, v)$ is such an edge, the set of all incoming edges to u will be denoted by* $\text{In}(e)$.

We now redefine a network code and network coding solution. For the remainder of this thesis, network codes will be specified on the augmented graph $\hat{G}$. Note that the augmented graph is acyclic if the underlying graph is acyclic.

Definition 5 (Network Code) *Given a network coding instance with underlying acyclic graph G and augmented graph $\hat{G}$, a network code is given by specifying the following data:*

- *An* edge alphabet $\Sigma(e)$ *for each generalized edge* e.
- *A* function f_e *for each generalized edge* e.

For a specified network code and set A of generalized edges, let

- $\Sigma(A) = \prod_{e \in A} \Sigma(e)$.
- $f_A : \Sigma(\mathcal{S}) \rightarrow \Sigma(A)$ such that for every k-tuple of messages M,

$$f_A(M) = \{f_{e_1}(M), f_{e_2}(M) \ldots f_{e_{|A|}}(M)\}$$

- $\text{In}(A) = \cup_{e \in A} \text{In}(A)$.

Definition 6 (Network Coding Solution) *A network code defined on an augmented directed acyclic graph $\hat{G}$is a* solution *to an instance of the k-pairs communication problem if it meets the following conditions. Let M be the k-tuple of messages.*

- *For every edge $S(i) \in \mathcal{S}$, $f_{S(i)}(M) = M_i$.*
- *For every generalized edge $e \in \hat{E} \setminus \mathcal{S}$, the function $f_e : \prod_i \Sigma(\mathcal{S}) \rightarrow \Sigma(e)$ is computable from the functions on edges in* $\text{In}(e)$.
- *For every edge $T(i) \in \mathcal{T}$, $f_{T(i)}(M) = M_i$.*

Our new definition of a network coding solution does not restrict the amount of information sent across an edge. Intuitively in a solution which uses much larger edge alphabets than source alphabets the sources communicate at a much lower information rate than the rate of the links in the network. To precisely capture this, we define the rate of a network coding solution.

Definition 7 (Rate) *We say a network coding solution defined on the augmented graph $\hat{G}$ achieves rate r if there exists a constant b such that* $\log_b |\Sigma(e)| \leq c(e)$ *for each $e \in E$, and* $\log_b |\Sigma(S(i))| \geq r d_i$ *for each commodity i.*

The constant b in the definition of rate is chosen to allow a network code to use any choice of alphabet size. The idea is that a network code can use a large alphabet on the edges but if the rate of the solution is still good then the alphabet for each commodity must also be large.

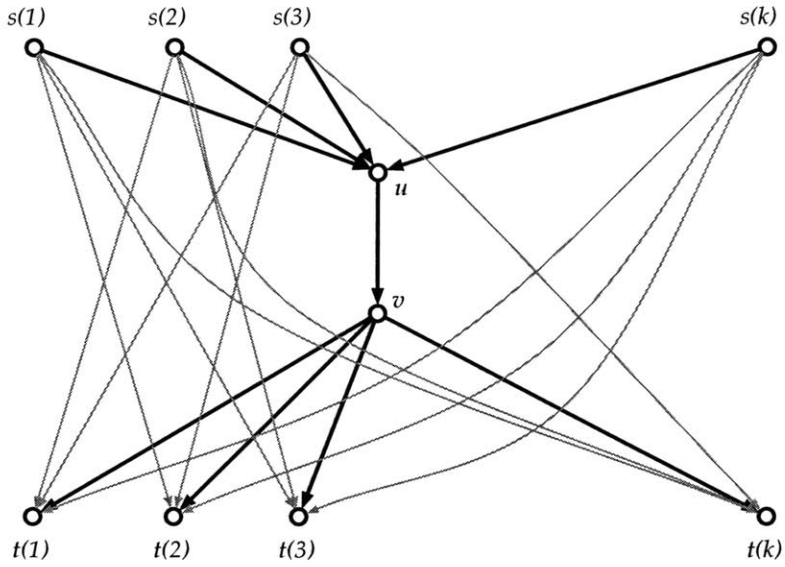


Figure 4-2: In this example, source $s(i)$ has an edge to every sink other than $t(i)$ and an edge to u. Sink $t(i)$ has an in-edge from v and an in-edge from every source except $s(i)$. Without network coding the maximum rate is $1/k$. A rate 1 network coding solution sends $M_1 \oplus M_2 \oplus \ldots M_k$ over edge (u, v). Sink $s(i)$ also receives M_j directly from $s(j)$ for all $j \neq i$.

In this chapter we focus on the k-pairs communication problem and consider various necessary and sufficient conditions. We first consider some natural cut-based conditions and show that while some are necessary conditions, none are sufficient conditions. The proofs of these results lead to conditions based on the structure of the underlying graph and properties of the entropy function. We can prove that a network coding solution for any *general* instance must satisfy these conditions. It is an open question as to whether these are sufficient conditions.

4.2 Sparse Edge Cuts

In our search for necessary and sufficient conditions, it is natural to consider the capacity of any edge set whose removal disconnects a number of source-sink pairs. However, consider a modification of the canonical example in Figure 4-2. (This example is due to Nick Harvey and was noticed independently by Li and Li [26].) The middle vertical edge has capacity 1. Removing this edge from the graph disconnects all sources from their respective sinks. Therefore, the sparsest *edge* cut in the graph has sparsity $1/k$. At first glance it appears that the best possible rate requires that all source-sink pairs share the capacity of the middle

vertical edge. Using coding over a finite field, a rate 1 network coding solution sends the exclusive or of all messages across the middle vertical edge. Each sink also receives unencoded every message other than the one it requests. Since a rate 1 solution exists, it is clear that the capacity of a set of edges disconnecting each source from its respective sink is not an upper bound on the achievable coding rate.

It is worthwhile to consider why there exists a solution of rate 1 when the sparsity of this graph is $1/k$. The network coding solution sends sinks information that they didn't request but that enables them to decode the message they want. This motivates the next cut-based condition. Specifically, we consider cuts that separate an entire set of sources from an entire set of sinks. We require that such a cut separate sources even from sinks for other commodities.

4.3 Meagerness

Given a set of edges A, we define the capacity of A to be the sum of the capacities of the edges in A.

$$c(A) = \sum_{e \in A} c(e)$$

In light of the above example, consider edge cuts which separate a set of sources entirely from a set of sinks.

Definition 8 (Isolation) *Given an edge set $A \subseteq E$ and a subset of commodities $P \subset \mathcal{I}$, we say A* isolates *P if for all $i, j \in P$, every path from $s(i)$ to $t(j)$ intersects A.*

A cut A that isolates a set of commodities P must disconnect each source for a commodity in P from the sink for *every* commodity in P. The demand of the commodities in P is written $d(P)$.

Consider again the example in Figure 4-2. Suppose the set of commodities P contains all commodities. A cut A which isolates P must separate the source $s(1)$ from all sinks. This implies that all the gray out-edges from $s(1)$ must be in A. Similarly, all the gray out-edges from $s(2)$ must be in A. Therefore, cuts which isolate a set P may need to be much larger than a cut that separates every source in P from its corresponding sink.

Definition 9 (Meagerness) *The* meagerness *of a cut A is defined to be ∞ if A does not isolate any commodities and otherwise is defined as:*

$$\mathcal{M}(A) = \min_{P: A \text{ isolates } P} \left\{ \frac{c(A)}{d(P)} \right\}$$

For a graph G, the value of most meager cut in G is denoted by

$$\mathcal{M}_G = \min_{A \subseteq E} \mathcal{A}$$

Note that the meagerness of a graph G is defined in terms of G itself and not $\hat{G}$. Consider again our example in Figure 4-2. If we choose P to be a single commodity, say $P = \{1\}$, then we only need to ensure that A separates the source for that one commodity from its sink. Removing the middle edge separates $s(1)$ from $t(1)$ and therefore isolates the set $P = \{1\}$. Therefore, the meagerness of the graph in Figure 4-2 is 1.

The following lemma upper bounds the achievable rate by the meagerness of G.

Lemma 22 *Consider an instance of the k-pairs communication problem on a directed graph $G = (V, E)$. The maximum rate achievable by a network code is at most $\mathcal{M}_G$.*

Proof. Let A be the most meager cut in G and let P be the set of commodities isolated by A. Every path from the sources for a commodity in P to a sink for any commodity in P must intersect A. Therefore, the information transmitted on edges in A must be sufficient to uniquely determine the $|P|$-tuple of messages set from sources for commodities in P.

Let $\hat{G}$ be the augmented graph. The symbols on edges in A can take on $|\Sigma(A)| = \prod_{e \in A} |\Sigma(e)|$ different values. There are $\prod_{i \in P} |\Sigma(S(i))|$ different $|P|$-tuples of messages for the commodities in P.

$$\prod_{i \in P} |\Sigma(S(i))| \leq \prod_{e \in A} |\Sigma(e)|$$

Let r be the rate of this solution. Then there exists a constant b such that $\log_b |\Sigma(e)| \leq c(e)$ for all $e \in E$ and $\log_b |\Sigma(S(i))| \geq r d_i$.

$$\begin{aligned} \log_b \left(\prod_{i \in P} |\Sigma(S(i))| \right) &\leq \log_b \left(\prod_{e \in A} |\Sigma(e)| \right) \\ \sum_{i \in P} \log_b |\Sigma(S(i))| &\leq \sum_{e \in A} \log_b |\Sigma(e)| \end{aligned}$$

The left side is at least $\sum_{i \in P} r d_i = r d(P)$ and the right-hand side is at most $\sum_{e \in A} c(e) = c(A)$. □

4.4 The Split Butterfly

In the previous section we proved that the meagerness of a graph is an upper bound on the maximum achievable network coding rate. In this section we introduce an example,

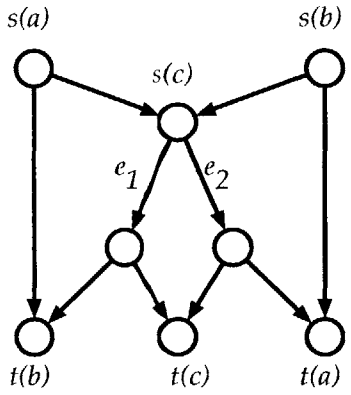


Figure 4-3: The meagerness of this graph is 1 but the maximum rate at which all three sources can communicate with their respective sinks is $2/3$.

called the split butterfly, in which the maximum achievable rate using network coding is strictly smaller than the meagerness of the graph. In the next section we introduce the new techniques which are used to prove that the maximum achievable rate is less than the meagerness. We then prove the gap between the meagerness of the graph and the maximum achievable coding rate for the example.

Consider the instance depicted in Figure 4-3. Let $G = (V, E)$ be the graph corresponding to this instance. The set of commodities is $\mathcal{I} = \{a, b, c\}$ and the demand for each commodity is 1. Each edge in G has capacity 1.

We show that the value of the most meager cut in G is 1. We also give a solution of rate $2/3$. Later, we will show that there does not exist a network coding solution for any rate $r > 2/3$.

Lemma 23 *The value of the most meager cut in G is 1.*

Proof. For each subset of the commodities, we determine the meagerest cut separating them. Since all edges have capacity 1 a cut that must cut ℓ edge-disjoint paths to isolate P must have capacity at least ℓ. By considering all possible sizes for P, we show there are at least $|P|$ edge-disjoint paths which need to be cut in order to isolate P.

Suppose P has one commodity i. For all i, there is a path from $s(i)$ to $t(i)$ with capacity 1. Now suppose P has two commodities. If $c \in P$, then the two edge-disjoint paths from $s(c)$ to $t(c)$ must be cut. If $P = \{a, b\}$, then the edges $(s(a), t(b))$ and $(s(b), t(a))$ must be cut. Finally, if $P = \{a, b, c\}$ then the two edge-disjoint paths from $s(c)$ to $t(c)$ must be cut. In addition the two edges $(s(a), t(b))$ and $(s(b), t(a))$ must be cut. Therefore the capacity of a cut A isolating P is at least $|P|$. Since each commodity has demand 1, this proves the claim. □

A rate $2/3$ solution for the instance in Figure 4-3 is the following. Let $\Sigma(S(i)) = \{0,1\}^2$ for all i and $\Sigma(e) = \{0,1\}^3$ for all e. The message M_a is transmitted on the only path from $s(a)$ to $t(a)$. Similarly, the message M_b is transmitted on the only path from $s(b)$ to $t(b)$. Each of the bits for message M_c are transmitted on one of the two paths from $s(c)$ to $t(c)$. Therefore, edges e_1 and e_2 must transmit 3 bits each. Every other edge transmits fewer than 3 bits.

Our next step is to prove an upper bound of $2/3$ on the maximum network coding rate for the graph in Figure 4-3. This will show that the meagerness of this graph is not equal to the maximum achievable rate. Intuitively, the edges e_1 and e_2 must transmit all the information from the three sources to the three sinks. The proof of this result is based on tighter necessary conditions. To derive the tighter conditions we take an information theoretic perspective on the problem of network coding. We first present the necessary definitions and conditions and then return to considering this example.

4.5 Entropy Based View of Network Coding

We describe the entropy based view of network coding for the k-pairs communication problem. Let $\hat{G}$ be the augmented graph for an instance of the k-pairs communication problem. Suppose for each source edge $S(i)$ we chose a message uniformly at random from $\Sigma(S(i))$. We assume the messages are chosen independently. For each generalized edge e, we associate with e a random variable Y_e.

$$\Pr(Y_e = \alpha) = \Pr(f_e(M) = \alpha)$$

where M is chosen uniformly at random from $\Sigma(\mathcal{S})$. For a set $A = \{e_1, e_2 \ldots e_{|A|}\}$ of generalized edges, $Y_A = \{Y_{e_1}, Y_{e_2} \ldots Y_{e_{|A|}}\}$. We use the standard definition of the joint entropy of a set of random variables (See Appendix A for definitions and statements of basic results). As a shorthand, we use $H(A)$ to refer to $H(Y_A)$ for any set A of generalized edges.

The following Lemma applies to any network coding solution on a directed acyclic graph.

Lemma 24 *Let A be a set of generalized edges. Given a network coding solution,*

$$H(\mathcal{S}, A) = H(\mathcal{S})$$

and for all $i \in \mathcal{I}$

$$H(S(i), A) = H(T(i), A)$$

The above lemma follows directly from the definition of a network coding solution. The following lemma relates the entropy of sources and edges in a rate r solution.

Lemma 25 *Given a network coding solution defined on an augmented graph $\hat{G}$ of rate r, there exists a constant b such that the following hold. For all edges $e \in E$,*

$$H(e) \leq c(e) \log_2 b$$

and for all commodities $i \in \mathcal{I}$,

$$H(S(i)) \geq r d_i \log_2 b$$

Proof. By the definition of a rate r code, there exists a constant b such that $\log_b |\Sigma(e)| \leq c(e)$ for all $e \in E$ and $\log_b |\Sigma(S(i))| \geq r d_i$ for all $i \in \mathcal{I}$. Therefore,

$$\begin{aligned} |\Sigma(e)| &\leq b^{c(e)} \\ |\Sigma(S(i))| &\geq b^{r d_i} \end{aligned}$$

Since we choose each source message uniformly at random from $\Sigma(S(i))$,

$$\begin{aligned} H(S(i)) &= -\sum_{\alpha \in \Sigma(S(i))} \Pr\left(f_{S(i)}(M) = \alpha\right) \log_2 \Pr\left(f_{S(i)}(M) = \alpha\right) \\ &= -\sum_{\alpha \in \Sigma(S(i))} \frac{1}{|\Sigma(S(i))|} \log_2 \frac{1}{|\Sigma(S(i))|} \\ &= \log_2 |\Sigma(S(i))| \\ &\geq \log_2 (b^{r d_i}) \\ &= r d_i \log_2 b \end{aligned}$$

For edge $e \in E$,

$$\begin{aligned} H(e) &= -\sum_{\alpha \in \Sigma(e} \Pr\left(f_e(M) = \alpha\right) \log_2 \Pr\left(f_e(M) = \alpha\right) \\ &\leq -\sum_{\alpha \in \Sigma(e)} \frac{1}{|\Sigma(e)|} \log_2 \frac{1}{|\Sigma(e)|} \\ &= \log_2 |\Sigma(e)| \\ &\leq \log_2 (b^{c(e)}) \\ &= r d_i \log_2 b \end{aligned}$$

Where the second inequality follows from the fact that the entropy function is maximized by the uniform distribution. □

4.6 Downstreamness

We now consider a relationship between the entropy of edge sets in $\hat{G}$. We begin by defining a relationship between edge sets called downstreamness. We then show that if edge set B is downstream of edge set A, $H(A) \geq H(B)$. This will be a key to analyzing the example in Figure 4-3.

Definition 10 (Downstreamness) *Let $\hat{G}$ be the augmented graph for an instance of the k-pairs communication problem. Given two sets A and B of generalized edges, we say B is downstream of A if for all edges e in B, every path p such that $e \in p$ and $p \cap \mathcal{S} \neq \emptyset$ satisfies $p \cap A \neq \emptyset$. We write $A \rightsquigarrow B$ if B is downstream of A.*

As an example, consider the instance in Figure 4-4. The edge $T(b)$ is downstream from the pair of edges $\{S(a), e_1\}$. Similarly, $\{S(b), e_2\} \rightsquigarrow T(a)$. Note that our definition of $A \rightsquigarrow B$ allows for the possibility that $A = \emptyset$. A set of edges B is downstream from the empty set when there is no path from a source to an edge in B in $\hat{G}$.

Lemma 26 *For any sets A and B of generalized edges, if $A \rightsquigarrow B$, then there exists a function $h_{AB} : \Sigma(A) \rightarrow \Sigma(B)$ such that*

$$f_A \circ h_{AB} = f_B$$

Proof. Let D be the set of all generalized edges that are downstream of A. The set B is a subset of D. We prove for each edge $e \in D$, there exists a function $h_{Ae} : \Sigma(A) \rightarrow \Sigma(e)$ such that

$$f_A \circ h_{Ae} = f_e$$

Order the edges in D in topological order so that every edge in $\mathrm{In}(e)$ is either not in D or comes before e in the ordering. For the base case, let e be the first edge in the ordering. Then every edge in $\mathrm{In}(e)$ is not in D. If $e \in A$ then the claim follows immediately. Otherwise, by the definition of downstreamness, e is an edge with no in-edges in $\hat{G}$. In this case f_e must be a constant function since there is no path from any source to e. Therefore, the claim follows.

Now assume the claim is true for edges $e_0, e_1 \ldots e_{k-1} \in D$. Let

$$D_{k-1} = \{e_0, e_1, \ldots e_{k-1}\}$$

Suppose $\tilde{e}$ is an in-edge to e_k. If $\tilde{e} \notin D$, then there is a path from a source to $\tilde{e}$ and therefore also a path from a source to e_k. Thus every edge in $\mathrm{In}(e_k)$ is in D_{k-1}. By the definition of a network coding solution, there exists a function $g_{e_k} : \prod_{e \in \mathrm{In}(e_k)} \Sigma(e) \rightarrow \Sigma(e_k)$ which maps the functions on edges in $\mathrm{In}(e_k)$ to f_{e_k}. Using g_{e_k} and the functions h_{Ae_i} for $e_i \in D_{k-1}$ we can construct h_{Ae_k}. □

The definition of downstreamness implies a relationship between the $H(A)$ and $H(B)$ if $A \rightsquigarrow B$. We prove this relationship in the following lemma. Downstreamness can be viewed as a structural relationship which implies that the corresponding random variables form a Markov chain.

Lemma 27 *For sets of generalized edges A and B, if $A \rightsquigarrow B$, then*

$$H(A) \geq H(B)$$

Proof.

Let Y_U refer to the random variable associated with a set U of generalized edges. By downstreamness, $Y_S \rightarrow Y_A \rightarrow Y_B$ forms a Markov chain. By the data-processing inequality, $I(Y_S, Y_A) \geq I(Y_S, Y_B)$. Equivalently,

$$H(Y_A) - H(Y_A|Y_S) \geq H(Y_B) - H(Y_B|Y_S)$$

Since Y_S is the random variable representing the choice of messages transmitted on all out-edges from sources, $H(Y_A|Y_S) = 0$ and $H(Y_B|Y_S) = 0$. Therefore, $H(Y_A) \geq H(Y_B)$. □

Using downstreamness and the properties of the entropy function, we now prove that the maximum rate achievable with network coding for the instance in Figure 4-3 is $2/3$. Figure 4-4 shows the augmented graph for this instance.

Lemma 28 *In the instance depicted in Figure 4-3 the maximum achievable rate with network coding is $2/3$.*

Proof. We use the following three downstreamness relationships:

$$\begin{aligned} \{S(a), e_1\} &\rightsquigarrow \{S(a), T(b), e_1\} \\ \{S(b), e_2\} &\rightsquigarrow \{T(a), S(b), e_2\} \\ \{S(a), S(b), e_1, e_2\} &\rightsquigarrow \{S(a), S(b), T(c), e_1, e_2\} \end{aligned}$$

The first downstreamness relationship implies

$$\begin{aligned} H(S(a), e_1) &\geq H(S(a), T(b), e_1) \\ &\geq H(S(a), S(b), e_1) \end{aligned}$$

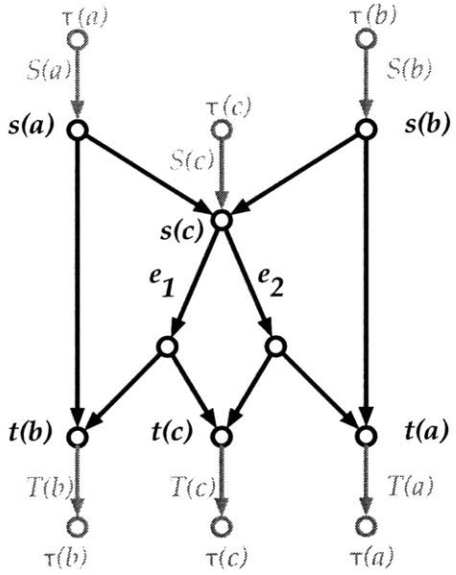


Figure 4-4: The augmented graph for the instance in Figure 4-3. The edges in gray are the edges added in the augmentation process. The meagerness of this instance is 1 but the maximum rate at which all three sources can communicate with their respective sinks is $2/3$.

where the second inequality follows by Lemma 24. Similarly,

$$H(S(b), e_2) \geq H(S(a), S(b), e_2)$$

Adding these two inequalities together we get

$$H(S(a), e_1) + H(S(b), e_2) \geq H(S(a), S(b), e_1) + H(S(a), S(b), e_2)$$

We combine the two terms on the right side using the submodularity of entropy.

$$H(S(a), e_1) + H(S(b), e_2) \geq H(S(a), S(b), e_1, e_2) + H(S(a), S(b))$$

On the left side, we can upper bound $H(S(a), e_1)$ by $H(S(a)) + H(e_1)$ and likewise for $H(S(b), e_2)$.

$$H(S(a)) + H(S(b)) + H(e_1) + H(e_2) \geq H(S(a), S(b), e_1, e_2) + H(S(a), S(b))$$

Using the fact that the sources are independent, we can cancel $H(S(a)) + H(S(b))$ on the left with $H(S(a), S(b))$ on the right.

$$H(e_1) + H(e_2) \geq H(S(a), S(b), e_1, e_2)$$

By the third downstreamness relationship,

$$H(S(a), S(b), e_1, e_2) \geq H(S(a), S(b), T(c), e_1, e_2)$$

Using this inequality and replacing $T(c)$ with $S(c)$,

$$H(e_1) + H(e_2) \geq H(S(a), S(b), S(c), e_1, e_2)$$

The term on the right contains all sources in the instances. Therefore,

$$\begin{aligned} H(e_1) + H(e_2) &\geq H(S(a), S(b), S(c)) \\ &\geq H(S(a)) + H(S(b)) + H(S(c)) \end{aligned}$$

where the second inequality follows because the sources are independent. Finally, we apply Lemma 25 to the last inequality. If rate r is achievable, then there exists a constant b such that

$$\begin{aligned} (c(e_1) + c(e_2)) \log b &\geq (d_a + d_b + d_c) r \log b \\ (c(e_1) + c(e_2)) &\geq (d_a + d_b + d_c) r \\ \frac{2}{3} &\geq r \end{aligned}$$

The last inequality follows because all edges have capacity 1 and all commodities have demand 1. □

4.7 Entropy-based axioms

Examining the proof of Lemma 28 carefully, it uses only the following entropy based axioms.

Submodularity of entropy: H is a non-negative, non-decreasing, submodular set function.

Downstreamness: If $A \rightsquigarrow B$, then $H(A) \geq H(B)$.

Independence of sources: For any set $S(i_1), S(i_2), \ldots, S(i_j)$ of sources,

$$H(S(i_1), \ldots, S(i_j)) = H(S(i_1)) + \ldots + H(S(i_j))$$

Correctness: The random variables $Y_{S(i)}$ and $Y_{T(i)}$ are equal. Consequently, for any set of generalized edges U, $H(U \cup \{S(i)\}) = H(U \cup \{T(i)\})$.

Rate: The entropy of a random variable is maximized when the random variable is distributed uniformly over its sample space. Therefore

$$H(e) \leq c(e) \log b$$

and

$$H(S(i)) \geq r d_i \log b$$

in a rate r solution with constant b.

For any instance of a network coding problem there is a finite (but exponentially large) set of constraints which can be derived from these axioms. All of these constraints are linear, which leads to the following linear program for computing an upper bound on the maximum rate r achievable via network coding. We ignore the constant $\log b$ which arises in the rate constraint.

LP - Acyclic

$$\begin{array}{lllll}
\max & r & & & \\
\text{s.t.} & H(U) & \geq 0 & & (\forall U \subseteq \hat{E}) \\
& H(U \cup \{e\}) & \geq H(U) & & (\forall U \subseteq \hat{E}, e \in \hat{E}) \\
& H(U) + H(W) & \geq H(U \cup W) + H(U \cap W) & & (\forall U, W \subseteq \hat{E}) \\
& H\left(U \cup S(i)\right) & = H\left(U \cup T(i)\right\} & & (\forall\, U \subseteq \hat{E}, \forall i \in \mathcal{I}) \\
& H(U) & \geq H(U') & & (\forall\ U, U' \subseteq \hat{E} : U \rightsquigarrow U') \\
& H\left(e\right) & \leq c(e) & & (\forall\ e \in \hat{E} \backslash \{\mathcal{S} \cup \mathcal{T}\}) \\
& H\left(S(i)\right) & \geq r\, d_i & & (\forall\ S(i) \in \mathcal{S}) \\
& H(\mathcal{S}) & = \sum_{i=1}^{k} H(S(i)) & &
\end{array}$$

For every subset $U \subseteq E$, there is a variable $H(U)$. The intuition is that $H(U)$ represents the joint entropy of the functions associated with edges in U. Note that by interpreting $H(\cdot)$ as the base b entropy function, ignoring the $\log b$ arising in the rate constraints does not effect the admissibility of a rate r solution. The first three constraints ensure that the assignments to variables are non-negative, non-decreasing and submodular. The fourth constraint ensures that a solution to the LP obeys correctness. The fifth constraint ensures that the downstreamness axiom holds. The sixth constraint says that edge e has capacity $c(e)$. The seventh constraint is that every source transmits at rate at least r. The last constraint specifies that the sources are independent.

We put forth the following conjecture for directed acyclic graphs:

Conjecture 29 *The maximum achievable network coding rate is equal to the optimal value of the linear program* **LP-acyclic**.

It was shown by Song, Yeung and Cai in [36, 39] that the maximum rate found by a similar LP is an upper bound on the maximum network coding rate. (See [36] and Chapter 15 of [39]).

4.8 The General Network Coding Problem

Although we described all of our results in terms of the k-pairs communication problem, it is easy to extend them to general network coding problems defined on directed acyclic graphs. In this section we describe an augmented graph for a general network coding problem. The definitions of a network coding solution and rate follow. Downstreamness, as defined for the k-pairs communication problem, also applies to the general network coding problem. Finally we discuss the extensions of Lemma 24 to the general problem and the applicability of Lemmas 25, 26 and 27.

4.9 Definitions

Definition 11 (Augmented Graph $\hat{G}$) *Given a network coding instance on underlying directed acyclic graph G, the* augmented graph $\hat{G} = (\hat{V}, \hat{E})$ *is obtained by applying the following transformation to G.*

- *For each commodity i, we add a new vertex $\sigma(i)$ with one outgoing edge $S(i,v) = (\sigma(i), v)$ for every node $v \in V$ which was a source for commodity i. The set of out-edges from $\sigma(i)$ is denoted $\mathcal{S}(i)$ and the set of all edges $S(i,v)$ for all commodities i is denoted by $\mathcal{S}$.*

- *For each commodity i and each node $v \in V$ that was a sink for i, we add a new vertex $\tau(i,v)$ with one incoming edge $T(i,v) = (v, \tau(i,v))$ and no outgoing edges. Let $\mathcal{T}(i) = \{T(i,v) : v \in V\}$ and $\mathcal{T} = \cup_i \mathcal{T}(i)$.*

A generalized edge *of G is an edge of $\hat{G}$. If $e = (u,v)$ is such an edge, the set of all incoming edges to u will be denoted by* $\mathrm{In}(e)$.

We can use the same definition of a network code given for the k-pairs communication problem but need to redefine a network coding solution.

Definition 12 (Network Coding Solution) *A network code defined on an augmented directed acyclic graph $\hat{G}$ is a* solution *to an instance of the network coding problem if it meets the following conditions. Let M be the $|\mathcal{I}|$-tuple of messages.*

- *For every commodity i, for every edge $S(i,v) \in \mathcal{S}(i)$, $f_{S(i,v)}(M) = M_i$.*
- *For every generalized edge $e \in \hat{E} \setminus \mathcal{S}$, the function $f_e : \prod_i \Sigma(\mathcal{S}) \rightarrow \Sigma(e)$ is computable from the functions on edges in* $\mathrm{In}(e)$.
- *For every commodity i, for every edge $T(v,i) \in \mathcal{T}(i)$, $f_{T(v,i)}(M) = M_i$.*

Notice that this definition of a network coding solution requires that every edge in $\mathcal{S}(i)$ or $\mathcal{T}(i)$ uses the same alphabet and transmits the same message. For commodity i, let $\Sigma(i)$ be this alphabet.

Definition 13 (Rate) *We say a network coding solution defined on the augmented graph $\hat{G}$ achieves rate r if there exists a constant b such that $\log_b |\Sigma(e)| \leq c(e)$ for each $e \in E$, and $\log_b |\Sigma(i)| \geq r d_i$ for each commodity i.*

4.10 Extension of Techniques

Using similar ideas as in Section 4.5, we can define the entropy of a source and the random variable associated with an edge. First we extend Lemma 24 to the general problem.

Lemma 30 *Let A be a set of generalized edges. Given a network coding solution for an instance of the general network coding problem,*

$$H(\mathcal{S}, A) = H(\mathcal{S})$$

For all $i \in \mathcal{I}$, let $B \subset \mathcal{S}(i)$, $C \subset \mathcal{T}(i)$, $S(i,v) \in \mathcal{S}(i)$ and $T(u,i) \in \mathcal{T}(i)$,

$$H(B, A) = H(S(i,v), A) = H(T(u,i), A) = H(C, A)$$

This lemma follows directly from the definition of a network coding solution. Lemmas 25, 26 and 27 did not rely on any special structure of the k-pairs communication problem. Together with Lemma 30 and the properties of the entropy function, these lemmas imply that under the appropriate modifications, the linear program given in the previous section provides an upper bound on the maximum achievable rate for a general network coding problem defined on a directed acyclic graph.

4.11 Open Questions

There is one important open question related to the work in this chapter.

- **For an instance of the network coding problem, what is the maximum achievable rate?** Is the optimal value of the linear program in Section 4.7 equal to the maximum achievable network coding rate? A similar question was raised in [39, 36]. This linear program can be thought of as combining some conditions that the joint entropies of subsets of a set of random variables must satisfy with some conditions that the structure of the graph impose. The conditions that specify that entropy is a submodular, non-negative and non-decreasing function are known as *Shannon type information inequalities*. Recently, what are known as *non-Shannon type information inequalities* have been found [30, 39]. These are not fully understood and may be required to correctly characterize the maximum achievable rate.

4.12 References

The results in this chapter are joint work with Nicholas Harvey and Robert Kleinberg.

For the multicast problem, necessary and sufficient conditions for the existence of a rate r solution were given by Ahlswede et al. [1]. The example in Figure 4-2 is due to Nicholas Harvey and was independently discovered by Li and Li [25].

The linear program in Section 4.7 is similar to an upper bound presented by Song, Yeung and Cai [36]. The actual linear programming formulation is presented in Chapter 15 of [39].

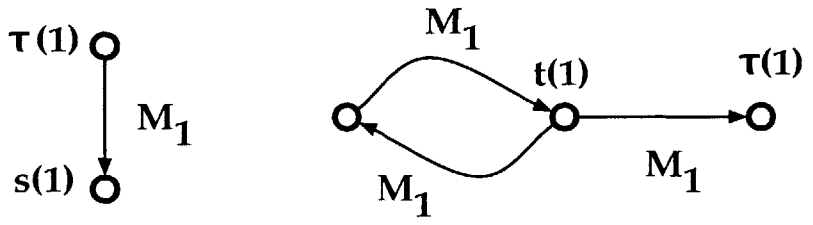
Chapter 5

Graphs with Cycles

How should one define network coding solutions in undirected graphs, or in graphs with cycles? In Chapter 4 we define a network code to be a solution only if the following condition is satisfied:

- The function $f_e : \Sigma(\mathcal{S}) \rightarrow \Sigma(e)$ is computable from the functions on edges in $\mathrm{In}(e)$.

In graphs with cycles, using this local condition is insufficient. Consider the following graph. The label next to each edge specifies the symbols transmitted over that edge.



Clearly, this shouldn't be allowed as a network coding solution. We consider three methods for defining a network coding solution that avoid these foundational problems. We show that all three are equivalent. Once a sound model has been established, we extend the techniques from the previous chapter to graphs with cycles. We define and prove all our results in terms of the k-pairs communication problem. The results can be extended to the general network coding problem using ideas similar to those presented at the end of Chapter 4.

For consistency, all three models assume that $\hat{G}$, the graph obtained by augmenting the underlying graph, is a directed graph. We extend the definition of the augmented graph $\hat{G}$ to instances in which the underlying graph G is undirected. Recall that our definition of a k-pair communication instance in Chapter 1 was not specific to any type of graph.

Definition 14 (Augmented Graph $\hat{G}$) *Given a k-pairs communication problem on underlying graph G, the* augmented graph $\hat{G} = (\hat{V}, \hat{E})$ *is obtained by applying the following transformation to G.*

- *If G is undirected, replace each undirected edge $\{u, v\}$ with two oppositely directed edges (u, v) and (v, u).*
- *For each commodity i, we add a new vertex $\sigma(i)$ with one outgoing edge $S(i) = (\sigma(i), s(i))$ and no incoming edges. The set of all edges $S(i)$ is denoted by $\mathcal{S}$. We also add a new vertex $\tau(i)$ with one incoming edge $T(i) = (t(i), \tau(i))$ and no outgoing edges. The set of all edges $T(i)$ is denoted by $\mathcal{T}$.*

A generalized edge *of G is an edge of $\hat{G}$. If $e = (u, v)$ is such an edge, the set of all incoming edges to u will be denoted by* $\mathrm{In}(e)$.

5.1 Models for Graphs with Cycles

The first model we discuss was introduced by Koetter and Médard[19] and uses polynomials in a formal variable to represent the messages transmitted across an edge in the graph. The second model we describe is a straight-line program formulation. Each line in the straight-line program is required to be computable from only the previous lines in the program. Multiple lines in the straight-line program can specify information to be transmitted over the same edge so long as the total capacity of the edge is not exceeded. The last model expands the underlying graph over t time steps. The functions on edges between layers i and $i + 1$ represent the information transmitted throughout the graph at the ith time-step.

5.1.1 Delay Variables

Koetter and Médard[19] considered network coding in a directed graph with cycles. Their approach was to use a formal variable D and to associate with each directed edge $e \in \hat{E}$ a polynomial in D:

$$g_e(D) = \sum_{j=0}^{\infty} a_{e,j} D^j$$

Thus, the coefficient of D^j in the polynomial $g_e(D)$ represents the information transmitted across edge e at time j. The coefficients $a_{e,j}$ are actually functions of the k-tuple of messages transmitted from the sources. Therefore

$$g_e(D, M) = \sum_{j=0}^{\infty} a_{e,j}(M) D^j$$

When we consider undirected graphs, we allow the designer of the code to split the capacity between a forward and backward edge. We do not require that this split be done in the same way at every time step. To allow for this flexibility, the image of the function $a_{e,j}$ is a time dependent alphabet $\Sigma(e, j)$.

Definition 15 (Network Code) *Given an instance of the network coding problem on underlying graph G, a network code is specified a maximum degree t and by defining for each edge $e \in \hat{E}$ and time $0 \leq j \leq t$ the following:*

- *An alphabet $\Sigma(e, j)$.*
- *A function $a_{e,j} : \Sigma(\mathcal{S}) \rightarrow \Sigma(e, j)$.*

The function $g(D, M)$ is no longer a polynomial because the image of $a_{e,j}$ is not the same for all j. For convenience, we still refer to $g(D, M)$ as a polynomial in D.

Without loss of generality, we alter Koetter and Médard's model by requiring that the polynomials in D be of degree at most $t - 1$ for $e \notin \mathcal{T}$ and degree t for $e \in \mathcal{T}$.

Definition 16 (Delay Variable Network Coding Solution) *We say a network code defined in the delay variable model is a* solution *if*

- *For all $S(i) \in \mathcal{S}$, $g_{S(i)}(D, M) = M_i$*
- *For all $e \in \hat{E} \setminus \mathcal{S}$ and $j \geq 0$, $a_{e,j}$ is computable from the coefficients of terms of degree less than j in the polynomials associated with edges in $\text{In}(e)$.*
- *For all $T(i) \in \mathcal{T}$, $g_{T(i)}(D, M) = M_i D^t$*

Koetter and Médard were able to make elegant use of this formulation. In particular, using other algebraic techniques they were able to extend their results for multicast coding to directed graphs with cycles. We will show that the next two models are equivalent to the delay variable model. The graph over time model is extremely cumbersome to work with but the natural model in which to extend our techniques from directed acyclic graphs to graphs with cycles. Given the equivalence of the three models, once we have extended our techniques to graphs with cycles using the graph over time model, it is possible to use any of the three for proving new results.

5.2 Straight-Line Program

We now describe a straight-line program formulation of a network code. Let $G = (V, E)$ be a graph and $\hat{G}$ be the augmentation of G. The information transmitted on each edge in G is determined by a finite set of rules. Each rule is associated with an edge $(u, v) \in \hat{E}$. A

rule maps the k-tuple of messages to some information to be transmitted across edge e. We denote a rule by $\texttt{rule}_{e,j}$ to indicate it is the jth rule and is associated with edge e. If a rule in the straight-line program specifies information to be transmitted on directed edge e, then this information should be computable from the preceding rules for edges in $\text{In}(e)$. The one exception is out-edges from sources. We require that there is exactly one rule associated with an out-edge of a source and this rule specifies that the edge transmits the message for that commodity (i.e. $\texttt{rule}_{S(i),j} = M_i$).

A network code defined by a straight-line program is a *solution* if all in-edges to sinks for commodity i transmit the message M_i. Without loss of generality we require that there is only one rule for an in-edge to a sink.

5.2.1 Time-Expanded Graph

Our third method for defining a network coding solution in a graph with cycles is to insist that it must be possible to implement such a network coding solution over time. This notion is made precise below by specifying a leveled directed acyclic graph G_t representing the flow of information in G over a sequence of t time steps, and requiring that a network coding solution in G should come from a network coding solution in G_t. A similar approach was advocated in [11], but that paper proposed a "memoryless" model of network coding based on a slightly different definition of G_t. Here, in contrast, we advocate a model which implicitly treats each node of G as having infinite memory.

Definition 17 (Time Expanded Graph G_t) *Given an instance of the network coding problem in a directed graph G, the* time-expanded graph $G_t = (V_t, A_t)$ *is a directed acyclic graph obtained using the following transformation.*

- *The vertex set V_t includes the set $V \times \{0, 1, 2, \ldots, t\}$. The vertex of V_t represented by the ordered pair (v, s) will be denoted by v_s.*
- *For each edge $e = (u, v) \in E$ there are t edges (u_{s-1}, v_s). Let $e_s = (u_{s-1}, v_s)$ for $1 \leq s \leq t$.*
- *For each $v \in V$ and each $s \in \{0, 1, 2, \ldots, t\}$, there is an edge $(v_{s-1}, v_s) \in E$. These edges are referred to as* memory edges.
- *For each commodity i, we add a new source vertex $\sigma(i)$ and we add the directed edge $S(i) = (\sigma(i), s(i)_0)$ to E_t*
- *Similarly, for each commodity i, we add a new sink $\tau(i)$ to V_T and a directed edge $T(i) = (t(i)_t, \tau(i))$ to E_t.*

If a network coding instance is defined on an undirected graph G, then the graph G_t is obtained from G by first replacing each undirected edge of G by two oppositely directed edges and then performing the above transformation.

Since G_t is a directed acyclic graph, the definition of a network code and network coding solution from Chapter 4 are still appropriate.

5.2.2 Equivalence of Models

We show these three models are equivalent by describing a mapping between network codes defined in the three models. We start with a code specified in the delay variable formulation and map it into a code specified in the time-expanded graph model. We then map a code in the time-expanded graph model to a straight-line program formulation for a code. Finally we map a code defined by a straight-line program into a code defined using a formal delay variable.

Lemma 31 *Given an augmented graph $\hat{G}$ for an instance of the network coding problem and a network coding solution specified with edge polynomials $\overline{g} = \{g_e(D, M) : e \in \hat{E}\}$, there exists a network code solution specified with a time-expanded graph G_t and edge functions $\overline{f} = \{f_e(M) : e \in E_t\}$.*

Proof. Suppose we are given a network code specified in the delay variable formulation and let t be the maximum degree of the polynomials for edges e such that $e \notin \mathcal{T}$. If $e \in \mathcal{T}$ then the degree of g_e is $t+1$. We map this network code to edge functions in the time-expanded graph G_t. We describe this mapping for each type of edge in E_t.

Source edges: In both models the functions transmitted on an out-edge $S(i)$ from the source for commodity i must be equal to M_i. Therefore, in the delay variable model $g_{S(i)}(D, M) = M_i$ and in the graph over time model we let $f_{S(i)}(M) = M_i$.

Edges of G: Let e be an edge in $\hat{e}$ that is not an out-edge from a source or an in-edge to a sink. In G_t the edges $\{e_1, e_2 \ldots e_t\}$ are associated with edge e. In the delay variable model the polynomial $g_e(D, M) = \sum_{j=0}^{t} a_{e,j}(M)D^j$ specifies the information transmitted on edge e over time. By definition, $a_{e,0} = 0$. For $j \geq 1$, let $f_{e_j} = a_{e,j}$.

Memory edges: If $e = (v_s, v_{s+1})$ is a memory edge in G_t then it has infinite capacity in the time-expanded graph model. In this case we use the function assigned to edge e to "hold" all the information received at node v in previous time steps. Specifically, if $\mathrm{In}(e)$ is the set of in-edges to e in E_t, then $f_e = f_{\mathrm{In}(e)}$.

Sink edges: If $e = T(i)$ is an in-edge to a sink $\tau(i)$ then the delay variable formulation requires $g_e(D, M) = M_i D^{t+1}$ and is computable from the functions for coefficients of terms of degree at most t in the polynomials on in-edges to e. We assign the function $f_{T(i)}(M) = M_i$.

We now show that the network code specified for G_t is a network coding solution. First, an edge $S(i)$ transmits the message M_i. Second, by definition, a memory edge e just copies

all the information transmitted on in-edges $\text{In}(e)$ and therefore its function is computable from the functions on $\text{In}(e)$.

Let $\tilde{E} = \{e \in \hat{E} : e \notin \mathcal{S} \cup \mathcal{T}\}$. We prove by induction on j that $f_{(e_j)}$ is computable from the functions in $\text{In}(e_j)$ for $e \in \tilde{E}$. The edge e_1 computes the function $f_{e_1} = a_{e,1}$. By definition, this function must be computable from the coefficients of terms of degree 0 in the polynomials on edges in $\text{In}(e)$. The only polynomials with non-zero coefficients for D^0 are the polynomials associated with out-edges from sources. Therefore, $a_{e,1}$ must be computable from the in-edges to e that are out-edges from sources. If $S(i) \in \text{In}(e)$ in $\hat{G}$, then by the definition of G_t, $S(i) \in \text{In}(e_1)$. Therefore, if $a_{e,1}$ is computable then f_{e_1} is computable.

Now assume for all $e \in \tilde{E}$ and all $j' < j$, $f_{e_{j'}} = a_{e,j'}$ is computable from the edges in $\text{In}(e_{j'})$. We need to show that the function $f_{e_j} = a_{e,j}$ is computable from the functions on edges in $\text{In}(e_j)$. Let $e_j = (u_{j-1}, v_j)$. The function associated with the memory edge (u_{j-2}, u_{j-1}) computes the value of all functions associated with terms of degree less than $j-1$ in the polynomials on edges in $\text{In}(e)$. If $e' \in \text{In}(e)$, then $f_{e'_{j-1}} = a_{e',j-1}$. Therefore the functions associated with edges in $\text{In}(e_j)$ compute the values of all the coefficients of terms of degree at most $j-1$ in polynomials associated with edges in $\text{In}(e)$ in the delay variable formulation. Since $a_{e,j}$ must be computable from these coefficients, f_{e_j} is computable from the functions on edges in $\text{In}(e_j)$. By a similar argument, for each commodity i, the function $f_{T(i)}$ is computable from edges in $\text{In}(i)$. □

Having mapped network coding solutions specified using delay variables to network coding solutions in the time-expanded graph model we now map solutions in the time-expanded graph model to straight-line program formulations.

Lemma 32 *Given an augmented graph $\hat{G}$ for an instance of the network coding problem and a network coding solution specified on a time-expanded graph G_t with edge functions $\overline{f} = \{f_e(M) : e \in E_t\}$, there exists a network coding solution specified with a straight-line program and rules $\overline{\texttt{rule}} = \{\texttt{rule}_{e,i} : e \in \hat{E}\}$.*

Proof. Suppose we are given a network coding solution specified in a time-expanded graph G_t. We map this network code to rules in a straight-line program. First, in both models the functions transmitted on an out-edge from a source for commodity i must be equal to M_i. Therefore, the first set of rules are

$$\text{for } 1 \leq i \leq k \qquad \texttt{rule}_{S(i),i} : \text{ Transmit } M_i$$

For the remaining rules, choose a topological ordering of the edges in E_t that are not out-edges of sources or in-edges to sinks. We consider each edge in turn according to this ordering. If edge e is a memory edge then we do nothing. Otherwise, edge e_j is associated with an edge $e \in \hat{E}$. Let $f_{e,j}(M)$ be the function associated with e_j. We create a rule $\texttt{rule}_{e,h(e,j)}(M)$ which transmits the same function as $f_{e,j}(M)$.

$$\texttt{rule}_{e,h(e,j)} : \text{Transmit } f_{e,j}(M)$$

The function $h(e,j) = k + \text{order}(e_j)$ where order(e) is the position of edge e in the chosen topological ordering.

Finally, for each in-edge to a sink a network coding solution in the time-expanded graph model requires that $f_{T(i)}(M) = M_i$. Similarly, in the straight-line program model the single rule associated with $T(i)$ must specify M_i.

$$\texttt{rule}_{T(i),\ell(i)} : \text{Transmit } f_{t(i)}(M)$$

where $\ell(i) = k + t * |E| + i$.

We now show that each rule for an edge e is computable from the rules preceding it associated with edges in $\text{In}(e)$. Let $e = (u, v)$ be an edge of E and let $\{e_1, \ldots e_t\}$ be the t edges in G_t associated with e. By the definition of a network code in a time-expanded graph G_t, $f_{e_s}(M)$ must be computable from the edge functions for $\text{In}(e_s)$ for all s. Included in $\text{In}(e_s)$ is a memory edge (u_{s-1}, u_s). The function on this memory edge is computable from the information received at u at prior time steps. Therefore, once we have specified a rule in the straight-line program for all non-memory edges in $\text{In}(e_j)$ for all $j < s$, the rule $\texttt{rule}_{e,h(e,s)}(M) = f_{e,s}(M)$ is guaranteed to be computable from the preceding rules. A similar argument shows that the rule for an edge $T(i)$ is computable from the previous rules for edges in $\text{In}(T(i))$. □

Finally, we demonstrate how to transform a straight-line program formulation for a network coding solution into a delay variable formulation.

Lemma 33 *Given an augmented graph $\hat{G}$ for an instance of the network coding problem and a network coding solution specified with a straight-line program and rules $\overline{\texttt{rule}} = \{\texttt{rule}_{e,j} : e \in \hat{E}\}$ there exists a network coding solution specified with edge polynomials $\overline{g} = \{g_e(D, M) : e \in \hat{E}\}$.*

Proof. Suppose we are given a network coding solution specified by a straight-line program. We map this network code to polynomials in a delay variable D. First, in both models the functions transmitted on an out-edge from a source for commodity i must be equal to M_i. If $S(i)$ is an out-edge from a source, then $g_{S(i)}(D, M) = M_i$.

Let $\tilde{E} = \{e \in \hat{E} : e \notin \mathcal{S} \cup \mathcal{T}\}$. For each edge $e \in \tilde{E}$, we create the polynomial $g_e(D, M) = \sum_{j=0}^{t} a_{e,j} D^j$ term by term. For $j = 0$, for all edges $e \in \tilde{E}$, $a_{e,0} = 0$. Assume that we have determined $a_{e,j'}$ for all $j' < j$ and all $e \in \tilde{E}$. For an edge $e \in \tilde{E}$, let $\{\texttt{rule}_{e,i_1}, \texttt{rule}_{e,i_2} \ldots \texttt{rule}_{e,i_k}$ be the rules in the straight-line program that have not been mapped to coefficients and are computable from rules in the straight-line program that have been mapped to coefficients. Let

$$a_{e,j}(M) = (\texttt{rule}_{e,i_1}(M), \texttt{rule}_{e,i_2}(M) \ldots \texttt{rule}_{e,i_k}(M))$$

Since $\{\texttt{rule}_{e,i_1}, \texttt{rule}_{e,i_2} \ldots \texttt{rule}_{e,i_k}\}$ are rules that are computable from rules we already mapped to coefficients, $a_{e,j}(M)$ is computable from the coefficients of terms of degree less than j in the polynomials associated with edges in $\text{In}(e)$. Since each rule in the straight-line program is definable from preceding rules, as long as there are rules in the straight-line program that have not been mapped to coefficients, we will be able to find a new rule which can be mapped to a coefficient. Since there are a finite number of rules, this process will terminate. Assume that the maximum degree of a polynomial associated with an edge in $\tilde{E}$ is t. By the definition of a network coding solution in the straight-line program model, for each in-edge to a sink there is a rule which specifies that the edge transmits the message M_i. In this special case, we map this rule to the polynomial $g_{T(i)}(D, M) = M_i D^{t+1}$. □

These three Lemmas show that all three models of network coding in graphs with cycles are equivalent and hence prove the following theorem.

Theorem 34 *The delay variable, straight-line program and time-expanded graph models of network coding are equivalent.*

These three models resolve the foundational issues of defining a network coding solution in a graph with cycles. By construction, a network code in any of these three models specifies the function to be transmitted across an edge for every k-tuple of messages. However, it is often easier to reason about the augmented graph $\hat{G}$ directly. For this purpose it is convenient to have a notion of a network coding solution defined on $\hat{G}$.

Definition 18 (Consise Network Coding Solution) *Given an instance of the network coding problem and a solution defined in the time-expanded graph model, we define a* concise *representation of this solution as follows.*

- *For $e \in \mathcal{S} \cup \mathcal{T}$, let $\Sigma(e)$ and f_e be equal to the edge alphabet and function defined for the edge $e \in G_t$.*
- *For $e \in \hat{E} \setminus \{\mathcal{S} \cup \mathcal{T}\}$, let $\{e_1, e_2 \ldots e_t\}$ in G_t be the edges associated with e and $f_{e_1}, f_{e_2} \ldots f_{e_t}$ be the functions assigned to these edges in the network coding solution.*
  - *The* edge alphabet *is defined as $\Sigma(e) = \prod_j \Sigma(e_j)$.*
  - *The* edge function *is defined as $f_e : \Sigma(\mathcal{S}) \rightarrow \Sigma(e)$*

$$f_e(M) = (f_{e_1}(M), f_{e_2}(M), \ldots f_{e_t}(M))$$

Although we defined a concise network coding solution in terms of a time-expanded graph, the mappings given in this section preserve $\Sigma(e)$ and f_e. Therefore, starting from any definition of a network coding solution, the above transformation will produce the same concise representation.

5.3 Rate of a Network Coding Solution

Using the concise representation of a network coding solution, we define the rate of a solution.

Definition 19 (Rate(Directed)) *Given an instance of the network coding problem with underlying directed graph G, we say a network coding solution achieves rate r if there exists a constant b such that $\log_b |\Sigma(e)| \leq c(e)$ for each $e \in E$, and $\log_b |\Sigma(S(i))| \geq r d_i$ for each commodity i.*

Definition 20 (Rate(Undirected)) *Given an instance of the network coding problem with underlying undirected graph G, we say a network coding solution achieves rate r if there exists a constant b such that*

- *for each edge $e \in E$ represented by oppositely directed edges $\vec{e}$ and $\overleftarrow{e}$ in $\hat{G}$,*

$$\log_b \left(|\Sigma(\vec{e})| \cdot |\Sigma(\overleftarrow{e})| \right) \leq c(e)$$

- *and for each commodity i,*

$$\log_b |\Sigma(S(i))| \geq r d_i$$

Our definition of a rate r code, allows the code designer to split the use of an undirected edge in any possible way. It should be noted that there could be other interpretations for an undirected edge. For example, one could assume that an undirected edge (u, v) with capacity c can be used, at each time step, as either a capacity c channel from u to v or as a capacity c channel from v to u. Our definition is more general than this. Since we are interested in upper bounding the maximum achievable rate, we choose this more general definition. Another possibility would be to interpret this edge as allowing a channel of capacity c from u to v *and* a channel of capacity c from v to u. This at most doubles the maximum achievable rate and would be more restrictive than just using our definition with doubled edge capacities.

5.4 Entropy Interpretation of Rate

When bounding the rate of a network coding solution, we use the entropy-based view of network coding defined in Chapter 4. Recall that we defined the entropy of random variables associated with generalized edges. For a graph with cycles, we define the entropy of an edge in terms of concise network coding solutions. In particular, for edge $e \in \hat{E}$, let Y_e be the random variable such that $\Pr(Y_e = \alpha) = \Pr(f_e(M) = \alpha)$ for all $\alpha \in \Sigma(e)$. As in Chapter 4 we use $H(e)$ as short-hand for $H(Y_e)$.

Lemma 35 *Given a concise network coding solution defined on an augmented graph $\hat{G}$ of rate r, there exists a constant b such that the following hold. For all edges $e \in E$,*

$$H(e) \leq c(e) \log_2 b$$

and for all commodities $i \in \mathcal{I}$,

$$H(S(i)) \geq r d_i \log_2 b$$

The proof of this result closely mimics the proof of Lemma 25 in Chapter 4 and is therefore omitted. Using similar techniques, we can also prove the following lemma for undirected graphs.

Lemma 36 *Let G be an undirected graph and let $\hat{G}$ be the augmented graph. Given a concise network coding solution of rate r, there exists a constant b such that the following hold. For any pair $\vec{e}$ and $\overleftarrow{e}$ of edges corresponding to an undirected edge e,*

$$H(\vec{e}) + H(\overleftarrow{e}) \leq c(e) \log_2 b$$

and for any edge $S(i)$

$$H(S(i)) \geq r d_i \log_2 b$$

5.5 Extending Techniques to Graphs with Cycles

We can also extend our techniques from the Chapter 4 to graphs with cycles. We start by extending the concept of downstreamness to sets of generalized edges in a graph with cycles.

Lemma 37 *Let A, B be sets of generalized edges in G, and let A_t, B_t be the corresponding sets of generalized edges in G_t. If $A \rightsquigarrow B$ then $A_t \rightsquigarrow B_t$.*

Proof. Let p be a path in G_t which intersects B_t and $\mathcal{S}$. Deleting the memory edges from p and mapping each remaining edge to the corresponding edge of G, we obtain a path p' which intersects B and $\mathcal{S}$. Since $A \rightsquigarrow B$, it follows that p' intersects A. Thus p intersects A_t. □

Corollary 38 *Given a concise network coding solution in augmented graph $\hat{G}$ and sets of generalized edges A, B such that $A \rightsquigarrow B$, there exists a function $g_{A,B} : \Sigma(A) \to \Sigma(B)$ such that $f_B = g_{A,B} \circ f_A$.*

Corollary 39 *Given a concise network coding solution in augmented graph $\hat{G}$, for any pair of generalized edge sets A, B such that $A \rightsquigarrow B$, we have $H(A) \geq H(B)$.*

Note that Corollary 39 ensures that the downstreamness axiom specified in Chapter 4 is still satisfied in the context of concise network codes in general graphs. It is trivial to verify that all of the other axioms in that section also remain valid in general graphs. We restate them here in terms of general graphs.

Submodularity of entropy: H is a non-negative, non-decreasing, submodular set function.

Downstreamness: If $A \rightsquigarrow B$, then $H(A) \geq H(B)$.

Independence of sources: For any set $S(i_1), S(i_2), \ldots, S(i_j)$ of sources,

$$H(S(i_1), \ldots, S(i_j)) = H(S(i_1)) + \ldots + H(S(i_j)).$$

Correctness: The random variables $Y_{S(i)}$ and $Y_{T(i)}$ are equal. Consequently, for any set of generalized edges U, $H(U \cup \{S(i)\}) = H(U \cup \{T(i)\})$.

Rate: The entropy of a random variable is maximized when the random variable is distributed uniformly over its sample space. Therefore in an instance defined on a directed graph, $H(e) \leq c(e) \log b$ and $H(S(i)) \geq r d_i \log b$ in a rate r solution with constant b. In an instance define on an undirected graph, $H(\overrightarrow{e}) + H(\overleftarrow{e}) \leq c(e) \log b$ and $H(S(i)) \geq r d_i \log b$ in a rate r solution with constant b.

These axioms constitute a powerful technique for proving upper bounds on the rate achievable by network coding in general graphs. To illustrate, we consider the k-pairs communication problem on a directed cycle.

5.5.1 Directed cycles

The maximum flow is equal to the sparsest edge cut in a directed cycle. Let r be the rate of the maximum multicommodity flow. We can use the axioms given above to show that the maximum achievable rate with coding is also r.

We begin by finding the sparsest edge cut. For the directed cycle, the sparsest edge cut contains a single edge. For a single edge u, the sparsity of the edge is given by the number of commodities whose shortest path from $s(i)$ to $t(i)$ on the cycle use u. For edge u, let $\mathcal{I}(u)$ be the set of commodities whose shortest path from $s(i)$ to $t(i)$ uses edge u. Let e be an edge such that

$$\frac{c(e)}{\sum_{i \in \mathcal{I}(e)} d_i} = r$$

In other words, the edge cut consisting of e is a sparse cut. Now we use downstreamness to show that $H(e) \geq \sum_{i \in \mathcal{I}(e)} H(S(i))$. We index the commodities in $\mathcal{I}(e) = \{i_1, i_2 \ldots i_q\}$ according to the order in which their sink is encountered on the directed cycle starting at the head of edge e.

Lemma 40 *For all $1 \leq j \leq q$, starting at the head of e and traversing around the directed ring, $t(i_{j'})$ appears strictly before $s(i_j)$ for $j' \leq j$.*

Proof. For the purposes of obtaining a contradiction, assume that traversing around the directed cycle starting at the head of e, $s(i_j)$ is encountered before $t(i_{j'})$ and $j' \leq j$. Since $i_j \in \mathcal{I}(e)$, the shortest path from $s(i_j)$ to $t(i_j)$ traverses edge e. Therefore $t(i_j)$ is encountered before $s(i_j)$ and therefore also before $t(i_{j'})$. This contradicts the definition of the indexing of the commodities. □

Let $\mathcal{S}(\overline{\mathcal{I}(e)}) = \{S(i) : i \notin \mathcal{I}(e)\}$. Using induction on j, we will show that

$$H(e, \mathcal{S}(\overline{\mathcal{I}(e)})) \geq H(e, \mathcal{S}(\overline{\mathcal{I}(e)}, S(i_1), S(i_2), \ldots S(i_j))$$

For the base case, $j = 1$. By the above lemma, starting at the head of e, all nodes $s(i_j)$ for $i_j \in \mathcal{I}(e)$ are encountered after $t(i_1)$. Therefore the only path from an edge $S(i_j)$ to $T(i_1)$ is through e. This implies the following downstreamness relationship

$$\{e, \mathcal{S}(\overline{\mathcal{I}(e)})\} \rightsquigarrow \{e, \mathcal{S}(\overline{\mathcal{I}(e)}), T(i_1)$$

By the downstreamness and correctness axioms, $H(e, \mathcal{S}(\overline{\mathcal{I}(e)})) \geq H(e, \mathcal{S}(\overline{\mathcal{I}(e)}), S(i_1))$.

Now suppose that

$$H(e, \mathcal{S}(\overline{\mathcal{I}(e)})) \geq H(e, \mathcal{S}(\overline{\mathcal{I}(e)}, S(i_1), S(i_2), \ldots S(i_{j'}))$$

for all $j' < j$. By Lemma 40, for all $p \geq j$ the only path from $S(i_p)$ to $T(i_j)$ traverses edge e. Therefore,

$$\{e, S_{i_1}, S_{i_2} \ldots S_{i_{j-1}}, \mathcal{S}(\overline{\mathcal{I}(e)})\} \rightsquigarrow \{e, S_{i_1}, S_{i_2} \ldots S_{i_{j-1}}, \mathcal{S}(\overline{\mathcal{I}(e)}), T(i_j)\}$$

Combining this with the inductive hypothesis,

$$\begin{aligned} H(e, \mathcal{S}(\overline{\mathcal{I}(e)})) &\geq H(e, \mathcal{S}(\overline{\mathcal{I}(e)}, S(i_1), S(i_2), \ldots S(i_{j-1})) \\ &\geq H(e, \mathcal{S}(\overline{\mathcal{I}(e)}, S(i_1), S(i_2), \ldots S(i_{j-1}), S(i_j)) \end{aligned}$$

Where the second inequality follows by the downstreamness and correctness axioms. Using the submodularity of the entropy function and the Independence of sources,

$$\begin{aligned} H(e) + H(\mathcal{S}(\overline{\mathcal{I}(e)})) &\geq H(\mathcal{S}(\mathcal{I}(e))) + \sum_{i_j \in \mathcal{I}(e)} H(S(i_j)) \\ H(e) &\geq \sum_{i_j \in \mathcal{I}(e)} H(S(i_j)) \end{aligned}$$

If r^* is the rate of this concise solution, then by Lemma 35 there exists a constant b such

$$\begin{aligned} c(e) \log b &\geq H(e) \\ &\geq \sum_{i_j \in \mathcal{I}(e)} H(S(i_j)) \\ &\geq \sum_{i_j \in \mathcal{I}(e)} r^* d_i \log b \end{aligned}$$

Therefore,

$$r^* \leq \frac{c(e)}{\sum_{i_j \in \mathcal{I}(e)} d_i}$$

The above argument proves the following theorem.

Theorem 41 *Let G be a directed n-cycle with vertex set $V = \{v_1, \ldots, v_n\}$. For any k-pairs communication problem in G, the maximum rate achievable by network coding is equal to the maximum flow.*

5.6 Open Questions

In this chapter we defined three equivalent models for network coding in graphs with cycles. There are two open questions related to this work.

- **Are these models general enough?** The time-expanded graph model makes clear that a network coding solution in that model is implementable in a graph with cycles. However, are there network coding solutions which are ruled out by the models in this chapter?

- **Are there other models for network coding in undirected graphs?** When taking an information theoretic view of network coding, we require, roughly, that

$$H\left(\overrightarrow{e}\right) + H\left(\overleftarrow{e}\right) \leq c(e)$$

where the undirected edge e is modeled by two oppositely directed edges $\overrightarrow{e}$ and $\overleftarrow{e}$. What happens if, instead, we require

$$H\left(\overrightarrow{e}, \overleftarrow{e}\right) \leq c(e)?$$

Is this more appropriate in some situations? Another option would be to allow, at each time step, the edge e to be used in one of the two directions at full capacity. What solutions are possible under this model?

5.7 References

The delay variable model for network coding in a directed graph with cycles is due to Koetter and Médard[19]. The time-expanded graph model was sketched by Li, Cai and Yeung[24]. Both of these models were generalized for the purposes of our work.

The other results in this chapter are joint work with Nicholas Harvey, Robert Kleinberg and Eric Lehman.

Chapter 6

k-Pairs Communication in Undirected Graphs

At the end of Chapter 5, we extended our techniques for upper bounding the maximum rate of a solution to graphs with cycles. In this chapter we focus on the k-pairs communication problem in undirected graphs. We begin with some motivation and consider a small graph of interest. We use our techniques from the previous chapter to prove a gap between the sparsest edge-cut in this graph and the maximum rate achievable with network coding. This is the first result of its kind. We then consider a modification to this example that motivates the introduction of a new technique. Using these techniques we prove that the maximum achievable rate can be obtained using multicommodity flow techniques alone for a infinite class of interesting graphs.

6.1 Motivation

Our objective is to understand the relationship between multicommodity flow and network coding in undirected graphs. Network coding in undirected graphs is very different from network coding in directed graphs. To understand this we must review the definition of sparsity of a graph.

Definition 21 (Sparsity) *For a k-pairs communication problem on an undirected graph G, let $\mathcal{I}(A)$ be the set of commodities whose source and sink are in different connected components of $G' = (V, E \setminus A)$. The* sparsity *of a set of edges A is given by*

$$\texttt{sparsity}(A) = \frac{\sum_{e \in A} c(e)}{\sum_{i \in \mathcal{I}(A)} d_i}$$

and the sparsity *of G is equal to* $\min_{A \subset E} \texttt{sparsity}(A)$.

A pigeonhole argument shows that the sparsity of an undirected graph G is an upper bound on the maximum achievable network coding rate. In contrast, we saw in Chapter 4 that the sparsity of a set of edges in a directed graph has no relationship to the maximum achievable rate.

The sparsity of the graph is also an upper bound on the maximum multicommodity flow value. Therefore, in instances in which there exists a multicommodity flow of rate equal to the value of the sparsity of the graph, there is no advantage to using network coding; the maximum rate achievable with network coding is equal to the maximum rate achievable with flow techniques in these instances. The interesting open question is whether the maximum rate achievable with network coding is always equal to the maximum multicommodity flow rate for instances of the k-pairs communication problem on undirected graphs.

We start with two interesting examples in which there is a gap between the value of the sparsest cut and the maximum multicommodity flow rate. These examples lead to an infinite class of graphs for which the maximum achievable rate with coding is equal to the maximum multicommodity flow rate. This class includes an infinite number of instances in which the maximum multicommodity flow rate is less than the sparsity of the graph.

6.1.1 The Okamura-Seymour example

We consider a small example due to Okamura and Seymour[32] in which the maximum multicommodity flow rate is less than the value of the sparsest cut. This example is a 4-commodity flow problem in an undirected graph with 5 vertices. Each source has demand 1 and each edge has capacity 1.

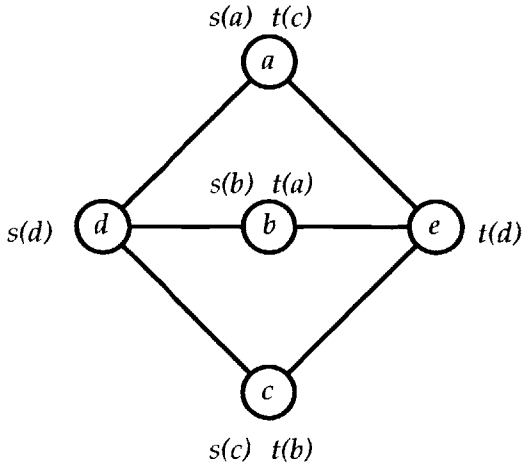


Figure 6-1: The Okamura-Seymour Example.

The maximum concurrent flow in this graph is $3/4$, achieved when each of the commodities a, b, c sends half of its flow on each of the two-hop paths from its source to its sink, and commodity d sends one-third of its flow on each of the two-hop paths from its source to its sink.

In this section we prove that the maximum rate achievable by network coding is also $3/4$. We use the axioms from Chapter 5 to prove that no rate greater than $3/4$ is achievable. The multicommodity flow demonstrates that this rate is achievable.

We consider three different edge cuts and prove an entropy inequality based on each. For each of these edge cuts, we exploit the symmetry of the problem to obtain two analogous inequalities. Combining the resulting nine inequalities together yields the desired bound.

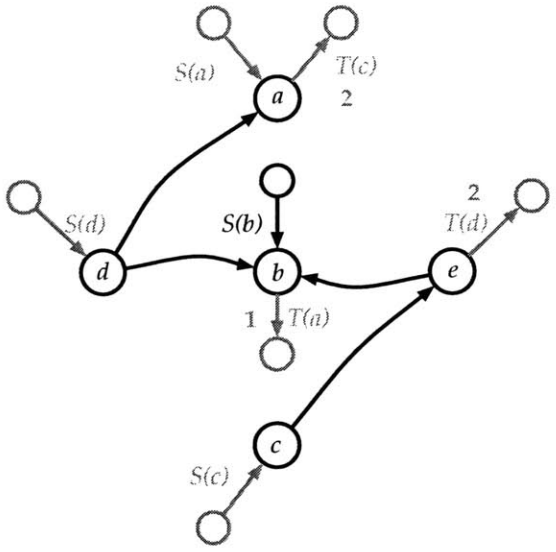


Figure 6-2: The first edge cut consists of the black generalized edges. The gray sinks are the ones that are downstream from the edge cut. The number beside a sink indicates in which downstreamness relationship it is involved.

First consider Figure 6-2. This illustrates the following downstreamness relations, in which we refer to an edge by naming its head and its tail, e.g. ad refers to the edge from a to d:

$$\{da, db, ce, eb, S(b)\} \rightsquigarrow \{da, db, ce, eb, S(b), T(a)\} \quad (1)$$
$$\{da, db, ce, eb, S(b), S(a)\} \rightsquigarrow \{da, db, ce, eb, S(b), S(a), T(c), T(d)\} \quad (2)$$

These in turn imply the following series of entropy inequalities, using the downstream-

ness and correctness axioms.

$$\begin{aligned} H(da, db, ce, eb, S(b)) &\geq H(da, db, ce, eb, S(b), T(a)) \\ &= H(da, db, ce, eb, S(b), S(a)) \\ &\geq H(da, db, ce, eb, S(b), S(a), T(c), T(d)) \\ &= H(da, db, ce, eb, S(b), S(a), S(c), S(d)) \end{aligned} \tag{6.1}$$

The submodularity axiom implies

$$H(da) + H(db) + H(ce) + H(eb) + H(S(b)) \geq H(da, db, ce, eb, S(b))$$

while independence of sources implies

$$H(S(a), S(b), S(c), S(d)) = H(S(a)) + H(S(b)) + H(S(c)) + H(S(d)).$$

Combining these two inequalities with (6.1), and canceling the term $H(S(b))$ on both sides, we obtain

$$H(da) + H(db) + H(ce) + H(eb) \geq H(S(a)) + H(S(c)) + H(S(d)). \tag{6.2}$$

The Okamura-Seymour graph has an automorphism which fixes d and e while cyclically permuting a, b, c, so there are two other such entropy inequalities:

$$H(db) + H(dc) + H(ae) + H(ec) \geq H(S(b)) + H(S(a)) + H(S(d)) \tag{6.3}$$
$$H(dc) + H(da) + H(be) + H(ea) \geq H(S(c)) + H(S(b)) + H(S(d)) \tag{6.4}$$

Let

$$\begin{aligned} D_{in} &= H(ad) + H(bd) + H(cd) \\ D_{out} &= H(da) + H(db) + H(dc) \\ E_{in} &= H(ae) + H(be) + H(ce) \\ E_{out} &= H(ea) + H(eb) + H(ec). \end{aligned}$$

Summing (6.2), (6.3), and (6.4), we obtain

$$2D_{out} + E_{in} + E_{out} \geq 2(H(S(a)) + H(S(b)) + H(S(c))) + 3H(S(d)). \tag{6.5}$$

Figures 6-3 and 6-4 lead, via a similar sequence of steps, to the following inequalities:

$$D_{in} + D_{out} + 2E_{in} \geq 2(H(S(a)) + H(S(b)) + H(S(c))) + 3H(S(d)) \tag{6.6}$$
$$2D_{in} + 2E_{out} \geq 2(H(S(a)) + H(S(b)) + H(S(c))). \tag{6.7}$$

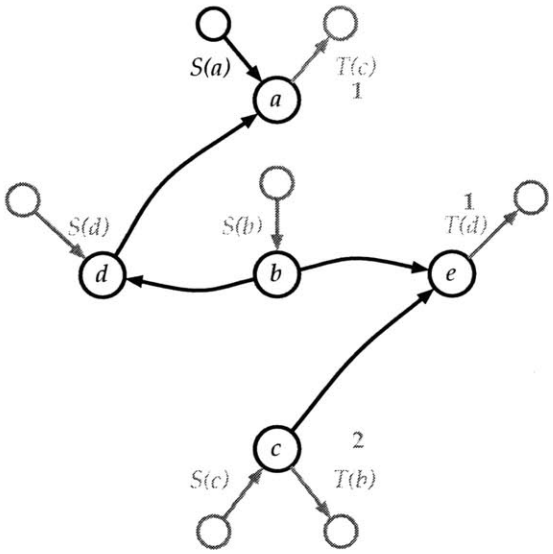


Figure 6-3: The second cut.

Summing (6.5–6.7), we obtain:

$$3(D_{in} + D_{out} + E_{in} + E_{out}) \geq 6(H(S(a)) + H(S(b)) + H(S(c)) + H(S(d))). \quad (6.8)$$

Using Lemma 36 we may interpret equation (6.8) as saying:

$$3 \sum_{e \in E(G)} c(e) \geq 6r \sum_{i \in \mathcal{I}} d_i.$$

Since all 6 edges have capacity 1, and all 4 commodities have demand 1, this reduces to $18 \geq 24r$, or $r \leq 3/4$, as desired.

Lemma 42 *The maximum achievable rate for the k-pairs communication problem depicted in Figure 6-1 is* $3/4$.

In Section 6.4 we will re-derive this result using a different technique.

6.1.2 The 5-commodity Okamura-Seymour example

Suppose that we enhance the Okamura-Seymour example by adding a fifth commodity with source e and sink d. It is easy to check that the maximum concurrent flow now has rate $3/5$. Later, we will present a proof that this is also the maximum rate achievable by a network coding solution. In the following section we develop the additional techniques which are used to prove this upper bound on the maximum network coding rate.

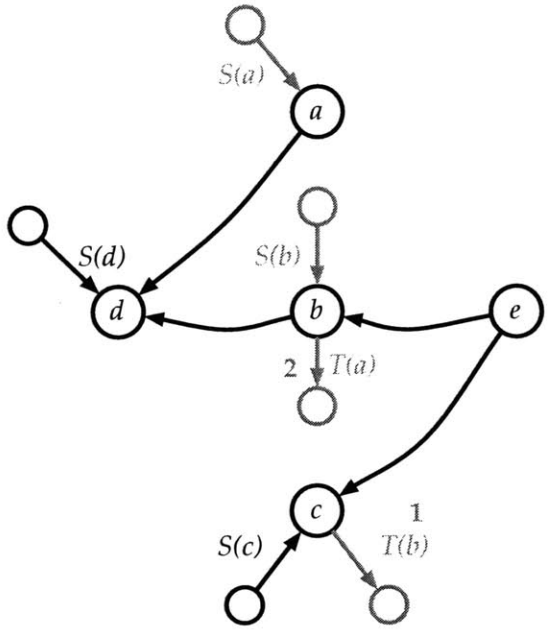


Figure 6-4: The third cut.

6.2 Operational Downstreamness

In this section we present a stronger condition than downstreamness. We first discuss a motivating example. We then define *operational downstreamness* and prove an entropy inequality which holds when a set of edges is operationally downstream from another set of edges. Using this, we reanalyze our motivating example. In the next section we present a combinatorial characterization of operational downstreamness.

6.2.1 A Motivating Example

Suppose we are given a k-pairs communication problem in a graph G. It is tempting to conjecture that, given any function h satisfying the axioms specified in Section 5.5, there exists a concise network coding solution whose entropy function is h. However this is not the case. This is best illustrated with an example. Consider the augmented graph G in Figure 6-5 with edges $(u, v), (v, u), S(a), S(b), T(a)$ and $T(b)$. Let $H(uv)$ and $H(vu)$ be the entropy of the random variable associated with edges (u, v) and (v, u).

We can use the axioms to prove lower bounds on $H(uv)$ and $H(vu)$.

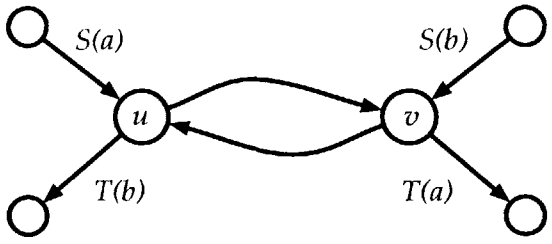


Figure 6-5: An instance of the network coding problem consisting of two vertices u and v and the edges (u, v) and (v, u). Vertex u is the source for a and sink for b. Vertex v is the source for b and the sink for a. The augmented graph is depicted above.

$$
\begin{array}{rcll}
H(uv) & \geq & H(uv, S(b)) - H(S(b)) & \text{submodularity of entropy} \\
& \geq & H(uv, S(b), T(a)) - H(S(b)) & \{(u,v),(S(b),v)\} \rightsquigarrow \{(v,t(a))\} \\
& = & H(uv, S(b), S(a)) - H(S(b)) & \text{correctness} \\
& = & H(S(b), S(a)) - H(S(b)) & \\
& = & H(S(a)) & \text{Independence of sources}
\end{array}
$$

By symmetry, $H(vu) \geq H(S(b))$. Now consider the joint entropy $H(uv, vu)$. Using downstreamness relations, we can only prove a lower bound on the joint entropy of the random variables associated with a set of edges if at least one in-edge to a sink is downstream of the set of edges. However, edge $T(a)$ is only downstream of edge sets containing $S(b)$. Similarly, edge $T(b)$ is only downstream of edge sets containing $S(a)$. Therefore, using downstreamness the strongest lower bound we can derive is $H(uv, vu) \geq H(S(i))$ where $H(S(i))$ is the entropy of one of the two sources.

Intuitively, the pair of edges (u, v) and (v, u) need to send two sources worth of information between the nodes u and v. To prove this we need to define a stronger condition than downstreamness and consider the time-expanded graph.

6.2.2 Definition of Operational Downstreamness

We present a definition of a relationship called operational downstreamness. We then prove an entropy inequality that holds if a set of edges is operationally downstream of another set of edges. Although it is not clear from the definition, there is a combinatorial condition that can be used to test if the relationship holds. We delay a discussion of the combinatorial characterization of operational downstreamness until the next section.

Definition 22 (Operational Downstreamness) *Let G be a directed graph. For generalized edge sets $U, U' \subseteq \hat{E}$, we say U' is operationally downstream of U, written $U \mathrel{\circ\!\!\rightsquigarrow} U'$,*

if for all network coding solutions on G_t, there exists a function g_{U_t,U'_t} mapping the symbols transmitted on edges in U_t to the symbols transmitted on edges in U'_t.

Note that if $U \rightsquigarrow U'$ then $U \circ\!\!\rightsquigarrow U'$.

Lemma 43 *If $U' \subset \hat{E}$ is operationally downstream from $U \subset \hat{E}$, then*

$$H(U) \geq H(U')$$

Proof. Let $X = \{Y_e : e \in \mathcal{S}\}$. Let Y_U and $Y_{U'}$ be the random variables associated with the edge sets U and U'. By the definition of operational downstreamness and causality, $X \rightarrow Y_U \rightarrow Y_{U'}$ forms a Markov chain. By the data-processing inequality, $I(X, Y_U) \geq I(X, Y_{U'})$. Equivalently,

$$H(Y_U) - H(Y_U|X) \geq H(Y_{U'}) - H(Y_{U'}|X)$$

Since X is the random variable representing the messages transmitted on all out-edges from sources, $H(Y_{U'}|X) = 0$ and $H(Y_U|X) = 0$. Therefore, $H(Y_U) \geq H(Y_{U'})$. □

6.2.3 Applying Operational Downstreamness to the Example

To understand the difference between downstreamness and operational downstreamness, we reconsider the graph in Figure 6-5. We will prove that $\{(u,v),(v,u)\} \circ\!\!\rightsquigarrow \{S(a), S(b)\}$. Using Lemma 43 this proves that $H((u,v),(v,u)) \geq H(S(a)) + H(S(b))$. There is no corresponding downstreamness relationship that proves this inequality.

Let $\hat{G}$ be the augmented graph in Figure 6-5 and let G_t be the time-expanded graph. Suppose that the relationship

$$\{(u,v),(v,u)\} \circ\!\!\rightsquigarrow \{S(a), S(b)\}$$

does not hold. Let U_t be the $2t$ edges in $\hat{G}$ associated with (u,v) and (v,u). If the operational downstreamness relationship does not hold, then there exists a network coding solution on G_t such that there does not exist a function mapping the symbols transmitted on edges in U_t to the symbols transmitted on $\{S(a), S(b)\}$. The edges $S(a)$ and $S(b)$ transmit the two messages M_a and M_b. If there does not exist a function mapping the symbols transmitted on edges in U_t to M_a and M_b then there are two 2-tuples of messages that induce the same symbols on edges in U_t. Let (x,y) and (x',y') be two assignments to (M_a, M_b) such that $(x,y) \neq (x',y')$ but $f_{U_t}(x,y) = f_{U_t}(x',y')$. Assume without loss of generality that $x \neq x'$.

Lemma 44 *For each edge $e \in \hat{E}$ with tail u_s $(0 \leq s \leq t)$,*

$$f_e(x',y) = f_e(x',y') = f_e(x,y). \tag{6.9}$$

For each edge $e \in \hat{E}$ with tail v_s $(0 \leq s < t)$,

$$f_e(x', y) = f_e(x, y) = f_e(x', y'). \tag{6.10}$$

Proof. In each of (6.9) and (6.10), the second equality follows from our assumption that $f_{U_t}(x, y) = f_{U_t}(x', y')$, so we need only prove the first equality in each. The proof is by induction on s. The base case $s = 0$ holds because every edge with tail u_0 satisfies $\mathrm{In}(e) = \{S(a)\}$ and therefore the value of f_e is completely determined by the message sent by source $s(a)$; the corresponding claim for edges with tail v_0 is established similarly. For the induction step, assume $s > 0$ and let $A = \mathrm{In}(e)$. By the induction hypothesis, $f_A(x', y) = f_A(x, y) = f_A(x', y')$. Since the functions $\{f_e\}$ form a network coding solution on G_t, there exists a function g_e which maps the symbols transmitted on edges in A to the symbol transmitted on e. Now

$$f_e(x', y) = g_e(f_A(x', y)) = g_e(f_A(x, y)) = f_e(x, y).$$

□

Applying the lemma to the edge $T(a) = (v_t, \tau(a))$, we conclude that $f_{T(a)}(x', y) = f_{T(a)}(x, y)$. This violates the correctness of the network coding solution, which requires that $f_{T(a)}(x', y) = x'$ and $f_{T(a)}(x, y) = x$.

Lemma 45 *Let G be the graph consisting of two vertices u, v and two directed edges $e_1 = (u, v)$ and $e_2 = (v, u)$, with two commodities a, b satisfying $s(a) = u, t(a) = v$ and $s(b) = v, t(b) = u$, as in Figure 6-5. Then $\{e_1, e_2\} \nrightarrow \{S(a), S(b)\} = S$.*

This lemma confirms our intuition that the two edges (u, v) and (v, u) have at least enough capacity as the sum of the demands between u and v. Next we extend this argument to a general directed edge cut in an undirected graph.

6.3 Characterization of Operational Downstreamness

The definition of operational downstreamness is convenient for proving the entropy inequality in Lemma 43. However, it isn't clear from the definition that given a set of edges A it is easy to determine which edges are operationally downstream of A.

In this section we present a combinatorial characterization of operational downstreamness. Given a set of generalized edges A, we are interested in determining the set of edges whose edge functions are completely determined by the symbols transmitted on edges in A *for all network coding solutions*. In order to do this we break the graph into *basins of influence*. The initial step in this decomposition is to determine for each source not in A the *basin of reachability* of the source. Roughly, for a source edge not in A, the basin of reachability with respect to edge set A is the subgraph reachable from the source without

crossing an edge in A. Once each source's basin of reachability is determined, we group overlapping basins of reachability into a single basin of influence.

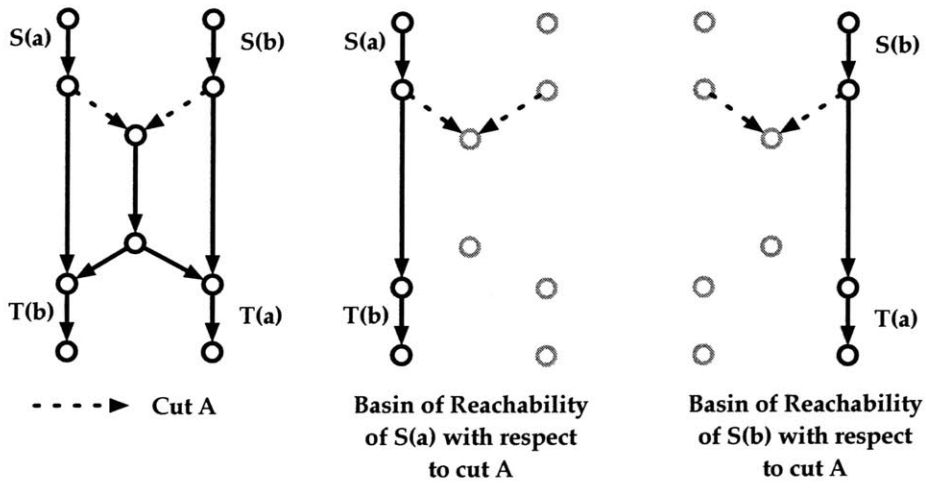
We prove that the source edge for a commodity whose source and sink are in different basins of influence with respect to an edge set A is operationally downstream of the edge set A. If for all commodities whose source edge is not in A the source and sink are in the same basin of influence, then the set of edges A does not operationally downstream any additional source edges.

For the rest of this section, we are given an instance of the k-pairs communication problem. Let $\hat{G} = = (\hat{V}, \hat{E})$ be the augmented graph for this instance. Let A be a set of generalized edges of $\hat{G}$.

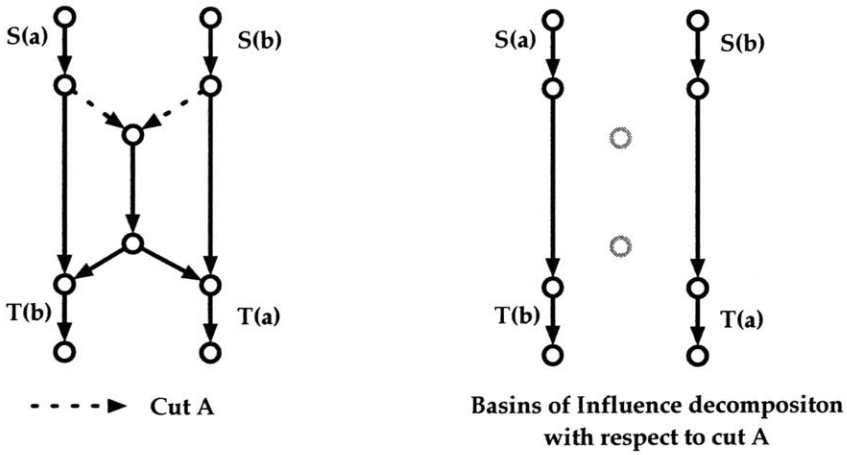
6.3.1 Basins of Influence Decomposition

We define the decomposition of the augmented graph $\hat{G}$ into basins of reachability and then basins of influence with respect to the set A of generalized edges. Note that operational downstreamness can be applied in directed as well as undirected graphs. The figures that illustrate the combinatorial decomposition use a directed graph as the example.

For a source edge $S(i)$, the nodes and edges of $\hat{G}$ reachable from $S(i)$ without crossing an edge in A is the *basin of reachability* of source $S(i)$ with respect to A. We denote this $\text{Basin}(S(i), A)$. The following figure shows a basins of reachability decomposition.



Let $\tilde{G}$ be the union of the basins of reachability. The weakly connected components of $\tilde{G}$ form the basins of influence. For the same graph and cut A, the following figure shows the basins of influence with respect to A.



Basins of Influence decompositon with respect to cut A

Note that edges e and e' are in the same basin of influence if there exists a sequence of source edges $S(i_1), S(i_2) \ldots S(i_\ell)$ such that

- $e \in \text{Basin}(S(i_1), A)$
- $e' \in \text{Basin}(S(i_\ell), A)$.
- for $1 \leq q < \ell$, the basins of reachability $\text{Basin}(S(i_q), A)$ and $\text{Basin}(S(i_{q+1}), A)$ share at least one node.

6.3.2 Basins of Influence and Operational Downstreamness: Part I

Understanding the combinatorial characterization of operational downstreamness is a two step process. We start by showing that the source edge for a commodity is operationally downstream of a set of generalized edges A if the source and sink for that commodity are in different basins of influence with respect to A.

Suppose source edge $S(i)$ and sink edge $T(i)$ are in different basins of influence with respect to edge set A. Our goal is to prove that $A \multimap S(i)$.

The argument is by contradiction. We assume that $S(i)$ is not operationally downstream of A. Let A_t be the set of edges in G_t associated with A. By definition, $A \multimap S(i)$ if for all network coding solutions, the symbols transmitted on edges in A_t determine the symbol transmitted on $S(i)$. Since we are assuming this relationship does not hold, there must exist a network coding solution such that

1. There exists two different k-tuples of messages.
2. The edge $S(i)$ transmits different symbols for the two different k-tuple of messages.

3. The edges in A_t transmit the same symbols under both k-tuples of messages.

Let z and z' be two such k-tuples of messages. Let f_{A_t} be the function for this network coding solution that maps k-tuples of messages to symbols transmitted on edges in A_t. Our assumption is that the k-tuples of messages z and z' assign different messages to commodity i and $f_{A_t}(z) = f_{A_t}(z')$.

Our argument begins by breaking the graph into two parts. Let B be the basin of influence containing $S(i)$ and $\overline{B}$ be $\hat{G} \setminus B$. Let B_t be the part of G_t associated with the basin B. We partition the commodities based on which part of the graph they are in. Let $\mathcal{S}(B)$ and $\mathcal{S}(\overline{B})$ be the source edges in B and $\overline{B}$ respectively. Now, from z and z' we create four k-tuples of messages.

- $x = \pi_z(\mathcal{S}(B))$
- $y = \pi_z(\mathcal{S}(\overline{B}))$
- $x' = \pi_{z'}(\mathcal{S}(B))$
- $y' = \pi_{z'}(\mathcal{S}(\overline{B}))$

The four k-tuples of messages which we use in our argument are $z = (x, y)$, $z' = (x', y')$, (x, y') and (x', y). By assumption $x \neq x'$.

We now outline the argument. We start by considering the time expanded graph. In this graph we prove a set of downstreamness relationships. The first set of downstreamness relationships show that every edge in B_t is downstream of the sources $\mathcal{S}(B)$ and the edges in A_t. Similarly, we show that every edge in $\overline{B}_t$ is downstream of the edges in A_t and the sources $\mathcal{S}(\overline{B})$. These allow us to show that the edges in A_t transmit the same symbols under all *four* k-tuples of messages. In particular, we prove that the edges in A_t transmit the same symbols for the message tuple (x, y) and the message tuple (x', y). Since the edge $T(i)$ is downstream from the edges in A_t and the sources in $\mathcal{S}(\overline{B})$, it must also transmit the same symbol under the two k-tuples of messages. This creates a contradiction since we assumed that x and x' assign different messages to commodity i.

For $0 \leq s \leq t$, let $A_s = \{e_i \in A : i \leq s\}$. Therefore A_0 is the empty set, A_1 is the set of edges in A_t between the first two layers of the time-expanded graph G_t and so on.

Lemma 46 *For an edge* (u_s, v_{s+1}) *with* $u_s \in B_t$

$$\{A_{s-1} \cup \mathcal{S}(B)\} \rightsquigarrow (u_s, v_{s+1})$$

For an edge (u_s, v_{s+1}) *with* $u_s \in \overline{B}_t$

$$\{A_{s-1} \cup \mathcal{S}(\overline{B})\} \rightsquigarrow (u_s, v_{s+1})$$

Proof. Suppose $u \in B$. By assumption, every path from a source edge $S(j) \in \overline{B}$ to u intersects A. Therefore, in G_t every path from $S(j)$ to u_s intersects A_{s-1}. Therefore (u_s, v_{s+1}) is downstream of $A_{s-1} \cup \mathcal{S}(B)$. A similar argument proves the claim if $u \in \overline{B}$. □

Lemma 47 *For an edge $e = (u_s, v_{s+1}) \in A_t$ with $u_s \in B_t$*

$$f_e(x, y') = f_e(x, y) = f_e(x', y') \tag{6.11}$$

For an edge $(u_s, v_{s+1}) \in A_t$ with $u_s \in \overline{B}_t$

$$f_e(x, y') = f_e(x', y') = f_e(x, y) \tag{6.12}$$

Proof. In each of (6.11) and (6.12), the second equality follows from our assumption that $f_{A_t}(x, y) = f_{A_t}(x', y')$. We prove this claim by induction on s. Let $e_s = (u_s, v_{s+1})$. For the base case, $s = 0$. Suppose $u \in B$, then $\text{In}(e_0) \in \mathcal{S}(B)$. Since (x, y') and (x, y) send the same message on edges in $\mathcal{S}(B)$, $f_{e_0}(x, y') = f_{e_0}(x, y)$. Similarly for $u \in \overline{B}$.

Now assume the claim is true for all $i < s$. Let $u \in B$ and let $e = (u_s, v_{s+1})$. The message tuples (x, y) and (x, y') transmit the same messages on edges in $\mathcal{S}(B)$. By the inductive hypothesis they also induce the same symbols on edges in A_{s-1}. Let $C = \{A_{s-1} \cup \mathcal{S}(B)\}$.

$$f_C(x, y) = f_C(x, y')$$

By Lemma 46, $C \rightsquigarrow e$. Therefore, there exists a function h_{Ce} which maps the symbols on edges in C to the symbol on edge e.

$$f_e(x, y) = h_{Ce}(f_C(x, y)) = h_{Ce}(f_C(x, y')) = f_e(x, y')$$

A similar argument proves the claim if $u \in \overline{B}$. □

Applying the above Lemma to edge $T(i)$ yields

$$f_{T(i)}(x', y) = f_{T(i)}(x, y)$$

However, we assumed that x and x' transmit different functions on edge $S(i)$. Therefore, this is not a network coding solution.

Lemma 48 *For a k-communication problem, let $G = (V, E)$ be the underlying graph If $S(i)$ and $T(i)$ are in different basins of influence with respect to a set A of generalized edges, then $A \circ\!\!\rightsquigarrow S(i)$.*

6.3.3 Basins of Influence and Operational Downstreamness: Part II

We now show that the basins of influence condition in the previous Lemma is also a necessary condition if the underlying graph is undirected. Specifically, given a set A of generalized edges, if for all commodities i such that $S(i) \notin A$, the edges $S(i)$ and $T(i)$ are in the same basin of influence, then the set A does not operationally downstream any additional sources.

Lemma 49 *For a k-communication problem, let $G = (V, E)$ be the underlying undirected graph and $\hat{G}$ be the augmented graph. Given $A \subset \hat{E}$ if for all $S(i) \notin A$, $S(i)$ and $T(i)$ are in the same basin of influence, then A does not operationally downstream any edge $T(i)$ with $S(i) \notin A$.*

Proof. Without loss of generality, if A contains an edge $T(j)$ then we replace it with edge $S(j)$. We construct a network coding solution and two $\mathcal{I}$-tuples of messages x and x' such that

- the edges in A transmit the same symbols on inputs x and x',
- and for all i such that $S(i) \notin A$, $x_i \neq x'_i$.

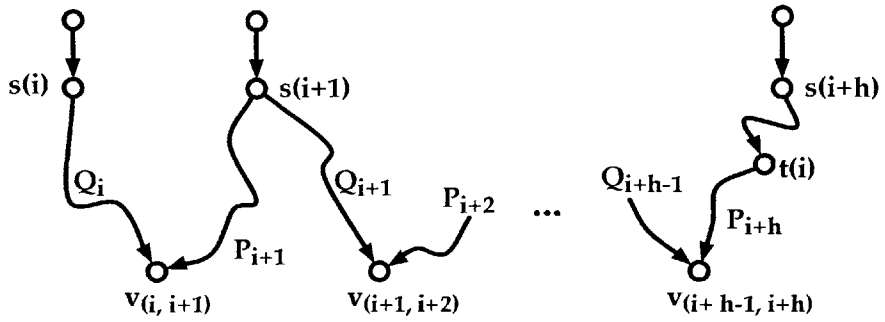
We construct a network coding solution in the straight-line program formulation. For all commodities j with $S(j) \in A$, we fix $\Sigma(S(j)) = 1$. For commodities j with $S(j) \in A$, let $\Sigma(S(j)) = \{0, 1\}$ and $x_j = 0$ and $x'_j = 1$.

In defining the straight-line program, let M be a variable which represents the $|\mathcal{I}|$-tuple of messages (i.e. if the $|\mathcal{I}|$-tuple of messages is x, then $M_i = x_i$ and if the $|\mathcal{I}|$-tuple of messages is x', then $M_i = x'_i$). The first k rules of the straight-line program specify that $S(i) \in \mathcal{S}$ transmits the message M_i.

We define the remaining straight-line program in blocks. For edge $S(i) \notin A$, the ith block specifies a code which allows $T(i)$ to receive M_i. We describe this block for commodity i. By the conditions of the lemma, $S(i)$ and $T(i)$ are in the same basin of influence. Therefore, there exists an alternating path P of edges connecting $S(i)$ to $T(i)$. (What is meant by alternating path is that the path uses some edges in the reverse direction but we do not require that the direction of the edges strictly alternate.) Without loss of generality, we assume the first edge in P is an out-edge from node $s(i)$. In addition, we assume the path P can be decomposed as:

$$P = Q_i, P_{i+1}, Q_{i+1}, P_{i+2}, Q_{i+2} \ldots P_{i+h-1}, Q_{i+h-1}, P_{i+h}$$

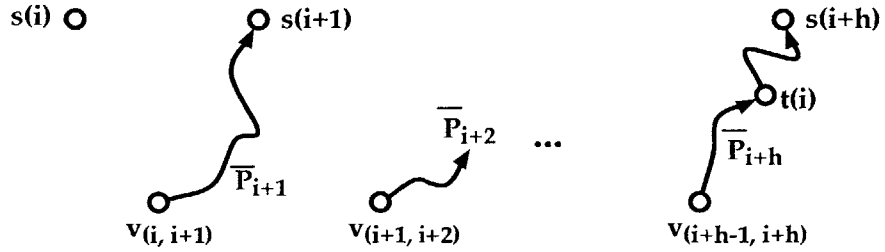
where the pair of paths P_j and Q_j are both directed paths that start at a node $s(j)$ where $S(j) \notin A$. The following figure shows what the path P may look like.



Note that the node $t(i)$ which is the head of $T(i)$ is contained in the path P_{i+h} but may not be the first node on that path. To simplify notation, reorder the commodities according to the appearance of $s(j)$ in path P.

For a pair of paths Q_j and P_{j+1}, let $v_{(j,j+1)}$ be the first node in common to the paths. For each j such that there is a sub-path P_j in P, we create a set of rules which sends the message M_j from the source $s(j)$ along P_j to $v_{(j-1,j)}$.

Consider the node $v_{(i,i+1)}$. There are rules in the network coding solution which have sent M_i to $v_{(i,i+1)}$ along Q_i. There are also rules which have sent M_{i+1} along P_{i+1} to $v_{(i,i+1)}$. Now we use a path $\tilde{P}_{i+1}$ to send $M_i \oplus M_{i+1}$ from node $v_{(i,i+1)}$ to node $s(i+1)$. Since the underlying graph is undirected and there is a path P_{i+1} from node $s(i+1)$ to node $v_{(i,i+1)}$, there must exist a path $\tilde{P}_{i+1}$ from $v_{(i,i+1)}$ to $s(i+1)$. The next figure depicts these paths.



After this set of rules has been specified, there are rules which have transmitted $M_i \oplus M_{i+1}$ to node $s(i+1)$. The source edge $S(i+1)$ is an in-edge to $s(i+1)$. Therefore, there are already rules that have sent message M_{i+1} to node $s(i+1)$. Using the information in previously specified rules, the node is able to compute the value of the message M_i. We then create a set of rules which transmit M_i along path Q_{i+1} to node $v_{i+1,i+2}$.

The remaining rules in this block are specified as follows.

For $j = i+2$ to $j = i+h-1$

- Create a set of rules to transmit $M_i \oplus M_j$ from $v_{j-1,j}$ to node $s(j)$ along path $\tilde{P}_j$.

- Create a set of rules to transmit M_i from $s(j)$ to $v_{j,j+1}$ along path Q_j.

Lastly, a set of rules sends $M_i \oplus M_{i+h}$ from $v_{(i+h-1,i+h)}$ to $t(i)$ along part of path $\tilde{P}_{i+h}$. The last rule the block for commodity i sends M_i from $t(i)$ to $\tau(i)$ on edge $T(i)$.

For each commodity i such that $S(i) \notin A$ we create a block of rules as described above which transmit M_i to $T(i)$. It is not difficult to check that every rule can be computed from previously specified rules. Since edges in A always send $0 \oplus 0$ or $1 \oplus 1$, they transmit the same symbol under x and x'. Therefore, edge $S(i)$ is not operationally downstream of A if $T(i)$ is in the same basin of influence with respect to A. □

If G is an undirected graph and $A \subset E$ is a set of *undirected* edges, then the set of commodities which are operationally downstream from A is exactly the set of commodities whose source and sink are in different connected components of $G' = (V, E \setminus A)$. Therefore, as a corollary of Lemmas 48 and 49, we arrive at an alternate proof that the sparsity of G is an upper bound on the maximum rate achievable with network coding.

Corollary 50 *Given an instance of the k-pairs communication problem on an undirected graph G, the maximum achievable rate is at most the sparsity of G.*

Similarly, for an undirected cut A in G, operational downstreamness says that $H(A) \geq H(\mathcal{I}(A))$ where $\mathcal{I}(A)$ is the set of commodities whose source and sink are in different connected components of $G' = (V, E \setminus A)$. Recall the linear program at the end of Chapter 5. Suppose instead of considering directed cuts in G, we only consider undirected cuts and use operational downstreamness. If the optimal value for this LP is equal to the maximum multicommodity flow, then combining sparse cuts using the submodularity of the entropy function is sufficient to determine the maximum multicommodity flow. Such a relationship would shed new light on the gaps that can exist between the sparsity of G and the maximum multicommodity flow.

6.4 Okamura-Seymour with Five Commodities

Recall that the 5-commodity Okamura-Seymour example is the 5-node, 5-commodity instance depicted in Figure 6-6. We will name the edges of this graph by simply listing their tail and head, in that order. For instance, ad refers to the edge from node a to node d. Now consider the first half of Figure 6-7, which illustrates the downstreamness relation

$$\{da, ea, S(a)\} \rightsquigarrow \{da, ea, S(a), T(c)\}.$$

This implies the entropy inequality

$$H(da, ea, S(a)) \geq H(da, ea, S(a), T(c)) = H(da, ea, S(a), S(c)). \tag{6.13}$$

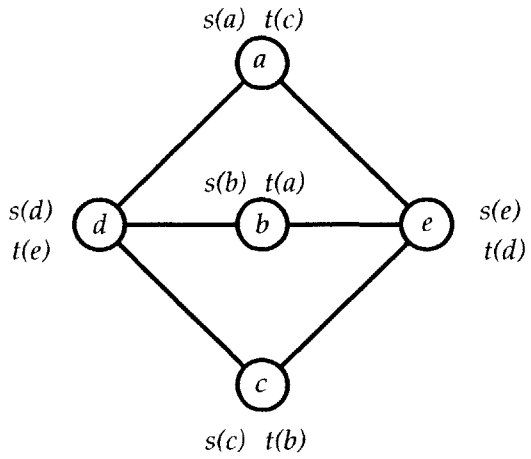


Figure 6-6: The 5-commodity Okamura-Seymour Example.

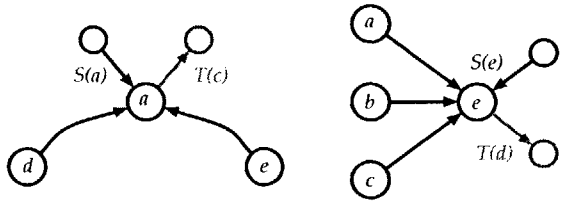


Figure 6-7: Two downstreamness relations.

Similar cuts using the incoming edges of nodes b and c lead to the corresponding inequalities:

$$H(db, eb, S(b)) \geq H(db, eb, S(b), S(a)) \tag{6.14}$$
$$H(dc, ec, S(c)) \geq H(dc, ec, S(c), S(b)). \tag{6.15}$$

Summing (6.13) and (6.14), and using submodularity, we obtain

$$H(da, ea, S(a)) + H(db, eb, S(b)) \geq H(da, ea, db, eb, S(a), S(b), S(c)) + H(S(a)). \tag{6.16}$$

Summing (6.15) and (6.16), and using submodularity and independence of sources, we

obtain

$$H(da, ea, S(a)) + H(db, eb, S(b)) + H(dc, ec, S(c)) \\ \geq \; H(da, ea, db, eb, dc, ec, S(a), S(b), S(c)) + H(S(b), S(C)) + H(S(a)). \tag{6.17}$$

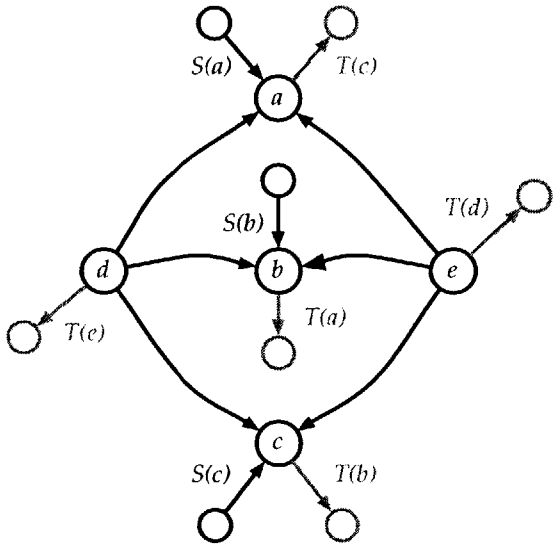


Figure 6-8: An operational downstreamness relation.

We now use operational downstreamness. Figure 6-8 illustrates the operational downstreamness relation

$$\{da, ea, db, eb, dc, ec, S(a), S(b), S(c)\} \multimap \{S(a), S(b), S(c), S(d), S(e)\}$$

because the basin of influence of each source is a singleton set. This implies the entropy inequality

$$H(da, ea, db, eb, dc, ec, S(a), S(b), S(c)) \geq H(S(a), S(b), S(c), S(d), S(e)).$$

Combining this with (6.17) and using independence of sources, we obtain

$$H(da, ea, S(a)) + H(db, eb, S(b)) + H(dc, ec, S(c)) \\ \geq \; 2H(S(a)) + 2H(S(b)) + 2H(S(c)) + H(S(d)) + H(S(e)). \tag{6.18}$$

Submodularity of entropy implies that

$$H(da) + H(ea) + H(S(a)) \geq H(da, ea, S(a))$$

and similarly for the other two terms on the left side of (6.18). Expanding out the left side using these inequalities, and canceling the term $H(S(a)) + H(S(b)) + H(S(c))$ on both sides, we arrive at:

$$\begin{aligned} & H(da) + H(ea) + H(db) + H(eb) + H(dc) + H(ec) \\ & \geq \quad H(S(a)) + H(S(b)) + H(S(c)) + H(S(d)) + H(S(e)). \end{aligned} \tag{6.19}$$

The second half of Figure 6-7 illustrates the downstreamness relation:

$$\{ae, be, ce, S(e)\} \rightsquigarrow \{ae, be, ce, S(e), T(d)\}.$$

Similarly,

$$\{ad, bd, cd, S(d)\} \rightsquigarrow \{ad, bd, cd, S(d), T(e)\}.$$

These lead to the entropy inequalities

$$H(ae, be, ce, S(e)) \geq H(ae, be, ce, S(e), T(d)) = H(ae, be, ce, S(e), S(d))$$

$$H(ad, bd, cd, S(d)) \geq H(ad, bd, cd, S(d), T(e)) = H(ad, bd, cd, S(d), S(e)).$$

Adding and using submodularity, we derive:

$$\begin{aligned} & H(ae, be, ce, S(e)) + H(ad, bd, cd, S(d)) \\ & \geq \quad H(ae, be, ce, ad, bd, cd, S(d), S(e)) + H(S(d), S(e)). \end{aligned} \tag{6.20}$$

Figure 6-9 illustrates the operational downstreamness relation

$$\{ae, be, ce, ad, bd, cd, S(d), S(e)\} \mathrel{\circ\!\!\rightsquigarrow} \{S(a), S(b), S(c), S(d), S(e)\}$$

because the basin of influence of each source is a singleton set. This implies the entropy inequality

$$H(ae, be, ce, ad, bd, cd, S(d), S(e)) \geq H(S(a), S(b), S(c), S(d), S(e)).$$

Combining this with (6.20) we obtain

$$\begin{aligned} & H(ae, be, ce, S(e)) + H(ad, bd, cd, S(d)) \\ & \geq \quad H(S(a), S(b), S(c), S(d), S(e)) + H(S(d), S(e)). \end{aligned}$$

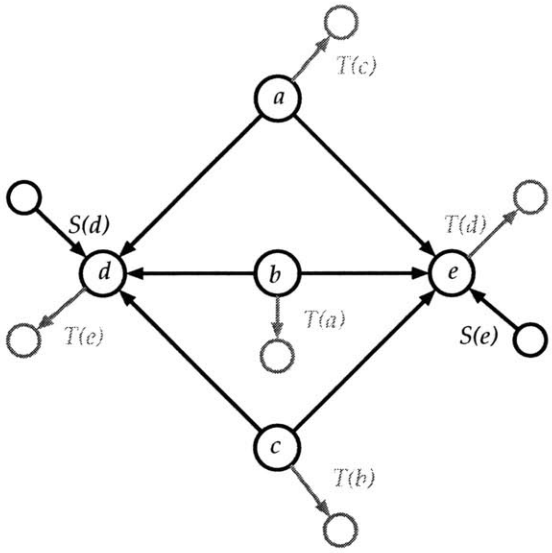


Figure 6-9: An operational downstreamness relation.

If we expand both terms on the left side using submodularity as we did with equation (6.18), expand the right side using independence of sources, and cancel the terms $H(S(d)) + H(S(e))$ which appear on both sides, we obtain:

$$\begin{aligned} & H(ae) + H(be) + H(ce) + H(ad) + H(bd) + H(cd) \\ & \geq \; H(S(a)) + H(S(b)) + H(S(c)) + H(S(d)) + H(S(e)). \end{aligned} \tag{6.21}$$

Summing (6.19) and (6.21), we find that the sum of the entropies of all 12 directed edges (i.e. the sum of the entropies of all 6 undirected edges) is at least twice the sum of the entropies of all 5 sources. Using Lemma 36, this implies that

$$\sum_{e \in E(G)} c(e) \geq 2 \sum_{i \in \mathcal{I}} r d_i. \tag{6.22}$$

In the 5-commodity Okamura-Seymour example, all undirected edges have capacity 1, and all commodities have demand 1, so (6.22) reduces to $6 \geq 10r$, i.e. $r \leq 3/5$, as desired. In the 4-commodity Okamura-Seymour example, all undirected edges have capacity 1, and $d_a = d_b = d_c = d_d = 1$ while $d_e = 0$. Thus (6.22) reduces to $6 \geq 8r$, i.e. $r \leq 3/4$, as desired.

6.5 Special Bipartite Graphs

The analysis of the 5-commodity Okamura-Seymour example extends to an infinite family of graphs. Specifically, suppose G is a bipartite graph whose vertex set is partitioned into two independent sets V, W. Consider an instance of the k-pairs communication problem in G with the property that for every commodity, the source and sink both belong to V or they both belong to W. Let $\mathcal{S}(V)$ denote the set of sources in V, and $\mathcal{S}(W)$ the set of sources in W. For a vertex v, let $\mathrm{In}(v)$ denote the set of incoming edges of v in $\hat{G}$, and let $\mathcal{T}(v)$ denote the set of edges in $\mathcal{T}$ whose tail is v. We have a downstreamness relation

$$\mathrm{In}(v) \rightsquigarrow \mathrm{In}(v) \cup \mathcal{T}(v).$$

Summing over all elements of V, and repeatedly applying the submodularity axiom, we derive

$$\sum_{v \in V} H(\mathrm{In}(v)) \geq H(\mathrm{In}(V)) + H(\mathcal{S}(V)).$$

Using operational downstreamness, we derive

$$H(\mathrm{In}(V)) \geq H(\mathcal{S}).$$

Combining these two inequalities and rearranging some terms using submodularity and independence of sources,

$$\sum_{e \in E(W,V)} H(e) \geq H(\mathcal{S}).$$

Here $E(W, V)$ denotes the set of directed edges from W to V in $\hat{G}$. Similarly, we may derive

$$\sum_{e \in E(V,W)} H(e) \geq H(\mathcal{S}).$$

Summing these two inequalities, we obtain an entropy inequality which implies

$$\sum_{e \in E} c(e) \geq 2 \sum_{i \in \mathcal{I}} r d_i,$$

i.e.

$$r \leq \frac{\sum_e c(e)}{2 \sum_i d_i}.$$

This inequality is tight in instances where each source-sink pair is joined by a 2-hop path, and a dual-optimal length function assigns length 1 to every edge of G, as was the case with the 4-commodity and 5-commodity Okamura-Seymour examples. It is not hard to come up with infinitely many other bipartite graphs satisfying this property.

Theorem 51 *Multicommodity flow achieves the same maximum rate as network coding in any instance of the k-pairs communication problem defined on an underlying undirected bipartite graph G where a dual-optimal length function assigns 1 to every edge and every source-sink pair is distance 2 apart.*

6.6 Open Problems

There are three fundamental open questions related to the work in this chapter.

- **What is the maximum rate achievable for an instance of the network coding problem defined on an undirected graph?** Our proofs in this chapter made use of the properties of the entropy function and the concepts of downstreamness and operational downstreamness. Are these the essential ingredients in determining the maximum achievable rate? These conditions can be specified as a linear program, which is extremely large even for small instances. Is the optimal value of this linear program equal to the maximum achievable network coding rate?

- **For an instance of the k-pairs communication problem on an undirected graph, do multicommodity flow techniques always achieve the maximum rate?** The answer is yes for all instances in which the maximum multicommodity flow value is equal to the value of the sparsest cut. In addition, we've found an infinite class of instances for which the answer is yes even though there is a gap between the maximum multicommodity flow value and the value of the sparsest cut. Does this result extend to all instances of the k-pairs communication problem on undirected graphs? A related question is to understand the relationship between the polytope of feasible multicommodity flows and the polytope characterizing the entropies associated with network coding solutions for a given instance of the k-pairs communication problem.

- **Do sparse cuts and submodularity characterize the maximum multicommodity flow value?** It is well known that there are instances of the multicommodity flow problem for which there is a large gap between the value of the sparsest cut and the maximum flow value. Suppose that the maximum rate achievable with network coding is also achievable using flow techniques. Then techniques which characterize the maximum network coding rate also characterize the maximum multicommodity flow.

  Theorem 50 provides an information theoretic characterization of the sparsity of an undirected cut. Is there a way to combine the resulting inequalities together and use the submodularity of the entropy function to prove a tight upper bound on the maximum multicommodity flow rate?

6.7 References

This chapter explores the gap between the maximum rate achievable with and without coding. In directed acyclic graphs, network coding can achieve a rate $\Omega(n)$ times larger, where n is the number of nodes in the graph, than multicommodity flow techniques. Li et al. [27] showed that for the multicast problem in *undirected* graphs, an optimal tree packing exists which achieves at least half the maximum rate achievable with network coding. This result showed that communication problems in directed and undirected graphs have different properties.

Li and Li [26] and Harvey et al. [11] conjectured that for undirected graphs, the maximum rate for an instance of the k-pairs communication problem is achievable with multicommodity flow techniques.

The results in this chapter are joint work with Nicholas Harvey and Robert Kleinberg. Very similar results have been independently discovered [17].

Chapter 7

Future Work

We now consider four broad directions for future work. More specific questions were listed at the end of each chapter.

Where can network coding be used? Optimal solutions for multicast network coding problems can be found efficiently. For general communication problems beyond multicast, network coding can substantially increase the amount of information that can be transmitted through a network. An algorithm for finding high rate network codes for the general network coding problem would be very useful but none is currently known. Even approximation algorithms could result in more efficient use of network bandwidth.

What is the complexity of the network coding problem? The multicast problem is known to be in efficiently solvable. However, determining if there exists a solution of a given rate for the general network coding problem could be undecidable. One issue is that there is no known upper bound on the sizes of the alphabets that need to be considered. Suppose we knew a function $p(n)$ such that for every solvable instance on a network with n nodes, there exists a solution in which every edge uses an alphabet of size at most $p(n)$. Then we could determine if an instance is solvable, at any rate, by checking all network coding solutions with alphabets of size at most $p(n)$ for each edge. Another possibility for showing decidability is to prove that the optimal values of the linear programs presented in Chapters 4 and 5 define the maximum achievable rate.

Does there exist a subset of all functions that are sufficient for all network coding problems? All the known algorithms for the multicast problem make use of the fact that any solvable instance admits a linear solution over some finite field $\mathbb{F}$. Can we find a restricted class of functions such that any solvable instance of the general network coding problem admits a solution using only functions from that class? If

this class of functions had a nice enough structure, then efficient algorithms might follow from its characterization.

What is the capacity of an information network? This is the $64,000 dollar question. Prior to the paper by Ahlswede et al. [1], the capacity of information networks was not well understood. Their characterization of the feasible rate region for the multicast problem was the first significant progress on this important open question. In Chapters 4 and 5, we presented some upper bounds on the maximum achievable rate. Are these upper bounds tight? Do we need additional techniques to understand the capacity of information networks? In the k-pairs communication problem on undirected graphs, how does the maximum achievable rate compare to the maximum multicommodity flow?

Appendix A

Entropy

We review the definition of Shannon entropy and some of its properties.

Definition 23 Entropy: *Let X_i be a random variable taking on values in Σ_i. Let $\Pr(\alpha_i)$ be the probability that $X_i = \alpha_i$. The Shannon* entropy *of X_i is given by*

$$\begin{aligned} H(X_i) &= -\sum_{\alpha_i \in \Sigma_i} \Pr(\alpha_i) \log_2 \Pr(\alpha_i) \\ &= -Ex(\log_2 \Pr(\alpha_i)) \end{aligned}$$

Let $X = \{X_1, X_2 \ldots X_n\}$ be a set of random variables taking on values in $\Sigma = \Sigma_1 \times \Sigma_2 \ldots \Sigma_n$. Let $\Pr(\alpha) = \Pr(\alpha_1, \alpha_2 \ldots \alpha_n)$ be the joint probability that $X_i = \alpha_i$ for all i. The joint entropy *$H(X) = H(X_1, X_2 \ldots X_n)$ is defined as follows.*

$$\begin{aligned} H(X_1, X_2 \ldots X_n) &= -\sum_{(\alpha_1, \alpha_2 \ldots \alpha_n) \in \Sigma} \Pr(\alpha_1, \alpha_2 \ldots \alpha_n) \log \Pr((\alpha_1, \alpha_2 \ldots \alpha_n)) \\ &= -Ex(\log \Pr(\alpha_1, \alpha_2 \ldots \alpha_n)) \end{aligned}$$

For a pair of random varaibles X and Y, we can also define the conditional entropy $H(X|Y) = -\mathrm{Ex}(\log \Pr(x|y))$. The joint entropy of a pair of random variables can be written in terms of conditional entropies: $H(X,Y) = H(X) + H(Y|X)$.

The mutual information between two random variables, written $I(X,Y)$, can be expressed as

$$\begin{aligned} I(X,Y) &= H(X) + H(Y) - H(X,Y) \\ &= H(X) - H(X|Y) \\ &= H(Y) - H(Y|X) \end{aligned}$$

Random varaibles X, Y, and Z are said to form a Markov chain, $X \rightarrow Y \rightarrow Z$ if X is conditionally independant of $Z|Y$. In the special case where $Z = g(Y)$ for any function g, $X \rightarrow Y \rightarrow Z$ always holds.

Theorem 52 Data Processing Inequality*[5] If $X \rightarrow Y \rightarrow Z$, then $I(X;Y) \geq I(X;Z)$.*

A few other charateristics of the entropy function are also useful. (See page 297 in [39].)

Nonegativity: For any random varaible X,

$$H(X) \geq 0$$

Nondecreasing: For a set of random variables $X = \{X_1, X_2 \ldots X_n\}$ and any random variable X_{n+1},

$$H(X \cup \{X_{n+1}\}) \geq H(X)$$

Submodularity: For any two sets of random variables $X = \{X_1, X_2 \ldots X_n\}$ and $Y = \{Y_1, Y_2, \ldots Y_m\}$,

$$H(X) + H(Y) \geq H(X \cup Y) + H(X \cap Y)$$

Bibliography

[1] Rudolf Ahlswede, Ning Cai, Shuo-Yen Robert Li, and Raymond W. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, July 2000.

[2] M. Bern and P. Plassman. The steiner problem with edge lengths 1 and 2. *Information Processing Letters*, 32:171–176, 1989.

[3] Yu. D. Burago and V. A. Zalgaller. Geometric inequalities. In *Grundlehren der mathematischen Wiseenchaften*. Spinger-Verlag, 1988. Number 285.

[4] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Cliff Stein. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 2001.

[5] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, 1991.

[6] R. Dougherty, C. Freiling, and K. Zeger. Linearity and solvability in multicast networks, December 2003.

[7] R. Dougherty, C. Freiling, and K. Zeger. Insufficiency of linear coding in network information flow, February 2004.

[8] Christina Fragouli and Emina Soljanin. Information flow decomposition for network coding, 2004.

[9] G.H. Hardy and E.M. Wright. *An Introduction to the Theory of Numbers, 5th Edition*. Clarendon Press, Oxford, England, 1979.

[10] Nicholas J. A. Harvey, David R. Karger, and Kazuo Murota. Deterministic network coding by matrix completion. In *Proceedings of the Sixteenth Annual Symposium on Discrete Algorithms (SODA 05)*, page To Appear, 2005.

[11] Nicholas J. A. Harvey, Robert D. Kleinberg, and April Rasala Lehman. Comparing network coding with multicommodity flow for the k-pairs communication problem. Technical Report 964, M.I.T. LCS, September 2004.

[12] Pavol Hell and Jaroslav Nešetřil. On the complexity of h-coloring. *Journal of Combinatorial Theory, Series B*, 48:92–110, 1990.

[13] Tracey Ho. *Networking from a network coding perspective*. PhD thesis, MIT, 2004.

[14] Tracey Ho, Ralf Koetter, Muriel Médard, David R. Karger, and Michelle Effros. The benefits of coding over routing in a randomized setting. In *Proceedings of the IEEE International Symposium on Information Theory*, 2003.

[15] Sidharth Jaggi, Peter Sanders, Philip A. Chou, Michelle Effros, Sebastian Egner, Kamal Jain, and Ludo Tolhuizen. Polynomial time algorithms for multicast network code construction. *IEEE Transactions on Information Theory*. Submitted July 2003.

[16] K. Jain, M. Mahdian, and M.R. Salavatipour. Steiner trees. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms (SODA)*, pages 266–274, 2003.

[17] Kamal Jain. Private communication, January 2005.

[18] Ralf Koetter. Network coding home page. `http://tesla.csl.uiuc.edu/~koetter/NWC/`.

[19] Ralf Koetter and Muriel Médard. An algebraic approach to network coding. *IEEE/ACM Transactions on Networking*, 11:782–795, October 2003.

[20] Lap Chi Lau. An approximate max-steiner-tree-packing min-steiner-cut theorem. In *Proceedings of the 45th annual IEEE Symposium on Foundations of Computer Science (FOCS 2004*, pages 61–70, 2004.

[21] April Rasala Lehman and Eric Lehman. Complexity classification of network information flow problems. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, January 2004.

[22] April Rasala Lehman and Eric Lehman. Network coding: Does the model need tuning? In *Proceedings of the Sixteenth Annual Symposium on Discrete Algorithms (SODA 05)*, page To Appear, 2005.

[23] Tom Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM*, 46(6):787–832, November 1999.

[24] S.-Y. R. Li, Raymond W. Yeung, and Ning Cai. Linear network coding. *IEEE Transactions on Information Theory*, 49(2):371–381, 2003.

[25] Zongpeng Li and Baochum Li. Network coding: the case of multiple unicast sessions. In *Proceedings of the 42nd Allerton Annual Conference on Communication, Control, and Computing*, 2004.

[26] Zongpeng Li and Baochun Li. Network coding in undirected networks. In *Proceedings of the 38th Annual Conference on Information Sciences and Systems (CISS 2004),*, 2004.

[27] Zongpeng Li, Baochun Li, Dan Jiang, and Lap Chi Lau. On achieving throughput with network coding. In *Proceedings of INFOCOM 2005*, March 2005.

[28] L. H. Loomis and H. Whitney. An inequality related to the isoperimetric inequality. *Bulletin of the American Mathematical Society*, 1949.

[29] D. Ron M. Feder and A. Tavory. Bounds on linear codes for network multicast. *Electronic Colloquium on Computational Complexity (ECCC)*, 10(033), 2003.

[30] K. Makarychev, Y. Makarychev, A. Romashchenko, and N. Vereshchagin. A new class of non shannon type inequalities for entropies. *Communications in Information and Systems*, 2(2):147–166, December 2002.

[31] Muriel Médard, Michelle Effros, Tracey Ho, and David Karger. On coding for non-multicast networks. In *41st Annual Allerton Conference on Communication Control and Computing*, 2003.

[32] H. Okamura and P.D. Seymour. Multicommodity flows in planar graphs. *Journal of Combinatorial Theory, Series B*, 31:75–81, 1981.

[33] Soren Riis. Linear versus non-linear boolean functions in network flow. In *Conference on Information Sciences and Systems (CISS)*, March 2003.

[34] Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer-Verlag, 2003.

[35] D. B. Shmoys. Approximation algorithms for cut problems and their applications to divide-and-conquer. In D.S. Hochbaum, editor, *Approximation Algorithms for NP-hard problems*, pages 192–235. Course Technology, 1997.

[36] Lihua Song, Raymond Yeung, and Ning Cai. Zero-error network coding for acyclic networks. *IEEE Transactions on Information Theory*, 49:3129–3139, December 2003.

[37] Alexandre Tiskin. A generalization of the Cauchy and Loomis-Whitney inequalities. `http://www.dcs.warwick.ac.uk/~tiskin/pub/`.

[38] Vorlesungen über Inhalt. Oberfläche and isoperimetrie. In *Grundlehren der mathematischen Wissenchaften*. Springer-Verlag, 1957. Number 93.

[39] Raymond W. Yeung. *A First Course in Information Theory*. Kluwer, 2002.