Guessing Numbers Based on Network Routing Capacity

Christopher Buckley B.Sc. (Hons.) Dunelm

M.Sc. Thesis

Department of Computer Science

Durham University

30 May 2009

Abstract

In this thesis, a new method of studying problems in network information flow is introduced. This is established by developing on the relatively new concept of guessing numbers, which is related to network coding, and combining it with another technique known as fractional routing. The resultant guessing game, called a *fractional guessing game*, is a modification of the existing guessing game that incorporates the notion of fractional routing for multiple-unicast networks. It is essentially an intersection of two major areas of mathematics which allows further study into network flow problems, and the relationship between guessing numbers and the solvability of network flow problems.

To illustrate this relationship, we construct a graph (the pentagon or 5-cycle) that has a non-integer guessing number, by using a guessing strategy which ensures a guessing number in the half-open interval $(2, 2.5]$. This is an application of the link between guessing numbers on graphs and the solvability of their related networks. It is shown that the fractional guessing number of an example graph is higher than the guessing number computed using the existing linear approach. This guessing number is also dependent on the size of the alphabet from which the messages are constructed, answering an open question from the literature relating to the possibility of such a dependency. Finally, the extension of the method to include techniques of both routing capacity and network coding is discussed.

Acknowledgments

First and foremost, I would like to thank Stefan Dantchev, who has been my supervisor for both my third-year dissertation and my thesis. His intuition and knowledge in the field have been paramount in the writing of this thesis, and without which I would have had no idea where to even begin.

Also thanks go to Søren Riis, Stefan's Ph.D. supervisor, who has pioneered the concept of guessing numbers, and in personal communication has solved some of the most difficult questions I have had with this work, seemingly with ease.

Last, but by no means least, many thanks to my family and friends for supporting and putting up with me throughout my undergraduate degree and whilst I have been writing my thesis. I couldn't have finished it without you!

Statement of Copyright

Table of Contents

Page

List of Figures

Chapter 1

Introduction

This chapter provides an overall description of the field of work covered herein, as well as some outline for the remainder of the thesis.

The objective of this thesis is to introduce a new method — or at least improve on existing methods — for studying problems in network information flow. Linear coding methods have shown to be insufficient for certain solvable networks [12], and many open questions still remain regarding linear coding as an approach [24, 9]. Open questions are indeed a feature of a number of recent publications in this relatively new field; however, each paper does go some way to reducing the overall number of these questions. This thesis aims to do the same.

The approach for this thesis will be to study the theories and technologies currently in use, and to investigate where and why they prove to be inadequate. By highlighting these points of failure it should be possible to work toward an optimal solution to overcome these problems. Before investigating the modern technologies in use, however, it is necessary to provide some introduction of the history of networking.

1.1 Graph Theory

Graph theory is an area of mathematics, often regarded as being popularised when Leonhard Euler solved the problem of the Seven Bridges of Königsberg [5]. Euler's work, later generalised by other mathematicians, is said to be at the heart of topology, a branch of mathematics not introduced until over a century later. Graph theory contains a number of accessible problems, one of the most famous being the four colour theorem. This theorem is particularly famous due to the use of a computer to assist in the proof; the computer was used to show that only four colours were required for each of about 2000 maps, to one of which any map could be reduced [2]. In fact, the four colour theorem (or its earlier forms and attempts to solve it) has been suggested as the origin of the entire field of graph theory [34].

The use of a computer in proving mathematical theorems is a controversial issue, not least because of the lack of elegance in the proof. As with the four colour theorem, a computer is usually used to exhaustively prove a great number of cases that could not feasibly be produced by hand without years of computation. The objection to using computers in this manner is also that the proofs may not be logically verifiable, simply because of the number of steps. The algorithm and programs used must be fully trusted to accept the proof.

In addition to a number of colouring problems, there are various routing problems, some of which can be expressed as problems in computational complexity theory. Computational complexity theory is concerned with finding the most efficient algorithm to solve a given problem. An example is the travelling salesman problem, which seeks to find the Hamiltonian cycle with the least weight for a given complete weighted graph. In terms of computational complexity, finding such a cycle is **NP**-hard. A brute-force method for solving such a problem quickly becomes impractical, as the time required to find the shortest cycle grows with $n!$, where n is the number of nodes in the graph. It is critical, then, to develop algorithms to solve these problems more efficiently. Investigations have demonstrated that both algorithms and network flow problems are closely related to the colouring problems mentioned above [34].

1.2 Information Theory

Applied mathematics has long been used to solve problems in information theory. The discipline of information theory was formalised in 1948 by Claude Shannon [31], though existing communication methods had used the ideas for some years previously. Information theory is based on probability theory and encompasses a number of scientific and engineering disciplines, and as such is a fundamental part of many modern developments and inventions, from mobile phones and compact discs to the study of black holes [4]. Network information theory is concerned with the study of networks — or acyclic, directed graphs — and the transmission of information in these networks.

At the heart of Shannon's work is coding theory, dealing with the transmission of information across a noisy channel. The codes are separated into two types; source coding and channel coding. Source coding seeks to compress the data so that fewer bits are required to transmit it. The fewest number of bits required to transmit a piece of information is given by its entropy (see Section 2.5). On the other hand, channel coding adds redundant data to minimise the effect of the noise. The redundant data are used as error correcting codes. The paper also states that reliable communication is possible (i.e., errors may be made arbitrarily small) over a noisy channel if the rate of communication is less than the capacity of the channel. This paper is concerned with both types of coding.

1.3 Network Coding

Recently a new field of information theory, known as **network coding**, has emerged. The concept was introduced by Ahlswede *et al.*, and they give examples as to the benefits of using network coding to increase network throughput. The findings have significantly affected the discipline of information theory by showing that traditional routing methods do not suffice when attempting to achieve the optimal flow of a network [1]. Work on this new area has advanced significantly over the past few years (see, for example, [1, 16, 17, 24, 12]).

Network coding has emerged as a result of the theories above — initially developed for satellite communications [35], the tools introduced were new to multi-user information theory. Network theory uses graphs to represent relationships between discrete objects, and hence coincides with graph theory. Network coding builds on network routing, a problem in network theory, and shows that network routing is not sufficient (see Section 2.2).

1.4 Origins of Work

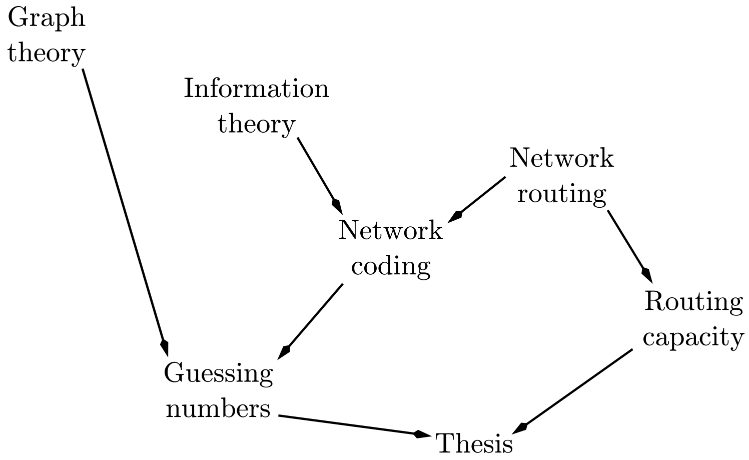


Figure 1.1: Each topic builds upon another previous area of work. This diagram shows the association between topics covered, and how they lead to the work in this thesis.

Figure 1.1 shows how the topics introduced above are relevant to the work in this thesis. Network coding builds on ideas from traditional network routing (Section 2.1) and combines it with information theory. This thesis combines the new concept of **guessing numbers** [25, 26] (Section 2.3) and an existing subfield of network routing known as **routing capacity** (Section 2.4). Each step can be seen as using the tools from one discipline to improve the techniques used in the other. For example, this thesis aims to use the tools used in routing capacity to improve on the concept of guessing numbers.

1.5 Criteria for Success

As mentioned at the beginning of this chapter, this thesis is concerned with producing a new or improved technique for dealing with information flow problems. Ideally, this technique should help to answer some of the open questions in the field of network coding. The criteria for success are outlined below and are evaluated in Chapter 5.

- Study and description of existing methods within network coding and network routing capacity;
- development of a new method for studying network flow problems based on existing techniques;
- abstraction and formalisation of a modified guessing game based on the new method;
- study and evaluation of cases in which the new game is more successful than the existing game.

1.6 Thesis Outline

Chapter 2 contains the literature review, defining some of the key concepts and discussing work in the fields of information theory and network coding. It describes how the shortcomings of traditional routing used in existing network technologies have brought about these new fields. In addition, it gives more detailed information on some of the open and solved problems. Specific attention is paid to three papers by Riis *et al.* [25, 26, 8], the only previously submitted papers detailing the concept on which this thesis expands. The background study required to understand the methods used in later chapters is reviewed here.

Chapter 3 expands on some of the open problems mentioned in Chapter 2, together with other problems that were approached. As well as expanding and improving on existing work, it covers definitions of new concepts in the field. New definitions and propositions are provided here. In particular, the new concept of a “routing guessing game” allows certain problems in network information flow to be viewed with a new approach.

Chapter 4 provides examples and applications of these new ideas. In addition, the main theorems required to validate this new concept are proved here. Specifically, the existence of a non-integer guessing number is proved using a specific guessing strategy on the pentagon.

Chapter 5 provides a critical evaluation of the issues tackled and overcome throughout the writing of the thesis. It discusses whether the problems raised at the beginning of the thesis have been sufficiently addressed. The criteria for success are each evaluated in turn to provide a complete review of the work undertaken. The achievements of the thesis as well as its impact on the field of information theory are discussed in this chapter, as well as some possibilities for further work.

Chapter 6 provides conclusions from the ideas discussed in the thesis by giving a general review of the work undertaken. Open questions that have either arisen from the work contained herein or from previous work are listed in this final chapter.

Chapter 2

Literature Review

The aim of the literature review is to introduce the problem domain. This chapter gives background information on the existing work in the field, noting the research most important and relevant to this project. It outlines the study into the connections between network flow problems and their related problems in graph theory, and suggests areas for expansion on the surveyed literature.

2.1 Traditional Routing

Traditional routing involves choosing paths in a network for data transmission. It treats messages sent along the network as packets, which cannot be divided into smaller units, or otherwise altered. This method is used in a large variety of real-world networking applications, such as traffic management, telephone networks and computer networks.

Routing may be controlled manually by configuring specific routing tables to direct packets along the most direct route, or in a more autonomous manner by assigning certain routing protocols allowing the packets more freedom in the route taken. These different methods of forwarding information are known as schemes (Figure 2.1), and vary in techniques —

from delivering the information to one specific node, to delivering the information to all adjacent nodes regardless of whether they requested it.

Unicast (Figure 2.1(a)) is the most common routing scheme used in current telecommunications networks. It involves the sending of the packets to exactly one destination. Broadcasting (Figure 2.1(b)) is the opposite of unicasting; the packets are forwarded to every node in the network. Multicasting (Figure 2.1(c)) is a more sophisticated approach, similar to broadcasting. In multicasting, packets are sent to a number of adjacent nodes determined by some calculation of a minimum spanning tree. Anycasting (Figure 2.1(d)) is slightly different in that it has a number of possible nodes to which it may forward the packet, but only one node is ultimately chosen to receive it.

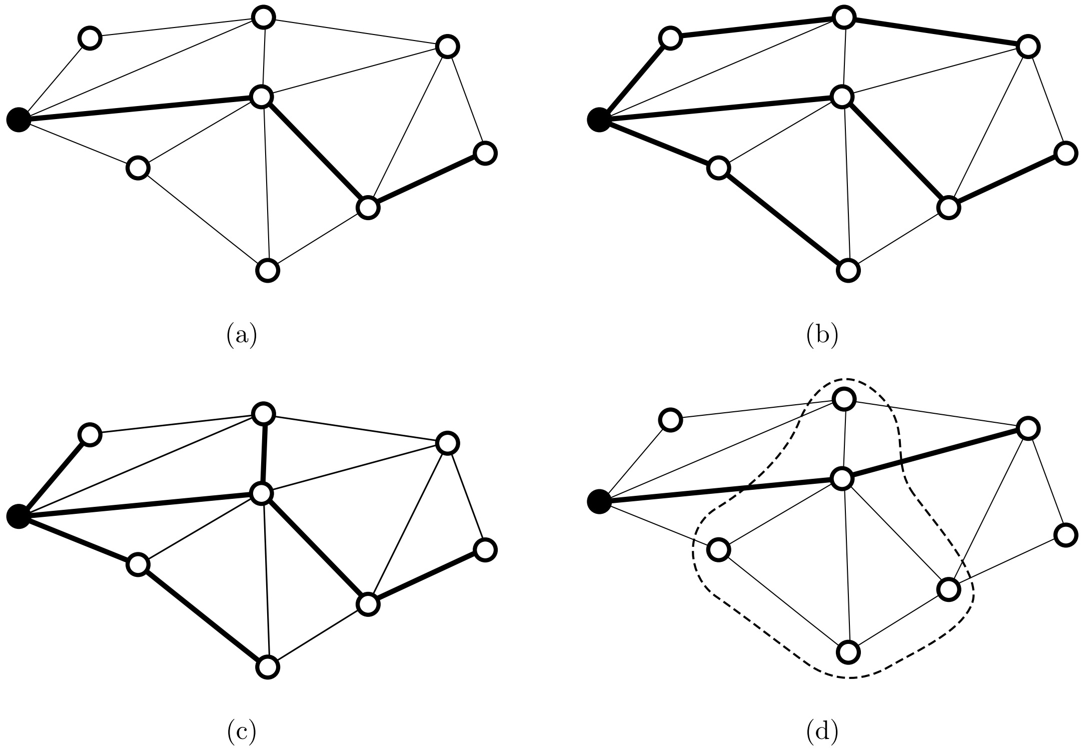


Figure 2.1: Examples of the different schemes used in packet forwarding: (a) unicast, (b) broadcast, (c) multicast and (d) anycast. Packets are forwarded from the black node along the bold edges.

Network routing is an enourmously successful data transmission method, seen by its use in so many existing technologies. However, its methods may limit the transfer rate in certain cases — for instance, bottlenecks may appear in certain network topologies. These may be due to the fact that information is viewed as a discrete unit, something which may not be broken down into smaller components. This is akin to viewing the atom as the building-block of the universe; whilst it was once viewed as the most advanced model, technology showed that it was possible to break it down into yet smaller components, and that different combinations of these components yielded different atoms. To eliminate these bottlenecks, a similar paradigm shift was required.

2.2 Network Coding

As mentioned above, traditional routing is used in a number of information networks, from communication in a processor to communication across the Internet. However, there are disadvantages to using these techniques; a simple network can be constructed where no routing method can achieve optimal flow.

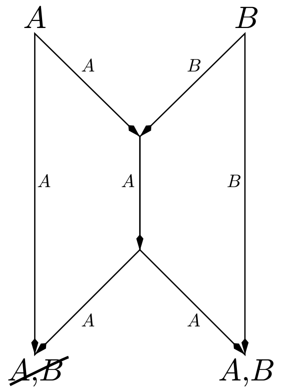


Figure 2.2: The butterfly network, first introduced by Ahlswede *et al.* [1]. Because of the central bottleneck, routing is insufficient: the left sink node fails to resolve both A and B.

In the so-called "butterfly" network in Figure 2.2, first introduced in [1], the two sink nodes at the bottom each need to receive both messages A and B from the two source nodes at the top. The edges between each vertex denote paths or channels along which data may flow, and the arrows signify the direction in which it may travel. Assume that each channel can transmit only one message per unit time. If traditional routing is used, then only trivial functions may be used. This means that the middle channel cannot transmit both A and B at the same time, resulting in a bottleneck. If the central channel transmits only A, then the left sink node does not resolve both messages. If this particular network set-up appeared in a real-world situation, for instance client computers requesting information from two separate servers, then one client would be forced to wait until all the information had been transmitted to the other client before completing its own requests.

Network coding is an emerging field of information theory in which messages are viewed as information. This means that any node in the network may replicate or alter the messages. In other words, the outgoing edges from a given node will transmit some function of the data received on the incoming edges. For instance, if on the network in Figure 2.2 the exclusive disjunction $A \oplus B$ is sent along the middle channel, the sink nodes can resolve both messages (Figure 2.3). To see this, node that the left sink node receives A from the left source node, and $A \oplus B$ from the central channel. Since $A \oplus B \oplus A = B$, it is possible to compute both original messages A and B, as desired. The process is similar for the right sink node.

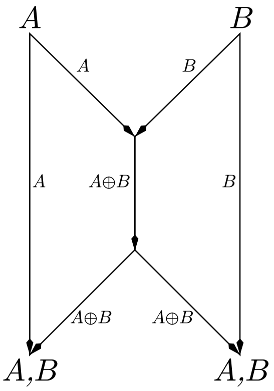


Figure 2.3: Using network coding allows both sources to successfully resolve both messages.

In this manner, the entire method of traditional routing can be viewed as the trivial case of network coding when only the trivial functions are used; in other words, the output of each function is equal to one of its inputs.

In general, if the message $m_{x,y} = f(x,y) \in A$ (for some finite alphabet A) is sent along the middle channel, then the problem can be solved if and only if $f(x,y)$ is a Latin function [28].[*] The main aim of using network coding is to find a set F of such functions, called a *solution*, to allow all of the sink nodes to be satisfied. The network is said to be *solvable* if F exists.

If a solution uses only trivial functions it is known as a *routing solution*, and if (as in the butterfly network) the output of each function is a linear combination of its inputs, it is known as a *linear solution* [9]. To refine these definitions even further, if the messages passed along the nodes are scalar quantities, then a solution is known as a *scalar linear solution.* Often, however, it is useful to consider message vectors, or blocks of scalar messages; in this case, a solution is called a *vector linear solution.*

The problem of the butterfly network above can be converted into a problem in graph theory by representing the network as a circuit, and identifying nodes. Such a representation is called *circuit representation.* Converting the networks in this manner allows all of the tools at the disposal of the graph theorist to be used on problems in network coding. In this representation, all of the computations in the network are carried out in the nodes. Circuit representation is a slightly different (but mathematically equivalent) representation of a network coding problem which serves to be a slightly more useful (if less accurate) method of depicting the network coding problem [25]. A source node s is identified with the output node(s) (or sink nodes with respect to s) attempting to compute the information transmitted from s. This particular problem results in K_3, or the complete graph on 3 nodes [25]. The proof of this conversion is illustrated in Figure 2.4. The motivation behind this conversion is to have an equivalent problem that can be expressed in graph theoretic terms [25, 26].

Network coding has shown its usefulness in a number of areas for concern; one important example is its improvements to wireless networking. Increases in throughput and energy

[*] The function $f(x,y)$ is Latin if for each $(x,y) \in A^2$, the functions $m_y : A \to A : x \mapsto m_y(x) = f(x,y)$ and $n_x : A \to A : y \mapsto n_x(y) = f(x,y)$ are bijective maps.

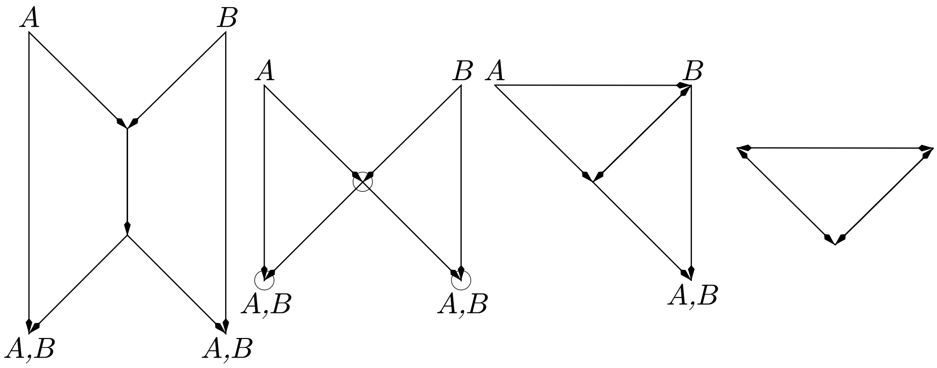


Figure 2.4: Start with Figure 2.2. First, the network is represented as a circuit information flow problem, with computation at the nodes. Then the sink nodes are identified with the source nodes, resulting in K_3.

efficiency are evident in wireless network topologies as well as other networks [29]. With wireless access points becoming so popular, and with their limited range requiring intermediate access points or mesh networks, there is growing concern for the security of data broadcast via radio waves. Network coding has been shown to be a useful approach to prevent eavesdropping on secure communications [7]. As mentioned above, information theory is primarily concerned with transmission of data over noisy channels, and network coding has been shown to be a successful alternative to other error-correction methods [18].

Network coding has also been considered for peer-to-peer (P2P) content distribution, and is even believed by some that it may improve performance over P2P networks, but Chiu *et al.* conclude that there is little or no coding advantage, due in part to the time and resources required to decode the delivered information [10].

2.2.1 Linear versus Non-Linear Coding

Previously, it has been conjectured by a number of authors (for example, [22, 3]) that linear network coding is optimal for an arbitrary network topology. Indeed it is known that every solvable multicast network does have a scalar linear solution provided the alphabet is a sufficiently large finite field [20]. Moreover, any solvable multicast network has a linear solution in some vector dimension [24]. However, it is also known that this result does not extend to arbitrary networks (see, for instance, [19]). Dougherty *et al.* [12] provide a solvable network that does not have a linear solution over any finite field, or indeed over any module.[†] Riis and Ahlswede introduce other counter-examples, but suggest that these are so far only “rare isolated incidences” [28].

2.3 Guessing Numbers

The concept of a **guessing number** was developed by Søren Riis [25, 26] to investigate the solutions of particular network flow problems. The concept was notably used to provide a counter-example [26] to the conjectures made by Leslie Valiant [32]. To understand the meaning of the guessing number of a graph, consider the following problem:

Imagine a game consisting of n players, each of whom has an s-sided die (for $s \in \mathbb{N}$, $s > 1$), where each side is labelled in the usual manner from 1 through to s.[‡] The dice are rolled simultaneously in such a way that no player knows the value of their own die. The task for the players is to guess the value of their own die.

1. What is the probability that all n players can correctly guess their die value?
2. Now assume that each player knows the value of every die except their own. What is the probability that they all guess correctly now?

In the first question, each player acts independently and, since the probability that each player is right is $1/s$, all n players guess correctly with probability s^{-n}.

[†] A module is a concept in abstract algebra that generalises the idea of a vector space. The notion of a vector space being expressed over a field is also generalised; a module is expressed over a ring.

[‡] In accordance with most papers in computer science, the set $\mathbb{N}$ of natural numbers is defined as the non-negative integers, that is $\mathbb{N} = \{n \in \mathbb{Z} : n \geq 0\}$.

At first glance, it may seem that allowing the players to see the other players' dice cannot improve their odds, since the information held on other dice in no way alters the value of one's own die. However, the players may adopt a "strategy" that uses the information provided to them from the others. There are many strategies that the players may adopt to utilise this information, but the best (or *optimal*) strategy is for the players to assume beforehand that the sum of all the dice is congruent to a modulo s. The player then calculates the sum total of the other dice, T, and "guesses" their value to be $a-T$ modulo s. Clearly if the result is 0, then the player must guess s, since no side is labelled 0. This strategy ensures that all players "guess" the value of their own die with probability $1/s$ if (and only if) one player guesses correctly. Note that since any one player can only guess correctly with probability $1/s$, this gives an upper bound on the guessing number, and hence this is an optimal strategy.

The game (or class of games) mentioned above is a subclass of a larger class of co-operative games, known as guessing games. A guessing game is a method of representing a problem from $\mathcal{C}_{mu}$, the class of multiple-unicast information flow problems [25]. A multiple-unicast problem is a network with n source nodes $\{i_r\}$ and n sink nodes $\{o_r\}$, where each output node o_r must resolve the message m_r from the corresponding input node i_r. The game is played on a directed graph $G = (V, E)$ where V is the set of vertices, and $E \subseteq V^2$ the set of edges between those vertices. In the case of a directed graph, there is an edge from $v \in V$ to $w \in V$ if and only if $(v, w) \in E$ — in other words, E is a collection of ordered pairs.[§] Each vertex represents a player in the game, and edges represent communication paths between the players. If the graph is directed, then the communication may only travel along the edge in one direction, for instance from the head to the tail. Note that all graphs in this class are simple, in that they do not contain any self-loops. In the game, a self-loop would correspond to a player immediately and explicitly knowing the value of their die once thrown.

[§] In the undirected case, E is a collection of unordered pairs, or sets of cardinality 2; an undirected edge between v and w is denoted $\{v, w\}$.

Definition 2.1 (Riis [25])**:** A *guessing game*, denoted GuessingGame(G, s), is played as follows: each player $v \in V$ is randomly assigned a value $m_v \in A$, where $A = \{1, 2, \ldots, s\}$. The players then communicate their values to adjacent players, i.e. to all players w with $(v, w) \in E$. Node w receives a set of values $M_w = \{m_v \in A : (v, w) \in E\}$. From the information received, each player must resolve their own value. Assuming that the players have agreed on a strategy, the probability that all players guess correctly can be calculated. Explicitly, a *strategy* or *protocol* is a set of functions $f_w : A^{|M_w|} \to A$ for each $w \in V$. A function f_w denotes player w's guess.

There are very many co-operative strategies for the players to adopt: in general, there are $s^{\sum_{v \in V} s^{|M_v|}}$ strategies. Each strategy has associated with it a probability determining the likelihood of success of that strategy. Clearly the strategy that yields the maximum probability is optimal.

The alphabet A from which the values are assigned need not be as simple a structure as described above; it could, for instance, be a field, a ring, or (as was used in one proof in [25]) a commutative group; suitable algebraic structures must include some notion of linearity. However, for most cases the description provided above is adequate.

Just like the butterfly network considered previously, the example guessing game corresponds to a complete graph; every player is connected by an edge to every other player. Therefore, the total number of guessing strategies available to the players of this game is $s^{ns^{n-1}}$. As shown above, the optimal strategy for players in this situation gives a probability of success of $1/s$, a factor s^{n-1} better than uncoordinated, random guessing. This gives us a *guessing number* of $n - 1$, the exponent of the quotient of the probabilities.

The formal definition of the guessing number of a graph is as follows:

Definition 2.2 (Riis [25])**:** Let $G = (V, E)$ be a graph where each vertex $v \in V$ corresponds to a player of the guessing game. G has *guessing number* $k = k(G, s)$ if there is a strategy that ensures their success with probability $s^{k-|V|}$. In other words, the strategy succeeds with probability s^k times higher than independent, random guessing.

The guessing number is often independent of s. Also, rather surprisingly, for many graphs the guessing number is also an integer. Though it is noted in [25] that there exists a directed graph where the guessing number depends on s, it is left as an open question as to whether the guessing number of an undirected graph is always an integer. Later, in Chapter 4, it is shown that there exists an undirected graph where the guessing number depends on s and is never an integer.

The guessing game G_N can be said to be a graph representation of a specific information network (or subclass of networks) $N \in \mathcal{C}_{mu}$. The mathematically correct representation of the network is called "standard representation". In this representation (and throughout this thesis) a network N is a directed acyclic multi-graph, where source nodes have in-degree 0, and sink nodes have out-degree 0. Each source node i_r is associated with exactly one message $m_r \in \Gamma_{var}$ (this notation is used for consistency with [25]). Each outgoing edge from a given node v transmits the same function $f_v : A^d \to A$, where d is the in-degree of v. Each sink node needs to resolve some subset of the variables in Γ_{var}.

The standard representation of the circuit information flow problem N is converted to the directed graph G_N by introducing a vertex for each variable or function assigned to an edge or node in N, and identifying each input node i_r with its corresponding output node o_r for each $r \in \{1, 2, \dots, n\}$.

If the guessing game has some meaning in information flow problems, then it is reasonable to assume that the guessing number has a similar generalisation. Before this generalisation is explained, some definitions are required.

Definition 2.3 (Riis [25])**:** The *global success rate* of N with respect to a set of coding functions F is the probability that all outputs correctly determine their messages, given that the inputs are selected randomly from a finite alphabet A and with independent probability distribution. It is denoted $p(N, s, F)$ where $s = |A|$.

The *maximum global success rate* $p(N, s)$ is the supremum of all global success rates $p(N, s, F)$ for any choice of coding functions F. Since the set of functions F is finite, $p(N, s) = p(N, s, F)$ for some F.

Definition 2.4 (Riis [25])**:** The *source transmission bandwidth* of the information network N over alphabet A is defined as $k(N, s) = \log_s(p(N, s)) + n$, where n is the number of source nodes of N.

This definition generalises the concept of a guessing number [25]. The guessing strategies used in the guessing game correspond exactly to the network coding functions in the network flow problem.

As proved in [26], a network flow problem N with n input/output nodes has a solution over alphabet A with $|A| = s$ elements if and only if the corresponding graph G_N has guessing number $k(G_N, s) \geq n$.

In [25], it was shown that if a network N' appears from a network N by application of r split moves and t inverse split moves, and N has global success rate p, then N' has global success rate ps^{t-r}. A "split move" is a simple operation applied to an inner (not an input or output) node $w \in V$ in a network $N = (V, E)$. The operation copies the node w, creating two nodes w_{input} and w_{output}. Explicitly, split : $\mathcal{C}_{mu} \to \mathcal{C}_{mu} : N \mapsto N'$ where $N' = (V', E')$, $V' = V \cup \{w_{\text{input}}, w_{\text{output}}\} \setminus \{w\}$ and E' contains all the edges of E except those connecting to w, plus $(w, u) \in E \Rightarrow (w_{\text{input}}, u) \in E'$ and $(u, w) \in E \Rightarrow (u, w_{\text{output}}) \in E'$, each counted with multiplicities. Since the split and inverse split moves are applied only to the inner nodes of N, in other words those nodes in $V \setminus \{i_1, i_2, \ldots, i_n, o_1, o_2, \ldots, o_n\}$, the source transmission bandwidth remains unchanged under them, and hence $k(N, s) = k(N', s)$.

It is worth noting that if N' can be realised from N by applying a certain number of split and/or inverse split moves, then the graphs G_N and $G_{N'}$ are identical. This is because creating the graph G_N is effectively the same as applying as many inverse split moves as possible to N. In other words, G_N is invariant under split moves on N.

2.4 Routing Capacity

Although the messages transmitted in a network are scalar quantities, it can be useful to consider blocks of these quantities as vectors. If the capacity of each edge in the network is of the same dimension as these message vectors, then any vector solution corresponds directly to a scalar solution where the alphabet consists of vectors. *Fractional coding* refers to the case where edges vectors and message vectors differ in dimension.

Cannons *et al.* [9] look at fractional coding for networks in the case of routing, referred to as *fractional routing.* The **routing capacity** ϵ of a network is "the supremum of all possible fractional message throughputs achievable by routing" [9]. In other words, it is the highest possible ratio of message dimension to edge capacity that can be obtained using a fractional routing solution. A (k, n) *fractional routing solution* is a solution where message vectors are of dimension k and edges have capacity n. Routing capacity is essentially a method brought about by the failure of other methods to provide the best possible data transmission for certain networks, and it is this advantage over linear network coding that will be exploited in Chapter 4.

The routing capacity of every network was shown to be both achievable and rational [9]. Furthermore, it is shown that for each $r \in \mathbb{Q}$, $r > 0$, there exists a network whose routing capacity is equal to r, and that any $r \in (0, 1] \cap \mathbb{Q}$ is the routing capacity of a solvable network. The concept of routing capacity is generalised to *coding capacity*, with the inferred definition. The coding capacity of a network is proven to be independent of the underlying alphabet.

The butterfly network in Figure 2.3 was shown to have a linear coding solution, namely if the central channel transmits the function $A \oplus B$. In Figure 2.5, if both A and B are to be transmitted along the central edge $(3, 4)$, the edge capacity n must be at least twice the vector dimension k, hence $2k \leq n$ and $\epsilon = k/n \leq 1/2$. Now by transmitting A along edges $(1, 3)$, $(1, 5)$ and $(4, 6)$, B along edges $(2, 3)$, $(2, 6)$ and $(4, 5)$, and transmitting (A, B) along $(3, 4)$, there is a fractional routing solution to the network, and $\epsilon \geq 1/2$. From above, the

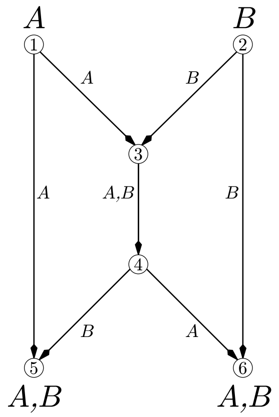


Figure 2.5: The butterfly network, which has been shown to have a linear coding solution, has routing capacity 1/2.

routing capacity of the butterfly network is $\epsilon = 1/2$ [9].

A slightly more complicated example of routing capacity is the network $\mathcal{N}$ shown in Figure 2.6. In [12], it was shown that $\mathcal{N}$ has no linear solution for any vector dimension over a finite field with odd cardinality (it was shown in the same paper, however, that it has a scalar linear solution over any ring of cardinality 2). The three messages A, B and C are transmitted by 1, 2 and 3 respectively, and are demanded by 14, 13 and 12 respectively. The following proof is reproduced from [9], and shows that the $\mathcal{N}$ has a coding solution with routing capacity 2/3.

The edges $(1, 12)$, $(3, 9)$ and $(7, 14)$ can be immediately ignored, since none of the edges can transmit any useful information to their sink nodes. For example, node 7 receives information about the messages B and C, but node 14 is only concerned with message A. Ignoring these edges implies that $(4, 6)$ and $(5, 7)$ carry all the information from the sources to the sinks. Since there are 3 messages to be transmitted along 2 edges, this quickly imposes an upper bound of $\epsilon = 2/3$ as $3k \leq 2n$ for any k, n. Now let $(k, n) = (2, 3)$ — so message have 2 components and edges have capacity 3 — and transmit the messages as follows:

$$\begin{aligned}
(1,4) = (11,14) &= (A_1, A_2) \\
(2,4) = (11,13) &= (B_1) \\
(2,5) = (10,13) &= (B_2) \\
(3,5) = (10,12) &= (C_1, C_2) \\
(4,6) = (6,9) &= (A_1, A_2, B_1) \\
(5,7) = (7,8) &= (B_2, C_1, C_2)
\end{aligned}$$

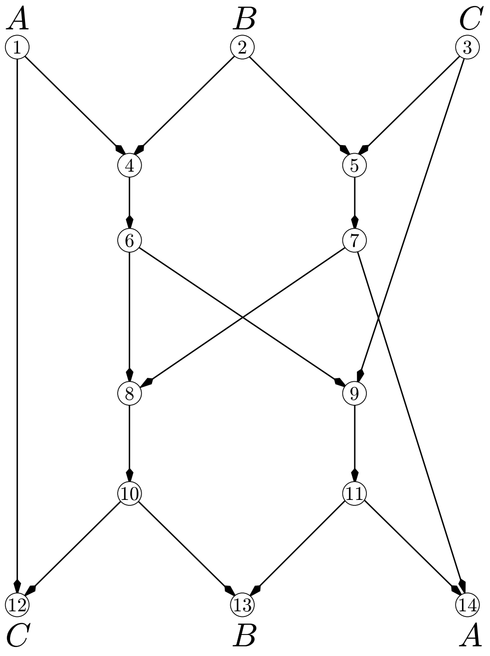


Figure 2.6: The network $\mathcal{N}$, reproduced from [9], has routing capacity 2/3.

This gives a fractional routing solution to $\mathcal{N}$, meaning $\epsilon \geq 2/3$ and hence $\epsilon = 2/3$. Note that when omitting the ignored edges together with $(6,8)$ (which also remains unused), we obtain a satisfyingly bisymmetric subgraph of $\mathcal{N}$, which goes some way to explain the symmetry of the routing solution.

As mentioned in Chapter 1, even though a number of questions are answered in [9], some questions are left open; for instance, whether the coding capacity is achievable for every network, and if algorithms exist for computing the coding capacity of a network. It has since been shown that the coding capacity of an arbitrary network is not always achievable by the network, unlike the case of routing capacity [13].

2.5 Information Entropy

Shannon's classical information inequalities [31] will be used to calculate the guessing numbers of cycles in Chapter 4. Shannon's inequalities are a number of rules governing **information entropy** $H(X)$, the measure of uncertainty of a particular random variable.

$$\begin{aligned} H(X) &= \mathrm{E}(I(X)) \\ &= \sum_{i=1}^{n} p(x_i)\ \log_2 (1/p(x_i)) \\ &= -\sum_{i=1}^{n} p(x_i)\ \log_2 (p(x_i)) \end{aligned}$$

Here, $I(X)$ is the self-information of X, or the information associated with the outcome of X, and $p(x_i) = \Pr(X = x_i)$. In the same manner, entropy can be thought of as the amount of self-information that is missing when the value of X is unknown.

A fundamental result in information entropy is Gibbs' inequality; that is,

$$\sum p_i \log_2 p_i \geq \sum p_i \log_2 q_i$$

for probability vectors $P = (p_1, \ldots, p_n)$, $Q = (q_1, \ldots, q_n)$.

A simple example would be a weighted die that can be altered to offer bias to different outcomes (Figure 2.7). The more bias toward a certain outcome, the less uncertainty there is in the result of the roll. If the die is weighted such that every roll produces a 6, say, then the roll provides no information, and the entropy of the outcome is 0. If the dice has no bias, then the uncertainty is maximised, and the entropy is $\log_2 6 \approx 2.58$. When logs are taken to base 2, entropy is measured in bits. In the case of statistical information, a "bit"

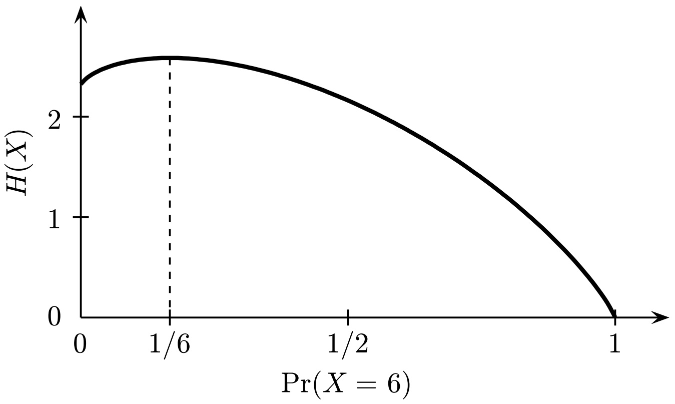


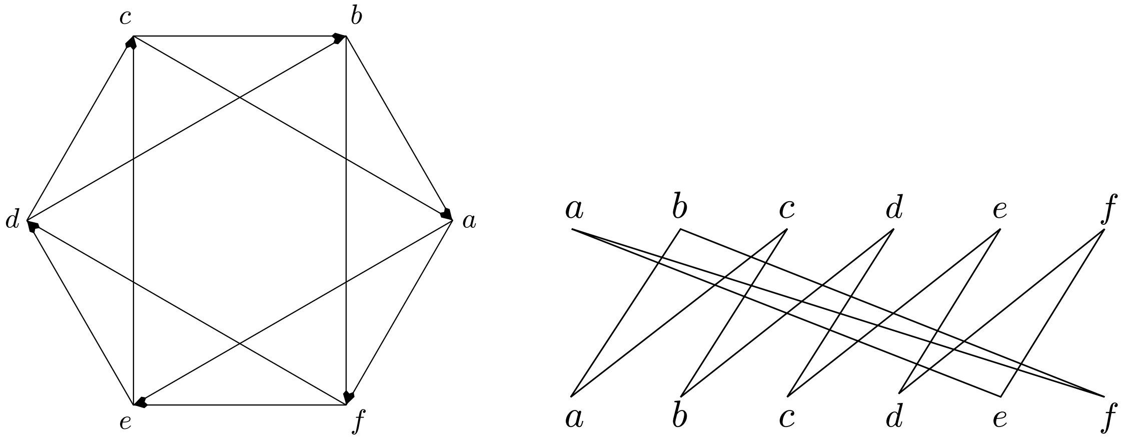
Figure 2.7: The graph of entropy shows that less information is provided when the probability of rolling a 6 approaches 1. The entropy function is maximised when the probability of throwing a 6 is 1/6, where the die is fairest.

is slightly different from the storage unit used in digital computers. In digital storage, a bit is a discrete quantity taking only one of the values in $\{0, 1\}$. In terms of entropy, a variable with n bits of information can theoretically, and on average, be stored on a computer using n bits, though this value is a minimum. The value indicates the number of bits of actual information contained in the result, and as such provides a lower bound on any lossless compression of the data.

This can be explained by the notion of "redundancy". Ordinary text in the English language has a great redundancy — certain phonemes or letters that are not strictly required to convey the original message. Take, for example, an advertisement for training in shorthand notation: "*f u cn rd ths msg u cn hv a hi pyng jb*" [15]. Compare also with telex speak and SMS language, where similar shorthand is used to convey the message in the limited number of characters available. If, as with ASCII encoding, each letter has an entropy of exactly 7 bits when chosen at random, English text has an entropy of between 1 and 1.5 bits per letter; it is quite easy to determine the next letter in the sequence [30]. This exact principle can be used in cryptography to break simple substitution ciphers, by working out the frequency of appearance of each letter.

2.6 Bipartite Graphs

It is possible to rearrange problems in network flow to a bipartite flow problem. In this arrangement, there are two copies of each node, one copy having the same in-degree as the original node, and the other having the same out-degree as the original node. This is the result of applying the split move mentioned in Section 2.3 to *all* inner nodes in a graph N to get the bipartite graph B_N with no inner nodes [25]. The result is shown in Figure 2.8(b), where the the edges are implicitly directed from top to bottom. As mentioned above, the two are different representations of mathematically equivalent problems.



(a) Network flow problem

(b) Equivalent bipartite flow problem

Figure 2.8: The graph in (a) is converted to the bipartite problem in (b) by making two copies of each node.

Bipartite graphs are the heart of a different game introduced in the same paper as the guessing game, known as PublicChannelGame(G, P). In this game, each node has access to a number of public messages $p \in P$, $p : A^n \to A$ as well as all the messages received from its own incoming vertices. In the case of a guessing game, it is always possible to substitute a suitable public channel for the guessing part of the game if and only if the solution is linear [25].

2.7 Matchings and Fractional Matchings

A *matching* M of an undirected graph G is a subset of the edges E of G such that no two edges in M share a common vertex (Figure 2.9(a)). A *perfect matching* is a matching that covers every node in the graph (Figure 2.9(b)).

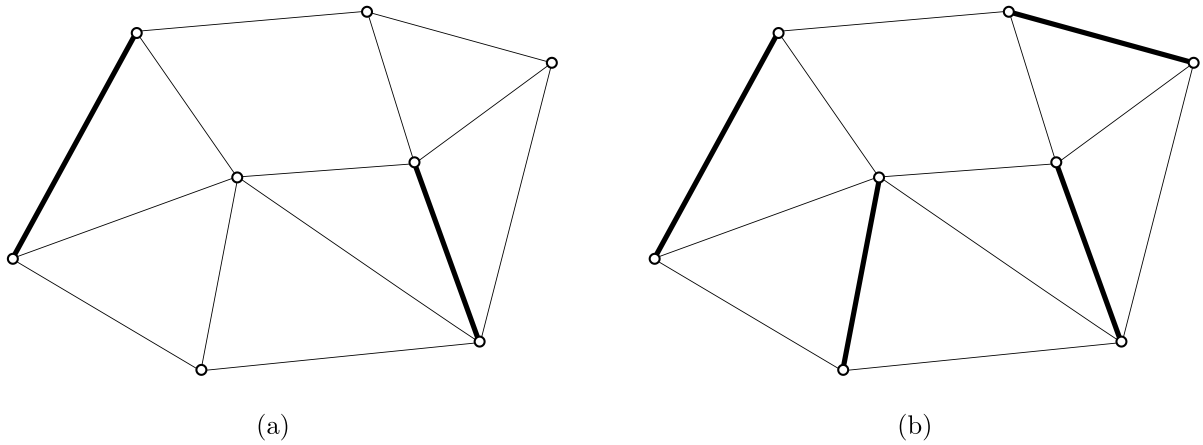


Figure 2.9: A matching (marked by bold lines) (a) is a set of pairwise non-adjacent edges, and a perfect matching (b) is a matching that covers all the vertices.

A fractional matching of a graph G is a function $f : E \to [0, 1]$ such that for any $u \in V$, $\sum_{v\,:\,\{u,v\}\,\in E} f(\{u, v\}) \leq 1$ [21]. The *fractional matching number* of a graph is the supremum of all fractional matchings, sup $\sum_{\{u,v\}\,\in E} f(\{u, v\})$.

A *perfect fractional matching* is observed when each of the sums is exactly equal to 1. As with matchings, fractional matchings only apply to undirected graphs. Here, the weightings refer to the capacities of the vertices.

2.8 The Fractional Matching Problem

The Fractional Matching Problem is a well-known problem in graph theory:

$$\begin{aligned} \text{Maximise} \quad & \sum_{e \in E} x_e \\ \text{such that} \quad & \forall v \in V \sum_{e \ni v} x_e \leq 1 \\ & x_e \geq 0. \end{aligned}$$

The above problem has been stated in many forms. It was shown by Bourjolly and Pulleyblank (see, e.g., [6]) that there exists a maximum solution to the fractional matching problem that is half-integral, with each edge having a capacity in $\{0, \frac{1}{2}, 1\}$. The theorem is stated below. The proof is outlined in two stages and follows from Theorem 2.7.

Theorem 2.5: For every graph, there exists a fractional matching that is half-integral, where each edge in the matching has its weighting in $\{0, \frac{1}{2}, 1\}$.

Let $G(V, E)$ be a graph, and denote by $c_e \in \mathbb{R}$ the weighting assigned to the edge $e \in E$. Denote by $c = (c_e : e \in E)$ the vector of these edge weights.

For each edge e, define a variable $x_e \in \{0, 1\}$. Now denote by x^J the incidence vector representing $J \in E$, where

$$x_e^J = \begin{cases} 0 & \text{if } e \notin J, \\ 1 & \text{if } e \in J. \end{cases}$$

Let $\mathbb{R}^E$ be the set of all real-valued vectors indexed by E. Let $x(J) = \sum (x_j : j \in J)$ for any $J \subseteq E, x \in \mathbb{R}^E$. Let $\mathcal{M} = \{x^M \in \mathbb{R}^E\}$, where M is a perfect matching in G. $\mathcal{M}$ is a (finite) set of $(0, 1)$-valued vectors, the convex hull of which is a polytope[¶] in $\mathbb{R}^E$. It is called the *perfect matching polytope*, and is denoted $PM(G)$.

[¶] A *polytope* is the generalisation of a polygon in two dimensions, a polyhedron in three dimensions, and so on. A *convex hull* for a set X is the minimal convex set containing X.

Since every polytope is the solution set of a finite system of linear equalities [33], we attempt to find such a system for $PM(G)$.

Theorem 2.6 (Edmonds [14])**:** For any bipartite graph $G(V, E)$, $PM(G)$ is the set of all $x \in \mathbb{R}^E$ satisfying

$$x \geq 0, \tag{2.1}$$

$$x(\delta(v)) = 1 \quad \forall\, v \in V. \tag{2.2}$$

Proof: Denote by P the set of solutions for (2.1) and (2.2). Because the incidence vector $x^M \in P$ for any perfect matching M, $\mathcal{M} \in P$ and therefore $PM(G) \in P$.

Now suppose $\exists \tilde{x} \in P$ which cannot be expressed by convex combinations of members of $\mathcal{M}$. Choose $\tilde{x}$ such that $F = \{e \in E : 0 < \tilde{x}_e < 1\}$ is minimal. $F \neq \emptyset$. Since $\tilde{x}$ satisfies (2.2), no node can be incident with just one member of F, so F contains the edges of some cycle C in G. C has an even number of edges, since G is bipartite.

Now let $d \in \mathbb{R}^E$ be a vector which is 0 for all edges $j \notin C$, and alternately 1 and -1 for those edges in C. For any $\Delta \in \mathbb{R}$, $\tilde{x} + \Delta \cdot d$ will satisfy $(\tilde{x} + \Delta \cdot d)(\delta(v)) = \tilde{x}(\delta(v)) = 1$. However if $|\Delta|$ is too large, $\tilde{x}+\Delta\cdot d \geq 0$ will not hold. Choose Δ_1 and Δ_2 as large as possible so $x_1 = \tilde{x} + \Delta_1 \cdot d \geq 0$ and $x_2 = \tilde{x} - \Delta_2 \cdot d \geq 0$. Then $x_1, x_2 \in P$ and, since each has fewer fractional components than $\tilde{x}$, each is in $PM(G)$. However, $\tilde{x} = (\Delta_2 x_1 + \Delta_1 x_2)/(\Delta_1 + \Delta_2)$ and hence $\tilde{x}$ is a convex combination of members of $PM(G)$ — we arrive at a contradiction.

□

Let $G(V, E)$ be any graph. Consider the following polyhedron:

$$x_e \geq 0 \quad \forall\, e \in E$$

$$x(\delta(v)) = 1 \quad \forall\, v \in V$$

This is called the *fractional matching polytope* of G, and is denoted by $FM(G)$.

Theorem 2.7 (Pulleyblank [23])**:** Let $x \in FM(G)$. Then x is a vertex if and only if the weighting $x_j \in \{0, \frac{1}{2}, 1\}$ $\forall j \in E$. Furthermore, the edges j for which $x_j = \frac{1}{2}$ form node-disjoint odd cycles.

Proof: Let $\bar{x}$ satisfy the above system, and let $c_j = -1$ if $\bar{x}_j = 0$, $c_j = 0$ if $\bar{x}_j > 0$. Then $\bar{x}$ is the unique member x of $FM(G)$ for which $cx = 0$, since $cx < 0$ $\forall x \in FM(G) \setminus \{\bar{x}\}$.

The necessity of the proof follows from Theorem 2.6. □

The significance of this is that when the length of the messages is even, the guessing number is half-integral, as shown by linear programming. We use this dependence on the length of the messages in Chapter 4.

2.9 Summary

This chapter reviewed the key ideas from the existing literature. It explained the existing methods used in traditional routing and why network coding improves upon those methods. The concepts of a guessing number and the guessing game were formally defined, and examples were provided to explain the real-world meaning of the concepts. It also reviewed work on network routing capacity in order to fully understand how these ideas could be included into the guessing game.

Shannon's information entropy and some of Shannon's inequalities are also explained, as these will be key to the solutions provided in Chapter 4. Other existing ideas that will prove important in subsequent chapters are also discussed here.

Chapter 3

Extensions to Guessing Numbers

This chapter builds on the more important aspects of the literature review. The first section gives a number of possible areas of research that were rejected for various reasons, be they ones of complication, time constraints or otherwise. Definitions required for the next chapter are also explained here. Finally, a theorem is stated linking the fractional guessing game to fractional routing, as an analogue to the theorem linking the ordinary guessing game to network coding.

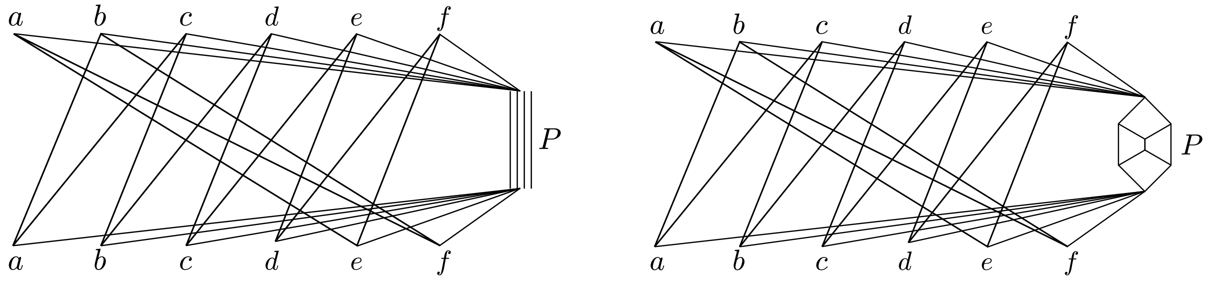
3.1 Some Ideas

As with most new technologies, little of the topic of guessing numbers is refined into a widely accepted theory. If, as in Section 2.1, network coding is to be viewed as Thomson's discovery of the electron and subsequent discoveries of other subatomic particles, then guessing numbers may be viewed as a move away from the Newton's corpuscular theory towards the concept of wave-particle duality; the combining of two separate existing notions to explain more complicated phenomena in the field. As such, there is much to investigate in order to find the most beneficial advancements. Some of these advancements, then, will lead to an academic impasse. The following are some propositions that were either of little value or proved to be too tricky or time-consuming.

3.1.1 Topological Equivalence of Networks

An interesting first concept was to investigate the *degree* of a given network — the degree of the polynomial required to give an optimal solution, and investigate the topological properties of networks with equal degree. However, as mentioned in Section 2.2.1, there are very few networks that have been shown to require non-linear codes for optimal solutions. Demonstrating topological equivalence would require a number of networks requiring quadratic (degree 2) solutions, yet more requiring cubic solutions, and so on. Finding single networks with these properties seems to be an entire topic in itself. Given the lack of networks known to have these properties, this was considered infeasible.

3.1.2 Bipartite Graphs



(a) Bipartite graph with an ordinary public channel

(b) Bipartite graph with a second graph in place of the public channel

Figure 3.1: Changing the ordinary public channel in (a) to the more complicated version in (b) does not help in finding a non-integer guessing number.

As mentioned in Section 2.6, it is possible to change a guessing number problem into a public channel problem, by converting the graph into its bipartite representation and substituting a suitable public channel for the guessing part of the game — if and only if the solution is linear [25]. The intention was to generalise or expand the public-channel game so that it could also be used in the non-linear case, perhaps by replacing the public channel with another network (Figure 3.1(b)). However, this led to the circular problem in which a network with a non-integer guessing number was required to find a network with a non-integer guessing number.

If a public channel is added to a network with a non-linear solution, then the guessing number k of that network may be increased, but the solution would then be an integer (i.e., a linear solution). It is not known, but would be interesting to investigate, whether the guessing number of the new graph would simply be $\lceil k \rceil$, the smallest integer not less than k.

3.1.3 Delays and Error

Many problems in network flow deal with delay-free (and error-free) networks, and indeed all problems concerning guessing numbers do not incorporate either delay or error-correcting codes. Real-world network flow problems may be translated into the theory of error-correcting codes [28], but no work has yet been produced involving the graph-theory approach of guessing numbers.

The implications of introducing these real-world events on a guessing game is not immediately clear; however one may intuitively guess that any delay will improve the guessing number of a particular graph, and error will reduce it.

3.2 Algebraic Calculations of Linear Guessing Numbers

It is shown in [25] that the *linear guessing number* has an algebraic definition:

Let $M = (m_{ij})_{i,j}$ be an $n \times n$ $(0,1)$-matrix, and let A be a finite field. Then define $\mathcal{C}_{\mathcal{A}}(\mathcal{M})$ to be the class of $n \times n$ matrices $M' = (m'_{ij})_{i,j}$ with entries in the field A for which $m_{ij} = 0 \Rightarrow m'_{ij} = 0$. Let I denote the $n \times n$ identity matrix and let $\mu(M) = n - \min_{M' \in \mathcal{C}_{\mathcal{A}}(\mathcal{M})} \operatorname{rank}(I + M')$.

Theorem 3.1: Let G be a directed graph with n nodes and adjacency matrix M_G. Let A be a finite field. Then the graph G has linear guessing number k (over the field A) if and only if $\mu(G_N) = k$.

Remark: The reason that the identity matrix I is added to the matrices M' is that the adjacency matrix M (and therefore the matrices M') have zeroes along the diagonal. In practice, any element $a \in A$ could be consistently used along the diagonal, but each a could be reduced to 1 by adding linear combinations of other rows or columns.

3.2.1 Generalising Riis' Equations

It is shown that it is possible to compute linear guessing numbers algebraically, by computing the ranks of a class of matrices. The linear guessing number has an algebraic definition $\mu(G_N)$ given above. It would be desirable if the algebraic definition could be extended to encompass non-linear guessing numbers. Network flow problems exist in the literature [12, 24] which only have non-linear solutions. Attempting to generalise the algebraic definition proves difficult, namely in finding the analogue of a 'rank' in non-linear functions. Whilst one may use Gröbner bases to solve simultaneous polynomial equations, the notion of rank does not extend to allow computation of a non-linear guessing number. It is worth pointing out that it is possible to rewrite Theorem 3.1 for non-linear guessing numbers, but this would merely be restating their definition. Instead of the rank, some notion of "whether this system is solvable" would be used.

3.3 Equivalence of Network Problems

The process of reduction alters a network coding problem into exactly one problem in graph theory. The reverse, however, gives a class of networks. It can be shown that these network coding problems are equivalent, simply by considering the number of split moves and unsplit moves required to produce them (see Section 2.3).

If network A and network B are both produced from graph G, then A and B have the same source transmission bandwidth. If A has global success rate p then it is possible (and easy) to compute the global success rate of B by first calculating the global success rate of G.

The motivation behind understanding this equivalence was to investigate if knowledge of it would allow investigation into the relationships between different network flow problems. However, it provided no extra information that was not already present in the literature; this equivalence would seem to be implicit, and as such is only made explicit here to clarify the link between network flow problems and their circuit representation.

3.4 Extending the Guessing Game to Include Routing Capacity

An extension of the guessing game is to include the concept of routing capacity on the network. A co-operative game involving only routing capacity can be imagined by players only being allowed to concatenate the visible bits, without any other operations. The addition of network coding allows the players to perform arbitrary bit operations.

Transforming network coding problems into problems on graphs works by identifying source nodes with their respective sink nodes. The equivalent transformation in the routing capacity case results in asymmetrical, directed graphs.

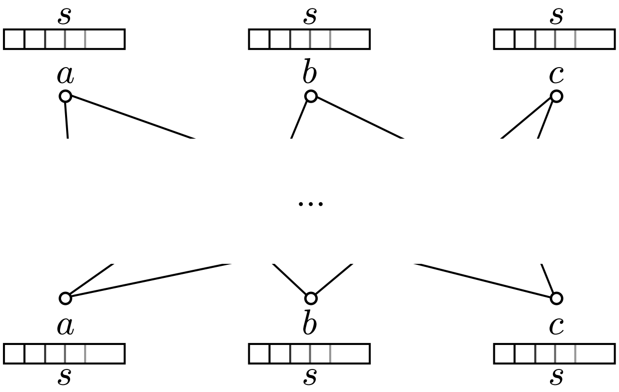


Figure 3.2: Asymmetrical graph transformed from a problem in routing capacity. The internal nodes are arbitrary.

The source nodes in Figure 3.2 are each transmitting messages of length log(s) bits. In general, the edges are of some capacity greater than 1. This means that the (arbitrary) internal nodes must be able to handle concatenations of larger inputs for the receiver nodes to compute their required messages. Mathematically, however, it is more interesting to investigate symmetrical, undirected graphs.

It has been shown that the solutions over an alphabet A (where $|A| = s$) of an information network flow problem $N \in \mathcal{C}_{mu}$ with n input nodes and n output nodes correspond directly with the optimal guessing strategies for a game played on G_N over the same alphabet A, and that each strategy ensures that the players succeed with probability $s^{n-|G_N|}$ ($|G_N|$ being the number of vertices in G_N) [25, 26].

The definition of the guessing game must be slightly modified to incorporate the different techniques used in routing problems as well as the size of the messages at the nodes. There is no longer a specific value $m_v \in A$, but a message consisting of bits from A.

The following is a formalisation of the new guessing game.

Definition 3.2: Let RoutingGuessingGame(G, s) or RGG(G, s) denote a co-operative game where each vertex $v \in V$ is assigned a message m_v. The message is actually a block $\mathbf{m_v} = m_1 m_2 m_3 \ldots m_s$ where each $m_i \in A$. This message is then transmitted to each vertex

w with $(v, w) \in E$. In other words, vertex w receives a set $M_w = \{m_v \in A^s : (v, w) \in E\}$. The goal is then to calculate the probability that the optimal protocol $\mathcal{P}$ (consisting of the functions $f_w : A^{s|M_w|} \to A^s$) succeeds.

This definition is a minor modification of the ordinary guessing game where the alphabet A is replaced with an alphabet A^s. The modification is motivated by the success of the specific guessing strategies highlighted in Chapter 4, and allows us to capture the idea of fractional routing into the guessing game. As we will see, the game has a particularly nice solution if the size of the alphabet is a perfect square, where $s = t^2, t \in \mathbb{N}, t > 1$. This corresponds to a version of the dice game where each player is assigned two dice values $x, y \in \{1, 2, \ldots, t\}$ (see Section 4.3.1).

The definition above describes the messages as bit strings, but for the case where s is a perfect square this is not really an appropriate definition since perfect squares are rarely a power of 2. It is possible to write the strings in a different base to achieve the desired effect, but in fact it makes more sense to write the problem as an entropy argument. This is motivated by proofs which appear in Chapter 4.

In this setting, the message m_v should be viewed as a collection of independent variables $\mathcal{M}_v = \{X_{v,1}, X_{v,2}, \ldots, X_{v,s}\}$, where each $X_{v,r}$ are statistically independent, so the joint entropy is equal to the sum of the individual entropies: $H(A, B) = H(A) + H(B)$, for any $A, B \in \mathcal{M}_v$.

As mentioned in Section 2.3, a multiple-unicast network N has a solution over an alphabet A if and only if the corresponding graph G_N has a guessing number not less than the number of input/output nodes in N. In exactly the same manner we can relate the new fractional guessing game defined above to fractional routing by the same theorem, the result of which is outlined below.

Theorem 3.3: A circuit information flow problem N with n input / output nodes has a fractional routing solution using words of length k if and only if the guessing number of the corresponding graph G_N has a fractional guessing number not less than n.

3.5 Summary

This chapter discusses various ideas for the progression of the thesis. Of the many ideas that were suggested, the more interesting research areas have been highlighted here, together with reasons for their infeasibility. Given time and other resources, it is entirely possible that such areas of work may produce fruitful work, hence their inclusion here.

In addition, a number of definitions are provided here as a precursor to Chapter 4. The main definition given of the fractional guessing game, which combines fractional routing and guessing numbers. This modified game will be used in the next chapter, which will discuss applications of the new game in solving particular problems in graph theory and network flow.

Chapter 4

Proofs

This chapter includes the main proofs related to the definition of the guessing game with fractional routing. Examples of the new game are provided to help explain the concept. These examples give an insight into where the new fractional guessing game can improve on the ordinary definition of the guessing game.

4.1 Examples

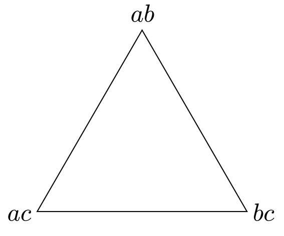


Figure 4.1: K_3 has $k = 3/2$ for block length 1.

It is interesting to study the strategies that yield the highest guessing numbers for players. In Figure 4.1 above, each player must resolve their own message (comprising $\log(s)$ bits) — however, only one bit may be received from each neighbour. When comparing bits, the probability that the two are the same is $s^{-1/2}$. If three bits are fixed, then since the other three are independent, the probability that each player correctly determines their own message is $s^{-3/2}$, giving a guessing number of $k = 3/2$. In other words, this strategy succeeds with higher probability than uncoordinated random guessing, but it is not as successful as the network coding strategy.

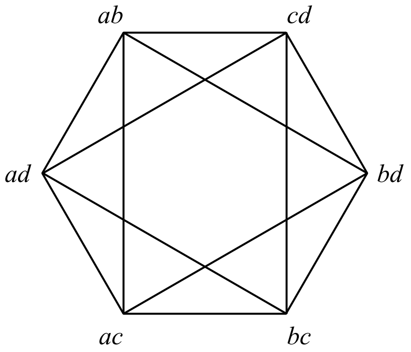


Figure 4.2: Grouping vertices on $Ci_6\,(1,2)$ yields the optimal solution.

The circulant graph $Ci_6\,(1,2)$ in Figure 4.2 contains 6 vertices. If 4 bits are fixed, then there are 8 free bits, giving a probability of success of $(s^{-1/2})^8 = s^{-4}$, and a guessing number of 2. However, if the graph is split into 3 node-disjoint pairs, then each pair independently guesses both messages correctly with a likelihood of $1/s$, giving a total probability of s^{-3}, or a fractional guessing number $k = 3$. Splitting the graph into two node-disjoint triangles gives the same result: $(s^{-3/2})^2 = s^{-3}$.

The ideas from fractional matching are used here; two node-disjoint odd cycles are formed in which each edge contributes 1/2 to the guessing number. Paring up the nodes allows each edge in the matching to contribute 1 to the guessing number.

This strategy can be continued by considering larger graphs. If a graph containing n vertices can be decomposed into p pairs and q triples, where each of the subgraphs is complete, then the probability that all players correctly determine their message is $s^{-p} \times s^{-3q/2} = s^{-n/2}$, since the number of vertices $n = 2p + 3q$. Any complete or suitably dense graph may be decomposed in this way. (Graphs containing any vertices of degree less than 2 are of no interest in this game.)

4.2 Guessing Numbers and Cycles

The graph in Figure 4.1 is an example where linear network coding is more successful than fractional routing. In general, this is not the case. K_3 can also be referred to as C_3, or a 3-cycle. An n-cycle is a graph containing n vertices and exactly 1 cycle; each vertex has degree 2. This section looks at different strategies on (undirected) n-cycles.

4.2.1 The $2n$-cycle

Here, we consider graphs containing an even number of nodes, each with degree 2, thus forming a cycle containing an even number of vertices. We arrive at the following theorem:

Theorem 4.1 ([27])**:** For each $n \in \mathbb{N}$, the $2n$-cycle has guessing number n.

Proof: First, consider the cycle as n pairs, each connected by an undirected edge; the other edges are simply ignored. Each pair contributes 1 to the guessing number; hence C_{2n} has a guessing number of at least n.

Now denote by $\{x_1, x_2, \ldots, x_{2n}\}$ the set of values assigned at random to the edges of the cycle. Fix any guessing protocol $\mathcal{P}$ and let $L_{\mathcal{P}}$ denote the set of $2n$-tuples $(x_1, x_2, \ldots, x_{2n}) \in A^{2n}$ for which all nodes correctly guess their assigned value. Now calculate the entropy of a string selected at random from $L_{\mathcal{P}}$, using the logarithm in base s, where $s = |A|$. To show the guessing number is at most 2 is equivalent to showing that $H(L_{\mathcal{P}}) \leq n$. Let $(x_1, x_2, \ldots, x_{2n})$ denote the random choice of a tuple from $L_{\mathcal{P}}$. Shannon's inequalities provide the following:

$$H(x_j) \leq 1 \quad \forall j \in \{1, 2, \ldots, 2n\}$$

$$H(x_{j-1}, x_j, x_{j+1}) - H(x_{j-1}, x_{j+1}) = 0 \quad \forall j \in \{1, 2, \ldots, 2n\}$$

where $j-1$ and $j+1$ are calculated modulo $2n$.

From this it can be seen that:

$$\begin{aligned} H(x_1, x_2, \ldots, x_{2n}) &= H(x_1, x_3, x_5, \ldots, x_{2n-1}) \\ &\leq \sum_{r=1}^{n} H(x_{2r-1}) \\ &\leq n. \end{aligned}$$

□

4.3 The 5-cycle

The 5-cycle, or pentagon, is an odd cycle, but unlike the 3-cycle, it is not complete. This loses the advantage of the previous optimal network coding strategy. This optimal strategy is the same as the strategy for the even cycle mentioned above, but with an odd cycle there will always be a vertex excluded from the pairings. Essentially, this means that the guessing number of C_n (utilising linear network coding) is given by n div 2, with the notable exception of C_3. Using fractional routing would seem to give a guessing number of $n/2$, giving this method an advantage for odd cycles.

This shows that there exists a non-integer guessing number using fractional routing. To see this, note that it is not possible for C_5 to have a guessing number of 3: if this was the case, then the network $N_5 \in \mathcal{C}_{mu}$ shown in Figure 4.3 would be solvable. However it is not solvable using only routing, so the guessing number is less than 3. This is a particular application of the link between guessing numbers and the solvability of networks. Since the guessing number has been shown to be at least 5/2, this proves the existence of a non-integer guessing number using fractional routing.

It is now clear why the new game must be imagined as two dice values, or some other example which splits the data at one node evenly between each of the two adjacent nodes. Imagine that it was not the case, and that each node transmitted a fraction p to the left, and a fraction $1-p$ to the right. To maintain the advantage described above, and for each

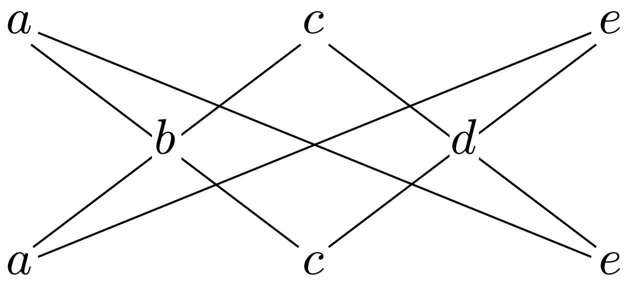


Figure 4.3: The pentagon in multiple-unicast form. As with bipartite graphs, there are two copies of the "external" nodes a, c and e.

node to receive the same amount of information, each must act consistently. Since the number of nodes is odd, one node will receive a fraction p from each neighbour, meaning $p = 1 - p$ and $p = 1/2$.

4.3.1 The Möbius-Band Guessing Strategy

As mentioned in Section 3.4, The guessing game played on the pentagon has a particularly elegant solution if the alphabet size is a square number, $s = t^2$. Here imagine that each player has two t-sided dice, numbered in the usual manner from 1 to t. the two values are given as an ordered pair $(x, y) \in \{1, 2, \ldots, t\}^2$. This provides an example of a network where the guessing number does indeed depend on s.

In the case of a pentagon, the players are assigned values (x_i, y_i), for $i \in \{1, 2, 3, 4, 5\}$. Consider the guessing strategy where each player assumes that the following identities hold:

$$\begin{aligned} x_1 &= x_2 \\ x_3 &= x_4 \\ x_5 &= y_1 \\ y_2 &= y_3 \\ y_4 &= y_5. \end{aligned} \tag{4.1}$$

This is analogous to each player assuming that one of their dice has the same value as one of their left neighbour's dice, and the other having the same value as one of their right neighbour's dice. This set of identities can be viewed geometrically as a Möbius band, by writing $x_1 = x_2, x_3 = x_4, x_5 =$ on one side of a strip of paper, $y_1, y_2 = y_3, y_4 = y_5$, on the other side, and fix the strip together with a half-twist [27], as in Figure 4.4. Note that other pairings could be used, for instance $x_i = y_{i+1}$ for $i \in \{1, 2, 3, 4, 5\}$, where i is calculated modulo 5, provided that there is a connecting edge between the two values.

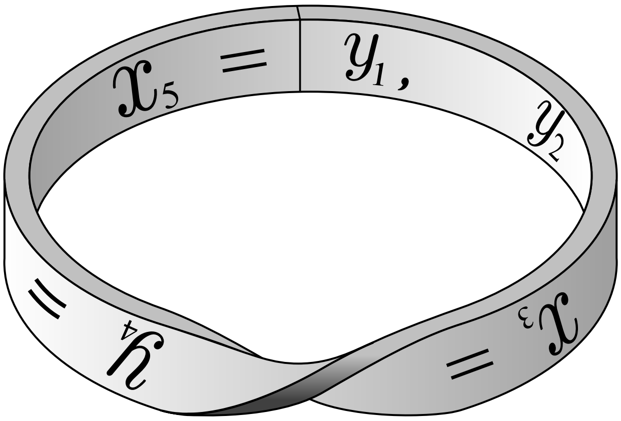


Figure 4.4: The Möbius-band guessing strategy is so called because it can be viewed as the set of solutions written on two sides of a strip of paper.

Using this guessing strategy on the pentagon ensures a guessing number in the half-open interval $(2, 2.5]$, with the upper bound of 2.5 only achievable if the underlying alphabet contains a square number of letters. In other words, if the size of the alphabet is not a perfect square, then the interval is fully open. Here we have an example where the guessing number depends on s. This idea is expressed in Lemma 4.2 and Lemma 4.3.

Lemma 4.2 ([27])**:** The Möbius-band guessing strategy ensures that $k(C_5, s) \geq 5/2$ where $s = t^2, t \in \mathbb{N}, t > 1$.

Proof: Using the Möbius-band guessing strategy, the players all guess correctly when the system of equations (4.1) hold, which happens with probability $t^{-5} = s^{-5/2}$. Since uncoordinated random guessing succeeds with probability s^{-5}, the guessing number is at least 5/2. □

As mentioned above, this lemma requires the alphabet to contain a square number of letters. In general, a similar method can be used, and indeed will be used below, though the upper bound of 5/2 is unachievable.

Lemma 4.3 ([27])**:** The guessing number of the pentagon is at most 5/2.

Proof: As with the proof for the $2n$-cycle, fix any guessing protocol $\mathcal{P}$ and let $L_{\mathcal{P}}$ denote the set of 5-tuples $(x_1, x_2, \ldots, x_5) \in A^5$ for which all nodes guess correctly. Select a string at random from $L_{\mathcal{P}}$, and calculate the entropies of the random variables in the string, using the logarithm in base s, where $s = |A|$. Again using Shannon's identities:

$$H(x_j) \leq 1 \quad \forall j \in \{1,2,3,4,5\}$$

$$H(x_{j-1}, x_j, x_{j+1}) - H(x_{j-1}, x_{j+1}) = 0 \quad \forall j \in \{1,2,3,4,5\}.$$

In addition to these identities there is a joint entropy relation, rather key to Shannon's theory of information, which states that

$$H(A,B,C) + H(C) \leq H(A,C) + H(B,C). \tag{4.2}$$

Now letting $(A,B,C) = (\{x_1\}, \{x_3\}, \{x_4, x_5\})$, the inequality becomes

$$\begin{aligned} H(x_1,x_2,x_3,x_4,x_5) + H(x_4,x_5) &= H(x_1,x_3,x_4,x_5) + H(x_4,x_5) \\ &\leq H(x_1,x_4,x_5) + H(x_3,x_4,x_5) \\ &= H(x_1,x_4) + H(x_3,x_5) \\ &\leq H(x_1) + H(x_3) + H(x_4) + H(x_5). \end{aligned}$$

Thus,

$$H(x_1,x_2,x_3,x_4,x_5) \leq H(x_1) + H(x_3) + H(x_4) + H(x_5) - H(x_4,x_5). \tag{4.3}$$

Next, using conditional entropy,

$$\begin{aligned} H(x_1,x_2,x_3,x_4,x_5) - H(x_2,x_5) &\leq H(x_3,x_4|x_2,x_5) \\ &= H(x_4|x_2,x_5) \\ &\leq H(x_4|x_5) \\ &= H(x_4,x_5) - H(x_5). \end{aligned}$$

This gives

$$\begin{aligned} H(x_1,x_2,x_3,x_4,x_5) &\leq H(x_2,x_5) + H(x_4,x_5) - H(x_5) \\ &\leq H(x_4,x_5) + H(x_2). \end{aligned} \qquad (4.4)$$

Adding (4.3) and (4.4) we get

$$2H(x_1,x_2,x_3,x_4,x_5) \leq H(x_1) + H(x_2) + H(x_3) + H(x_4) + H(x_5) \leq 5.$$

□

The same argument may be used to prove a similar theorem about odd cycles in general:

Theorem 4.4 ([11])**:** The optimal guessing number of an undirected cycle on $2k+1$ vertices is $k + \frac{1}{2}$.

Proof:

As before, setting $(A,B,C) = (\{x_1\}, \{x_3,\ldots,x_{2k-1}\}, \{x_{2k},x_{2k+1}\})$ in (4.2), we have

$$\begin{aligned} H(x_1,x_2,x_3,\ldots,x_{2k},x_{2k+1}) + H(x_{2k},x_{2k+1}) &= H(x_1,x_3,\ldots,x_{2k},x_{2k+1}) + H(x_{2k},x_{2k+1}) \\ &\leq H(x_1,x_{2k},x_{2k+1}) + H(x_3,\ldots,x_{2k+1}) \\ &\leq H(x_1,x_{2k}) + H(x_3,x_5,\ldots,x_{2k-1},x_{2k+1}) \\ &\leq H(x_{2k}) + \sum_{r=0}^{k} H(x_{2r+1}). \end{aligned}$$

This gives

$$H(x_1,\dots,x_{2k+1}) \le H(x_{2k}) - H(x_{2k},x_{2k+1}) + \sum_{r=0}^{k} H(x_{2r+1}). \tag{4.5}$$

Now setting $(A,B,C) = (\{x_2,\dots,x_{2k-2}\},\{x_{2k}\},\{x_{2k+1}\})$ in (4.2), we have

$$\begin{aligned} H(x_1,\dots,x_{2k+1}) + H(x_{2k+1}) &= H(x_2,\dots,x_{2k-2},x_{2k},x_{2k+1}) + H(x_{2k+1}) \\ &\le H(x_2,\dots,x_{2k-2},x_{2k+1}) + H(x_{2k},x_{2k+1}) \\ &\le H(x_2,x_4,\dots,x_{2k-4},x_{2k-2},x_{2k+1}) + H(x_{2k},x_{2k+1}) \\ &\le H(x_{2k},x_{2k+1}) + H(x_{2k+1}) + \sum_{r=1}^{k-1} H(x_{2r}). \end{aligned}$$

Therefore,

$$H(x_1,\dots,x_{2k+1}) \le H(x_{2k},x_{2k+1}) + \sum_{r=1}^{k-1} H(x_{2r}). \tag{4.6}$$

Finally, adding (4.5) and (4.6) we get

$$2H(x_1,\dots,x_{2k+1}) \le \sum_{r=1}^{2k+1} H(x_r) \le 2k+1.$$

□

4.3.2 General Case

As mentioned above, the guessing game has a nice solution when $s = |A|$ is a perfect square, and the guessing number of a pentagon in this case is $5/2$. In general, where the size of the alphabet is not necessarily a square, the guessing number is in the interval $(2, 2.5)$ as before. The cardinality of $L_{\mathcal{P}}$ is (in general) an integer.

Observation: The guessing number of any directed graph on n nodes over an alphabet of size s is of the form $k = \log_s(j)$ where $j \in \mathbb{N}$.

From this, and using the transcendence of numbers of the form $\log_s(j)$, the following is evident:

Corollary 4.5: Assume s is not a power. Then the guessing number of a directed graph over an alphabet of size s is either an integer or an irrational number. If, on the other hand, $s = t^k$ where t is not a power, each directed graph has either a guessing number of the form r/k where $r \in \mathbb{N}$, or is irrational.

4.3.3 Example

To illustrate the above observation, the simplest example is used here: the case of the pentagon where the underlying alphabet has size 2; $A = \{0, 1\}$. The guessing number is of the form $\log_2(j)$. Since the guessing number is at least 2 and at most $5/2$, there are only two possible values, where $j = 4$ and $j = 5$, giving guessing numbers of 2 and $\log_2(5) \approx 2.32$. There is actually a strategy that ensures a guessing number of $\log_2(5)$: All players guess with the assumption that there are never two consecutive 1s or three consecutive 0s. There are exactly five 5-tuples that satisfy these assumptions, namely $(0, 0, 1, 0, 1)$, $(0, 1, 0, 0, 1)$, $(0, 1, 0, 1, 0)$, $(1, 0, 0, 1, 0)$ and $(1, 0, 1, 0, 0)$. This strategy also uniquely determines the values available to the players from their neighbours. The guessing strategy succeeds with probability 5/32, or 5 times more successful than random guessing. Therefore, the guessing number of the pentagon for $s = 2$ is $\log_2(5)$.

4.3.4 Three-Layers Argument

The three-layers argument is useful as a pictorial description of the message pairing process used in the Möbius-band guessing strategy. In the case of even cycles, it is not possible to improve on the node-pairing method. To show that it is not possible to improve on a guessing number of n for a $2n$-cycle, consider a pattern where every evenly numbered node is fixed, so the odd-numbered nodes can be uniquely determined.

As in Section 4.3.1 above, each node in a pentagon is assigned two values (x_i, y_i), for $i \in \{1,2,3,4,5\}$, and $x_i = y_{i+1}$ for $i \in \{1,2,3,4,5\}$, where i is calculated modulo 5. Now, split all nodes x_i as with a bipartite graph, leaving nodes y_i as "internal" nodes. The initial and resulting graphs are shown in Figure 4.5.

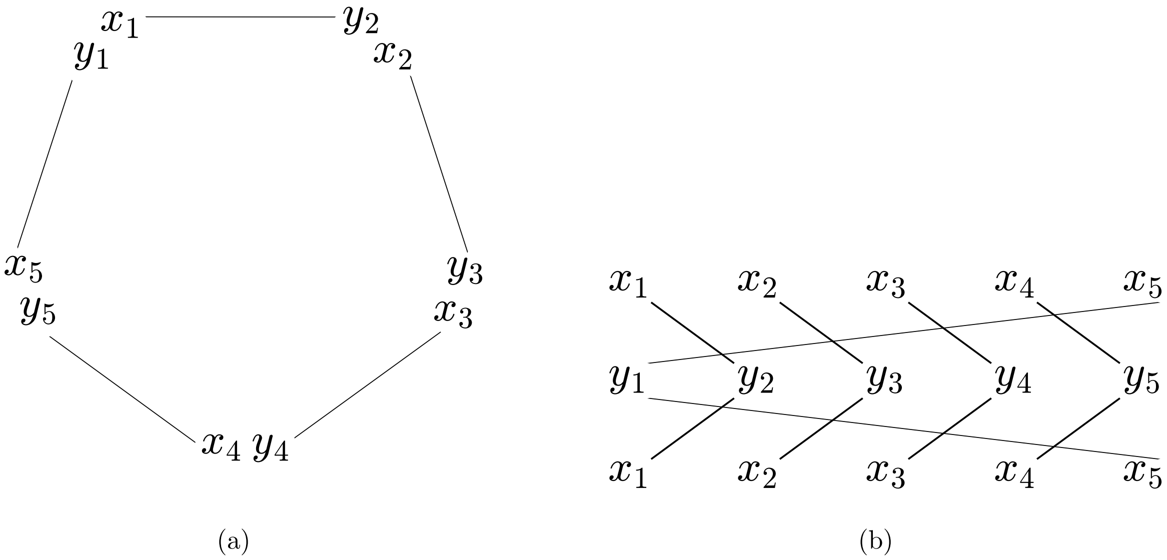

x_1
y_1
y_2
x_2
x_5
y_5
y_3
x_3
x_4 y_4
(a)
x_1 x_2 x_3 x_4 x_5
y_1 y_2 y_3 y_4 y_5
x_1 x_2 x_3 x_4 x_5
(b)

Figure 4.5: Each node of the pentagon (a) is assigned two values. Nodes x_i are then split as in a bipartite graph to give the three-layered graph (b). As with bipartite graphs, edges are directed downwards.

This argument seems to show quite well what happens when considering that each node has two messages to resolve.

4.4 Summary

This chapter contains the main theorems resulting from the new guessing game. The proofs of these theorems are also included, and centre mainly around the use of Shannon's inequalities to determine the guessing numbers of particular graphs, specifically undirected cycles. Two key observations are made about the properties of guessing numbers under certain conditions in the new game; finally, the observations are illustrated with a simple example.

Chapter 5

Evaluation

This chapter provides an evaluation of the new guessing game based on network routing capacity. It looks at the objectives mentioned in the introduction of the thesis, and attempts to critically evaluate how successful the thesis was in achieving those objectives. It also suggests possible further work, as a number of ideas for further research have become apparent during the course of developing the methods used here.

5.1 Evaluation of the Criteria for Success

In Section 1.5 the following criteria for success were outlined:

- study and description of existing methods within network coding and network routing capacity;
- development of a new method for solving network flow problems based on existing techniques;
- abstraction and formalisation of a modified guessing game based on the new method;

- study and evaluation of cases in which the new game is more successful than the existing game.

These criteria are evaluated below to determine the overall success of the thesis.

5.1.1 Study and Description of Existing Methods

Chapter 2 contains the literature review, which investigates existing work and methods currently in use. It follows on from the historic introduction provided in Chapter 1, which provides a basic chronology of the events leading up to the beginning of this thesis. The chapter gives an in-depth explanation of the traditional routing methods in use today, and follows on to give simple examples where network coding has been shown to be superior. Guessing numbers and routing capacity are also defined here, along with a number of definitions which are used throughout the thesis. A number of other relevant topics which will be used in later sections are given detailed explanations, together with simple examples to further illustrate their meaning.

Each major technique for dealing with network flow problems has been either superseded or generalised to tackle a larger set of problems, and though specific approaches and notations may have changed and improved, the overall goal remains the same: to model in theoretical terms particular problems which occur in the real world, to understand them and to investigate how they may be solved. Work that brings together two or more techniques within a field — like combining information theory with network routing [1], or indeed brings together work from two or more fields — like network coding and graph theory [25, 26], may be seen as an advancement towards some unified theory for understanding network flow. The literature survey has been satisfactory in determining the most useful features of existing work to be combined into the new game.

5.1.2 Development of a New Method

To this end, this thesis has brought together two recent techniques in network theory, namely Riis' largely uncharted field of guessing numbers and Cannons *et al.*'s study into routing capacity. Chapter 3 arrives at the conclusion that these are the best areas to consider, from discussing a number of possible approaches to arriving at a suitable amalgamation to be formalised and studied.

Section 3.1 lists some of the initial areas of research before moving on to study routing capacity. Throughout the research, a great number of ideas were discussed, before being disregarded due to their difficulty. It proved to be rather frustrating that so many questions were raised that simply could not be answered sufficiently, but it was necessary to discard these ideas in favour of more fruitful areas of research.

The main result of this development was to arrive at the combination of techniques mentioned above, extending and modifying the guessing game played on graph representations of network flow problems, to include the routing capacity of those networks. The game introduced is not *new*, however, and is merely an extension of the game existing in the literature.

What the game does do is to include the concept of fractional routing for multiple-unicast networks into the existing guessing game: the original guessing game was concerned with using a guessing strategy over an alphabet A. The notion of fractional routing uses block coding where strings of letters $\{a_1, a_2, \ldots, a_k\}$ are used, which is mathematically equivalent to using an underlying alphabet of A^k.

5.1.3 Abstraction & Formalisation of a Modified Guessing Game

Once the extension was developed, the ideas proposed were formalised to produce the co-operative guessing game RoutingGuessingGame(G, s) played on a graph G where the messages are selected from an alphabet of size s. This essentially incorporates the notion of fractional routing into the existing game. Particular problems were identified and examples were given in Chapter 4. Particularly, the main result has been to show the effectiveness of the new game with odd cycles C_{2k+1}, $k \in \mathbb{N}$.

To further illustrate the place of the new game in relation to the current literature, a theorem is stated linking the fractional guessing game to fractional routing in the same way that the ordinary guessing game was related to network coding. The connection between the fractional guessing number and the fractional routing solution of the related circuit information flow problem helps to formalise the guessing game.

A general objective mentioned in the introduction was to go some way to reducing the number of open questions in the field of network coding and guessing numbers. When guessing numbers were first defined, they helped to answer some questions, but also introduced questions into the field. One such question was whether the guessing number of an undirected graph is always an integer [25]. In Section 4.3.1, it is proven that the undirected pentagon C_5 has a non-integer guessing number. An unexpected result was that the guessing number also depended on the alphabet size s. This work is then generalised to odd cycles of the form C_{2k+1}, for $k \in \mathbb{N}$. It is these odd cycles where the newly introduced method has an advantage over the existing use of scalar linear network coding.

Although the thesis does indeed prove and give examples to satisfactorily answer this question, other questions arise, mostly regarding using this method in conjunction with existing techniques. For instance, it is not known whether a network exists for which both linear network coding and fractional routing are insufficient, yet some other method succeeds.

5.1.4 Evaluation of the New Method against the Old Method

The main objective mentioned in Section 1.5 was to give a new method of solving network flow problems. The result is that using a fractional routing approach, a class of networks has been found where this new approach improves on the success of linear network coding. Specifically, as mentioned in Section 4.3, there exists a simple graph (the 5-cycle) where a guessing strategy using fractional routing (and two dice) fares better than any (fractional) guessing strategy using one die. Whilst this method does improve on both the existing guessing strategy, it does so in a very small class of networks. The idea of using a fractional guessing strategy would need to be continued to see if it succeeds on other classes of networks where other guessing strategies do not. It would be interesting to see if there is any case where it could out-perform a network coding approach using non-linear functions.

The correspondence between the guessing game defined here and the game previously defined in [25] is not precise: messages in routing problems are generally larger than those in network coding. Consequently, it is difficult to provide a comparison between the games — sometimes a coding strategy fares better, other times a fractional routing strategy proves more successful. However, this does mean that we have extended the existing approach for studying this class of information flow problems.

Whilst the idea of combining the techniques of guessing games and fractional routing may make sense in terms of data transmission, it does not do so in terms of the games outlined in the previous chapters. However, it may prove to have some validity if a restriction is placed on the network coding strategy, such as only allowing linear functions to be used.

5.2 Extensions

One interesting extension has already been mentioned above, namely the combination of linear network coding with routing capacity to produce more advanced strategies for the guessing games. However, study in this field seems to be moving toward the use of non-linear coding functions to provide the best results in the case of larger networks. By comparison, the work in this thesis is on relatively simple networks, and may not translate to these more complex situations.

It may also be worth considering further study into the public-channel version of the guessing game, and extending that to include routing capacity in the same way that the ordinary game was altered. It is likely that this would back up the findings from the odd cycle: for each message to be uniquely determined on an odd cycle, a smaller public channel would be required for a fractional routing approach than for the existing approach.

Different representations of the same problem can yield different solutions, in some cases more successful than previous, known results. The examples given in Chapter 4 are cases where the fractional routing provides a solution which succeeds with higher probability than scalar linear solutions. Further extending these games would hopefully lead to a collection of approaches for tackling a great number of problems in network flow.

Chapter 6

Conclusion

This thesis is concerned with network coding, an emerging field of information theory which allows us to obtain the optimum flow of information through a network. It has a number of real-world applications which could reduce network delays, improve reliability or otherwise increase throughput. Such a great number of technological advancements rely on the transmission of data across such a network — radio and television, telecommunications and the Internet to name but a few — that understanding any possible improvements that can be made to this transmission is of great interest to the professionals who monitor them. With increasing focus placed on wireless communication instead of wireline systems, the detection and correction of data corruption becomes ever-more important, as does the compression and security of the transmission. Network coding has a number of diverse applications to which it can be applied.

In attempting to understand problems in network flow, a mathematical representation of the problem is used. In Riis' guessing game, the coding functions used in the network are represented by the guessing strategies adopted by the players. There is a one-to-one correlation between the solutions of an information network flow problem and the optimal guessing strategies adopted by the players on the corresponding graph.

The main intention of the thesis was to combine the techniques of fractional routing with the concept of guessing numbers to further the study of guessing numbers and their use with network information flow problems. In attempting this, it was also hoped to be possible to answer some of the open questions relating to network coding. The evaluation shows that these objectives have been met, but that the method only succeeds in a limited number of cases. However, further work could go some way into increasing the number of problems where fractional routing provides the optimum solution.

The new guessing game is an intersection of two major areas of mathematics, and shows what can be achieved with a relatively simple example. Regarding the specific criteria for success, it is fair to say that grandiose objectives such as a "new method" were perhaps overly ambitious, and to suggest that this objective had been entirely met would be over-stating the achievements: the work produced is only a minor modification of work introduced in the literature. The formulation of completely new methods is left to the experts; the work in this thesis simply aims to build on those methods already suggested. Beyond this, however, it is fair to conclude that the objectives were both well-proposed and satisfactorily met.

A larger range of different examples would be required to show the general usefulness of the new guessing game to the network coding. The field of network coding is still rather new, and more proof is needed of its viability before it is widely adopted as an alternative to traditional routing.

6.1 Open Questions

- The routing capacity of a network is always rational. Is this true for a guessing number?
- If not, what restrictions must be placed on the guessing number to force rationality?
- If the guessing number k of a given graph is independent of the alphabet size s, is k necessarily an integer?

References

[1] Rudolf Ahlswede, Ning Cai, Shuo-Yen Robert Li, and Raymond Yeung, *Network Information Flow*, IEEE Transactions on Information Theory **46** (2000), no. 4, 1204–1216.

[2] Kenneth Appel and Wolfgang Haken, *Solution of the Four Color Map Problem*, Scientific American **237** (1977), no. 4, 108–121.

[3] Ziv Bar-Yossef, Yitzhak Birk, Jayram Thathachar, and Tomer Kol, *Index Coding with Side Information*, Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science, October 2006, pp. 197–206.

[4] Jacob Bekenstein, *Black Holes and Entropy*, Physical Review D **7** (1973), no. 8, 2333–2346.

[5] Norman Biggs, Keith Lloyd, and Robin Wilson, *Graph Theory, 1736–1936*, Oxford University Press, 1986.

[6] Jean-Marie Bourjolly and William Pulleyblank, *König–Egerváry Graphs, 2-Bicritical Graphs and Fractional Matchings*, Discrete Applied Mathematics **24** (1989), 63–82.

[7] Ning Cai and Raymond Yeung, *Secure Network Coding*, 2002, p. 1.

[8] Peter Cameron, Søren Riis, and Taoyang Wu, *On the Guessing Number of Shift Graphs*, submitted.

[9] Jillian Cannons, Randall Dougherty, Chris Freiling, and Kenneth Zeger, *Network Routing Capacity*, IEEE Transactions on Information Theory **52** (2006), 777–788.

[10] Dah Ming Chiu, Raymond Yeung, Jiaqing Huang, and Bin Fang, *Can Network Coding Help in P2P Networks?*, 4th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks **52** (2006), 1–5.

[11] Stefan Dantchev, 2007, personal communication.

[12] Randall Dougherty, Chris Freiling, and Kenneth Zeger, *Insufficiency of Linear Coding in Network Information Flow*, IEEE Transactions on Information Theory **51** (2005), no. 8, 2745–2759.

[13] ———, *Unachievability of Network Coding Capacity*, IEEE Transactions on Information Theory **52** (2006), no. 6, 2365–2372.

[14] Jack Edmonds, *Paths, Trees and Flowers*, Canadian Journal of Mathematics **17** (1965), 449–467.

[15] James Gleick, *Chaos*, Cardinal Books, 1990.

[16] Tracey Ho, David Karger, and Muriel Médard, *Network Coding from a Network Flow Perspective*, Proceedings of the ISIT 2003, July 2003.

[17] Ralf Koetter and Murial Médard, *An Algebraic Approach to Network Coding*, IEEE/ACM Transactions on Networking **11** (2003), no. 5, 782–795.

[18] Peter Larsson and Niklas Johansson, *Multi-User ARQ*, Vehicular Technology Conference 2006 **4** (2006), 2052–2057.

[19] April Rasala Lehman and Eric Lehman, *Complexity Classification of Network Information Flow Problems*, Proceedings of the 41st Annual Allerton Conference for Communication Control and Computing, vol. 41, October 2003.

[20] Shuo-Yen Robert Li, Raymond Yeung, and Ning Cai, *Linear Network Coding*, IEEE Transactions on Information Theory **49** (2003), no. 2, 371–381.

[21] Yan Liu and Guizhen Liu, *The Fractional Matching Numbers of Graphs*, Networks **40** (2002), no. 4, 228–231.

[22] Muriel Médard, Michelle Effros, Tracey Ho, and David Karger, *On Coding for Non-Multicast Networks*, Proceedings of the 41st Annual Allerton Conference for Communication Control and Computing, vol. 41, October 2003, pp. 21–29.

[23] William Pulleyblank, *Optimization*, Handbooks in Operations Research and Management Science (George Nemhauser, Alexander Rinnooy Kan, and Michael Todd, eds.), vol. 1, North-Holland, 1989, pp. 371–446.

[24] Søren Riis, *Linear versus Non-Linear Boolean Functions in Network Flow*, Proceedings of CISS 2004, vol. 38, March 2004.

[25] ———, *Utilising Public Information in Network Coding*, submitted, June 2005.

[26] ———, *Information Flows, Graphs and Their Guessing Numbers*, 4th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, April 2006.

[27] ———, 2007, personal communication.

[28] Søren Riis and Rudolf Ahlswede, *Problems in Network Coding and Error Correcting Codes*, April 2005.

[29] Yalin Sagduyu and Anthony Ephremides, *Joint Scheduling and Wireless Network Coding*, April 2005.

[30] Bruce Schneier, *Applied Cryptography*, second ed., John Wiley & Sons, 1996.

[31] Claude Shannon, *A Mathematical Theory of Communication*, The Bell System Technical Journal (1948).

[32] Leslie Valiant, *On Non-Linear Bounds in Computational Complexity*, Proceedings of 7th annual ACM symposium on Theory of Computing, May 1975, pp. 45–53.

[33] Hermann Weyl, *The Elementary Theory of Convex Polyhedra*, Contributions to the Theory of Games, Volume I (Harold Kuhn and Albert Tucker, eds.), vol. 7, Princeton University Press, 1950, pp. 290–306.

[34] Robin Wilson, *Four Colours Suffice*, third ed., The Penguin Group, 2003.

[35] Raymond Yeung and Zhen Zhang, *Distributed Source Coding for Satellite Communications*, IEEE Transactions on Information Theory **45** (1999), no. 4, 1111–1120.