Utilising public information in Network Coding

Søren Riis
Queen Mary, University of London
(Technical report June 2005)

***Abstract*— We show that an information network flow problem N in which n messages have to be sent to n destination nodes has a solution (that might utilise Network Coding) if and only if the directed graph G_N (that appears by identifying each output node with its corresponding input node) has *guessing number* $\geq n$. The *guessing number* of a (directed) graph G is a new concept defined in terms of a simple cooperative game. We generalise this result so it applies to general information flow networks.**

We notice that the theoretical advantage of Network Coding is as high as one could have possibly hoped for: for each $n \in N$ we define a network flow problem N_n with n input nodes and n output nodes for which the optimal through-put using Network Coding is n times as large as what can be obtained by vector routing or any other technique that does not allow interference (between messages) . In the paper we obtain a characterisation of the set of solutions for each flow problem N_n. We use this to prove a number of theorems for information networks.

I. Network Coding

A. The wave approach to information network flow

In recent years a new area called Network Coding has evolved. Like many fundamental concepts, Network Coding is based on a simple mathematical model of network flow and communication first explicitly stated in its simplicity in [3]. Recently, ideas related to Network Coding have been proposed in a number of distinct areas of Computer Science and engineering (e.g. broadcasting in wireless networks [26], [25], [24], data security [4], distributed network storage [7], [1] and wireless sensor networks [16]). Network Coding has also a broad interface with various Mathematical disciplines (error correcting codes [19], [5], [11], circuit complexity [18], information theory [12], algebra [14], [13] and graph theory).

The basic idea underlying Network Coding has been explained in numerous papers e.g. [14], [3], [18], [8].The idea can be illustrated by considering the "butterfly" network in figure 1a.

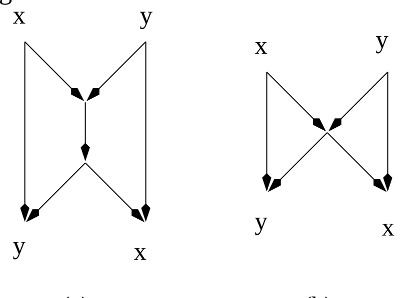

figure 1

The task is to send the message x from the upper left corner to the lower right corner, and to send the message y from the upper right corner to the lower left corner. The messages $x, y \in A$ are selected from some finite alphabet A. Assume that each information channel can carry at most one message at a time. If the messages x and y are sent simultaneously there is a bottleneck in the middle information channel. On the other hand if we, for example, send $x \oplus y \in A$ through the middle channel, the messages x and y can be recovered at 'output' nodes at the bottom of the network.

The network in figure 1a can be represented as the network in figure 1b. In this representation (which we will call the 'circuit representation') each node in the network computes a function $f : A \times A \rightarrow A$ of its inputs, and sends the function value along *each* outgoing edge. Historically, it is interesting to note that in this slightly different (but mathematically equivalent) form, the idea behind Network Coding (i.e. the power of using non-trivial boolean functions rather than "pushing bit") was already acknowledged in the 70s (though never emphasised or highlighted) in research papers in Circuit Complexity (see e.g. [23], [21], [17], [22], [2]). It is also worth mentioning that in Complexity Theory many lower bounds are proven under the assumption that the algorithm is *conservative*, or can be treated as such. Conservative means that the input elements of the algorithm are atomic unchangeable elements that can be compared or copied but can not be used to synthesise new elements during the course of the algorithm. From a perspective of Circuit Complexity, Network Coding is an interesting theory of information flows since it correspond to unrestricted models of computation.

Information flow in networks falls naturally within a number of distinct paradigms. Information flow can, for example, be treated in a fashion similar to traffic of cars in a road system. In this view each message is treated as **a packet** (e.g. a car) with a certain destination. Messages (cars!) cannot be copied, or divided. This way of treating messages is almost universally adopted in today's information networks (e.g. wireless communication, communication on the web, communication within processors or communication between processors and external devises). Another, less used possibility, is to treat messages in some sense as **a liquid** that can be divided and sent along different routes before it reaches its destination. This approach (like, for example, in vector routing [6]) allows messages to be spread out and distributed over large parts of the network. Another and more radical approach is to treat messages as **"waves"**. Recall that the signals carrying the messages are digital (discrete) and thus certainly do not behave like waves. It is, however, possible to transmit and handle the digital (discrete) signals in a fashion where the messages (not the bits carrying the messages) behave like waves subject to interference and super position. More specifically, assume that A is a (finite) alphabet of distinct (wave) signals that can be sent through a channel. The superposition of (wave) signals

$w_1, w_2 \in A$ creates a new (wave) signal $w = w_1 \oplus w_2 \in A$. Thus mathematically, in the wave picture, the set A of wave signals forms a (finite) commutative group with the neutral element $0 \in A$ representing the zero-signal.

The network in figure 2 illustrates the point that in specific Network topologies there can be quite a large advantage of treating messages as waves. The task of the network is to send messages x, y and z from the source (input) nodes i_1, i_2 and i_3 to the three output nodes o_1, o_2 and o_3. The receiver (output) node o_1 requires x, node o_2 requires y and node o_3 requires z. We assume that channels are one-way and that the messages are only sent downwards in the figure. All crossings in the figure are 'bridges' and it is, for example, only possible to move from i_1 to o_1 by moving through channel p.

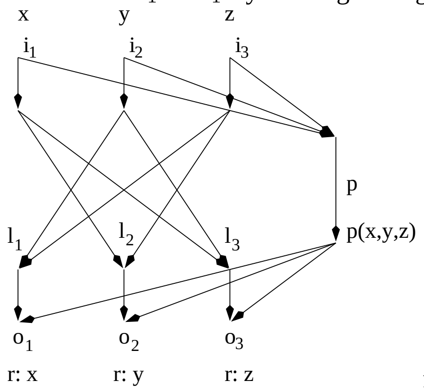

figure 2

If messages are treated as packets (cars) like in traditional routing, or if messages are treated as a liquid, there is no point in sending information through l_1, l_2 or l_3. All messages x, y and z must pass through the channel labelled with p (for 'public'). This clearly creates a bottleneck in channel p if we assume that only one message can pass at a time.

If, however, messages are treated as waves we can send $p(x, y, z) := x \oplus y \oplus z$, the superposition of the messages x, y and z, through channel p. And we can send superpositions $l_1 := -(y \oplus z)$, $l_2 := -(x \oplus z)$ and $l_3 := -(x \oplus y)$ through the nodes with these labels. Node o_1 can take the superposition of l_1 and $p(x, y, z)$ and then reconstruct the message x $= -(y \oplus z) \oplus (x \oplus y \oplus z)$. Similarly, node o_2 (or o_3) can take the superposition of l_2 (or l_3) and $p(x, y, z)$ and then reconstruct the message $y = -(x \oplus z) \oplus (x \oplus y \oplus z)$ (or $z = -(x \oplus y) \oplus (x \oplus y \oplus z)$. This shows that the wave approach allows us to eliminate the bottleneck in channel p in figure 2. Notice also that the wave approach increases the overall network performance (of the network in figure 1) by a factor 3. [1]

In general the advantage of the wave approach (compared to any approach that does not allow interference) can be as large as one could have possibly hoped for. We will later notice that there exist information flow networks (with n source nodes and n receiver nodes) for which the optimal throughput is n times larger using the wave approach. Actually, there are even networks where the success rate for each active channel using the wave approach is close (as close as we wish) to n times the success rate for each active channel in a routing solution. The wave approach usually requires more information channels to be involved than traditional routing (or other methods that do not allow interference). Yet, by allowing interference, the total network performance divided by number of active information channels can for some network topologies be close to n times higher than any approach that is unable to utilise interference.

Network Coding allows messages to be sent within the wave paradigm. In fact, super-positioning of signals (described above) represents an important type of Network Coding we will refer to as *Linear Network Coding* (see also [15]). Although Linear Network Coding represents a very important subclass of Network Coding, in general Network Coding involves methods that go beyond linear Network Coding. Certain network problems have no linear solutions, but require the application of non-linear boolean functions [18], [8]. Non-Linear Network Coding has no obvious physical analogue. Rather general Network Coding represents a paradigm of information flow based on a mathematical model where 'everything goes'. In this model there are no apriory restrictions on how information is treated. Thus in Network Coding, packets might be copied, opened and mixed. Sets of packets might be subject to highly complex non-linear boolean transformations.

II. Coherence: Utilising apparently useless information

A. A Guessing game with dice

While I was researching various flow problems related to Circuit Complexity it became clear that a key problem is to characterise and formalise what pieces of information are "useful" and what pieces of information are genuinely "useless" . It became clear that this distinction can be very deceptive. A piece of Information that is useless in one context, can sometime be very valuable in a slightly different context [18].

To illustrate the problem Mikkel Thorup developed the following game that illustrates a nice mathematical idea we developed during a meeting in 1997 [20]: Assume that n players each has a fair s-sided dice (each dice has its sides labelled as $1, 2, \dots s$). Imagine that each player (simultaneously) throws their dice in such a manner that no player knows the value of their own dice.

1) What is the probability that each of the n players is able to guess correctly the value of their own dice?
2) Assume that each player knows the values of all other dice, but has no information about the value of their own dice. What is the probability that each of the n players correctly guesses the value of their own dice? (**Hint:** *The probability is NOT* $(\frac{1}{s})^n$*- The players can do much better than uncoordinated guessing!!*)
3) Assume the i^{th} player receives a value $v_i = v_i(x_1, x_2, \dots x_{i-1}, x_{i+1}, \dots, x_n) \in \{1, 2, \dots s\}$ that is allowed to depend on all dice values except the i'th player's own dice. What is the probability that each of

[1]Notice that this increase of a factor 3 comes at a certain expense. In the routing approach only 7 channels are active (namely, $(i_1, p), (i_2, p), (i_3, p), (p, o_1), (p, o_2), (p, o_3)$ and channel p), while in the Network Coding solution all 19 channels are active. The success rate $\frac{3}{19}$ for each active channel is higher in the Network Coding solution than in the ordinary solution $\frac{1}{7}$

the n players correctly manages to guess the value of their own dice?

In question 1 the probability that each player is right is $\frac{1}{s}$ and thus with probability $(\frac{1}{s})^n$ all n players successfully manage to guess correctly their own dice' value simultaneously. Maybe somewhat surprisingly in question 2, the answer depends on the 'protocol' adopted by the players! An optimal protocol appears, for example, if the players agree in advance to assume that the sum of all n dice' values is divisible by s. *This protocol ensures that all players simultaneously 'guess' the value of their own dice with probability* $\frac{1}{s}$.

Question 3, can be answered using a minor modification of the protocol just discussed. Let v_i be defined as the sum $x_1 \oplus x_2 \oplus \ldots \oplus x_{i-1} \oplus x_{i+1} \oplus \ldots \oplus x_n$ modulo s. Each player then 'guesses' that $x_i = -v_i$ modulo s. Again, the probability that all n players simultaneously guess the correct value of their own dice is $\frac{1}{s}$.

B. Playing the guessing game on a graph

We will now define a generalisation of the dice guessing game that is (surprisingly?) directly related to a certain type (the so called multiple-unicast type) of information flow problems. Recall a graph $G = (V, E)$ consists of a set V (the vertex set) and a set $E \subseteq V \times V$ (the edge set). Our definition of a graph is sometimes referred to as directed graphs. In this paper all graphs are directed.

Definition

Assume that we are given a (directed) graph $G = (V, E)$ on a vertex set $V = \{1, 2, \ldots, n\}$ representing n players. We let GuessingGame(G, s) denote the cooperative game defined as follows: Each player $v \in \{1, 2, \ldots, n\}$ is randomly assigned a die value $\in \{1, 2, \ldots, s\}$. Each player sends the value of their die $\in \{1, 2, \ldots, s\}$ to each player $w \in \{1, 2, \ldots, n\}$ with $(v, w) \in E$. In other words, each node w receives dice' values from a set $A_w := \{v \in V : (v, w) \in E\}$. Each player has to guess the value of their own die. We want to calculate (assuming the players have agreed in advance on an optimal protocol) the probability that all the players (nodes) guess correctly their dice values simultaneously. Question 2 (in section II) corresponds to the case where G is the complete graph on n nodes.

Definition

A (cooperative) guessing strategy for the game GuessingGame(G, s) is a set of functions $f_\omega : \{1, 2, \ldots, s\}^{A_w} \rightarrow \{1, 2, \ldots, s\}$ with $\omega \in \{1, 2, \ldots, n\}$. Notice that each player (node) ω is assigned exactly one function f_ω.

In figure 3, we consider six simple examples:

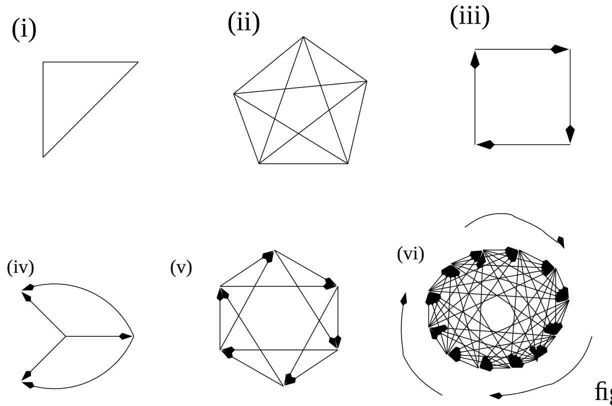

fig.3

Examples (i) and (ii) correspond to the dice guessing game we already considered (with 3 and 5 players). The players have a guessing strategy
that succeeds with probability $\frac{1}{s}$. In the guessing game based on (iii) (or in general the cyclic graph on n points) an optimal protocol appears if each node 'guess' that its own die value is the same as the value it receives. This strategy succeeds if each of the four dice has the same value i.e. with probability $(\frac{1}{s})^3$ (or in general $(\frac{1}{s})^{n-1}$). Though this probability is low, it is s times higher than if the players just make uncoordinated random guesses.

In (iv) the graph contains no cycles so the players cannot do any better than just guessing i.e. the players can achieve probability at most $(\frac{1}{s})^4$.

In (v) it can be shown that there are a number of distinct guessing strategies that guarantee the players' success with probability $(\frac{1}{s})^4$ (one, optimal strategy appears by dividing the graph into two disjoint cycles (triangles)).

Finally, in (vi) we consider a graph with 12 nodes (one for each hour on a clock) and edges from (i, j) if the 'time' from i to j is at most 5 hours. Using the type of argument we introduce later it is fairly simple to show that the players in the GuessingGame(G, s) have an optimal guessing strategy that ensures that the players with probability $(\frac{1}{s})^7$ (i.e. with a factor s^5 better than pure uncoordinated guessing) all simultaneously guess the value of their own die.

Definition

A graph $G = (V, E)$ has for $s \in N$ *guessing number* $k = k(G, s)$ if the players in GuessingGame(G, s) can choose a protocol that guarantees success with probability $(\frac{1}{s})^{|V|-k}$.

Thus the guessing number of a directed graph is a measure of how much better than pure guessing the players can achieve. If the players can achieve a factor s^k better than pure random uncoordinated guessing, the graph has guessing number $k = k(G, s)$. Notice that a directed graph has a guessing number for each $s = 2, 3, 4, \ldots$.

For many graphs (though not all) the guessing number is independent of s. The graphs in figure 3 have guessing numbers $2, 4, 1, 0, 2$ and 5 (independently of $s \geq 2$). From the definition there is no reason to believe that the guessing number of a graph is in general an integer. Yet remarkably many graphs have integer guessing numbers. Later we will show that there exist graphs for which the guessing number $k = k(G, s)$ (for alphabet of size $s \in N$) of a graph is not an integer. We will show that there exist graphs where the guessing number $k(G, s)$ even fails to be an integer for each $s \in \{2, 3, 4, \ldots, \}$.

Observation(A)

In GuessingGame(G, s) the graph G allows the players to do better than pure uncoordinated guessing if and only if G contains a loop.

Observation(B)

A graph $G = (V, E)$ contains a (directed) loop if and only if its guessing number is ≥ 1. If a graph contains k disjoint loops its guessing number $\geq k$ (for each $s \geq 2$). A graph is reflexive if and only it has guessing number $|V|$. Assume that the set of nodes V in the graph G can be divided in r disjoint subsets $V_1, V_2, \ldots, V_r$ of nodes such that the restriction of G to each subset V_j is a clique. Then the graph G has guessing number $\geq |V| - r$ (for each $s \geq 2$).

If a graph contains a (directed) loop the players can always guess in an "inconsistent" fashion (e.g. all but one players in the loop guess their own die value to be the same as the value they receive from the previous node in the loop. The 'odd one out' guesses inconsistently with the other players that his/her die value differ from the value received). If we combine this with Observation A, we notice the curious fact that, *the players have a "good" strategy that ensures that they all succeed with higher probability than uncoordinated random guessing if and only if the players have a "bad" strategy that insures they never succeed.*

Sometimes it is convenient to focus on certain more limited guessing strategies.

Definition

Let B be a class of functions $f : A^d \to A$ for $d = 1, 2, 3, \ldots$. An important class appears if we let A denote a fixed algebraic structure (e.g. a group, a ring or a vector space) of $s = |A|$ elements, and let the class $B = LIN$ consist of all homomorphisms (linear maps) $A^d \to A$ for $d = 1, 2, 3, \ldots$. If all the functions f_w belong to the class B we say the players have chosen a guessing strategy in B. If $B = LIN$ we say that the players use a linear guessing strategy.

Definition

A graph $G = (V, E)$ has *guessing number* $k = k_B(G, s)$ with respect to the functions in B if the players in GuessingGame(G, s) have a protocol with all guessing functions in B that guarantees success with probability $(\frac{1}{s})^{|V|-k}$. We say G has linear guessing number $k_{lin} = k_{lin}(G, s)$ if the players have a linear guessing strategy that guarantee success with probability $\geq (\frac{1}{s})^{|V|-k}$.

III. Network coding and guessing games

In this section we show that Mathematically there is a very close link between Network Coding and the guessing games we just defined. We will show that each information flow problem is equivalent to a problem about directed graphs.

The translation between information networks and directed graphs is most clean if we represent information networks such that we place all computations (Network Codings) in the nodes of the network. We refer to this representation as the *Circuit Representation*. This representation is of course "wrong" from a Network Coding perspecitive. However, from a mathematical perspective the different representations are essentially equivalent. In general the circuit representation is slightly more economical (usually save a few nodes) than the standard representation in Network Coding. The representation is more in line with circuit complexity, where the task of the network is in general a computational task. Formally, each source node is associated with a variable. Each node compute a function of incoming edges signals. Each outgoing edge from a node transmits the same signal (function value of node). Each receiver node is required to produce a specific input variable.

In general, given an information flow problem N (in the Circuit Representation) we obtain a directed graph G_N by identifying each source node with the corresponding receiver node.

In figure 4 we see a few examples of simple information networks together with their corresponding directed graphs.

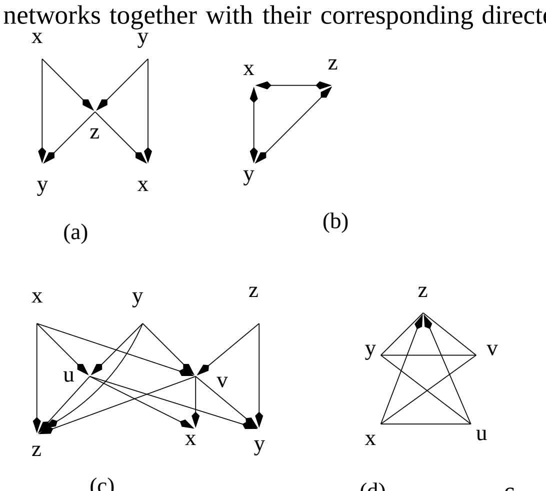

figure 4

The information network N in figure 4a (or figure 1b) is the usual 'butterfly' network (presented in Circuit Representation). If we identify the input node (source node) x with the output node (receiver node) x, and identify input node (source node) y with the output node (receiver node) y, we get the graph in figure 4b.

The information network in figure 4c does not have any obvious symmetries, but when input and output nodes are identified, we get the directed graph in figure 4d that clearly contains a number of symmetries. The translation shows that nodes x and u (as well as y and v) are equivalent points. That the points from a deeper mathematical perspective are also equivant in figure 4c is not obvious at a first glance (but this will follow from Theorem 1). The guessing number of the graph in (b), as well as the graph in (d), can be shown to have the value 2.

In general we let $C_{multiple-unicast}$ (the class of multiple-unicast directed information networks) consist of information networks N for which for some $n \in N$, n messages $m_1, m_2, \ldots, m_n \in A$ (selected from some alphabet A) have to be sent from input (source) nodes $i_1, i_2, \ldots, i_n$ to output nodes $o_1, o_2, \ldots, o_n$. Somewhat formally, each source node i_j is associated a variable x_j and each node w (except for the source nodes) are assigned a function symbol f_w representing a function f_w that is mapping all incoming signals $a_1, a_2, \ldots, a_{k_w}$ to an element $a = f(a_1, a_2, \ldots, a_{k_w}) \in A$. Each outgoing edge from a node transmits the same signal (the function value a of the node). Each receiver node is required to produce a specific input variable.

For an information network $N \in C_{multiple-unicast}$ we associate a directed graph G_N that appears by identifying each source (input) node i_j in N with its corresponding receiver (output) node o_j. If N has n input nodes, n output nodes and m inner nodes ($2n + m$ nodes in total) the graph G_N has $n + m$ nodes.

We are now ready to state the surprising link that shows that each information flow problem is equivalent to a problem about directed graphs.

Theorem(1)

An information Network flow problem $N \in C_{multiple-unicast}$ with n input/output nodes has a solution over alphabet A with $|A| = s$ elements if and only if the graph G_N has guessing number $k(G, s) \geq n$.

The main point of the theorem is that it replaces the flow problem - a problem that mathematically speaking involves slightly complicated concepts like *a set of source nodes*, *a set of receiver nodes* as well as *a set of requirements (demands)* that specifies the destination of each input - with an equivalent problem that can be expressed in pure graph theoretic terms (no special input or output nodes). Actually we show the theorem in a slightly stronger form:

Theorem(2)

The solutions (over alphabet A with $|A| = s$) of an information network flow problem $N \in C_{multiple-unicast}$ with n input/output nodes are in one-to-one correspondence with the optimal guessing strategies (over alphabet A with $|A| = s$). Each of these guessing strategies ensures that the players in the guessing game played on G_N have success with probability $(\frac{1}{s})^{|G_N|-n}$ (where $|G_N|$ is the number of nodes in G_N).

The following simple observation highlights (in quite a geometric fashion) the difference between Network Coding and traditional routing:

Observation(C)

An information flow network $N \in C$ has through put k using ordinary routing (i.e. pushing each message along a unique path) if and only the graph G_N contains k disjoint cycles.

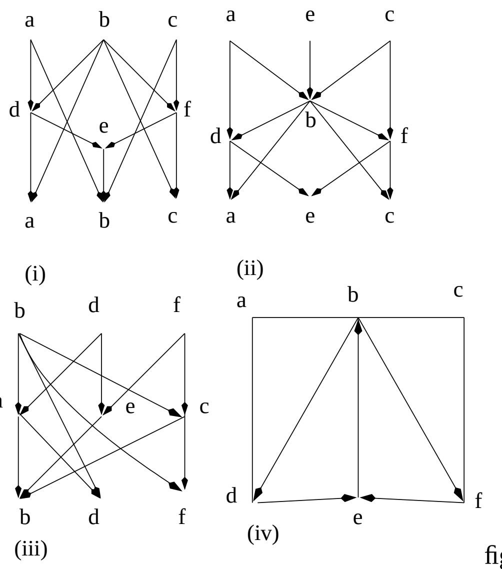

figure 5

Consider the three information flow problems in figure 5(i-iii). They are in circuit representation (i.e. all functions are placed in the nodes, and each outgoing edge from a node transmits the same function value). The three information networks in 5(i)-(iii) are non-isomorphic and are clearly distinct. However if we identify the source nodes and the receiver nodes in each of the networks, we get the *same* directed graph in figure 5 (iv).

According to Theorem 2 there is a one-to-one correspondence between solutions of each of the three information networks 5(i)-5(iii), and the successful strategies in GuessingGame(G, s). Thus, the set of solutions to each of the three information networks 5(i)-5(iii) is in a natural one-to-one correspondence. Before we prove Theorem 1 and Theorem 2, let us have a closer look at the networks in figure 5. A (cooperative) strategy for the players in the guessing game with the directed graph in figure 5 (iv) consists of 6 functions $g_1, g_2, \ldots, g_6$ such that:

$a^{guess} = g_1(b, d)$
$b^{guess} = g_2(a, c, e)$
$c^{guess} = g_3(b, f)$
$d^{guess} = g_4(a, b)$
$e^{guess} = g_5(d, f)$
$f^{guess} = g_6(b, c)$

For all players to guess their own message correctly we must have $a^{guess} = a$ i.e. we must have $a = g_1(b, d)$. Thus assuming that we work under the conditional situation with $a^{guess} = a$, we can substitute a with $g_1(b, d)$ leading to the equations:

$b^{guess} = g_2(g_1(b, d), c, e)$
$c^{guess} = g_3(b, f)$
$d^{guess} = g_4(g_1(b, d), b)$
$e^{guess} = g_5(d, f)$
$f^{guess} = g_6(b, c)$

Now pick any equation of the form $x^{guess} = h$ where x does not appear in the expression h. We might for example assume $c = g_3(b, f)$ (i.e. the $c^{guess} = c$). Substituting c with $g_3(b, f)$ in the equations we get:

$b^{guess} = g_2(g_1(b, d), g_3(b, f), e)$
$d^{guess} = g_4(g_1(b, d), b)$

$e^{guess} = g_5(d, f)$
$f^{guess} = g_6(b, g_3(b, f)$

This system of equations still contains one equation of the form $x^{guess} = h$ where x does not appear in the expression h. Let $e = g_5(d, f)$ (assuming $e^{guess} = g_5(d, f)$) and substitute this into the equations we get:

$b^{guess} = g_2(g_1(b, d), g_3(b, f), g_5(d, f))$
$d^{guess} = g_4(g_1(b, d), b)$
$f^{guess} = g_6(b, g_3(b, f))$

For any fixed choice of functions g_1, g_2, g_3, g_4, g_5 and g_6 let $0 \leq p \leq 1$ denote the probability that a random choice of b, d and f satisfies the equations (*):

$b = g_2(g_1(b, d), g_3(b, f), g_5(d, f))$
$d = g_4(g_1(b, d), b)$
$f = g_6(b, g_3(b, f))$

It is not hard to show that the probability that $a^{guess} = a$, $c^{guess} = c$ and $e^{guess} = e$ is $(\frac{1}{s})^3$ (essentially this is because the restriction of G to the nodes a, c and d form an acyclic subgraph. For a more general argument see the proof of Theorem 2). Thus, the conditional probability that the remaining players all guess correctly their own die value is p, and the probability all players are correct is $p(\frac{1}{s})^3$. Hence - in agreement with Theorem(1) - the guessing number of the graph in figure 5 (iv) is 3 if and only if there exist functions $g_1, g_2, \ldots, g_6$ such that the equations (*) hold for all b, d and f (i.e. hold with probability 1).

As it happens, we can solve the equations by turning the alphabet A into a commutative group $(A, \oplus)$, and the by letting $g_1(b, d) = b \oplus d, g_2(\alpha, \beta, \gamma) = \alpha \ominus \gamma$, $g_3(b, f) = b \oplus f$, $g_4(\alpha, \beta) = \alpha \ominus \beta$, $g_5(d, f) = d$ and $g_6(\alpha, \beta) = \beta \ominus \alpha$. Thus the players have a (cooperative) guessing strategy (in fact a linear guessing strategy) ensuring that all players are simultaneously able to guess their own message correctly with the probability $(\frac{1}{s})^3$. One strategy is given by:

$a^{guess} = b \oplus d$
$b^{guess} = a \ominus e$
$c^{guess} = b \oplus f$
$d^{guess} = a \ominus b$
$e^{guess} = d$
$f^{guess} = c \ominus b$

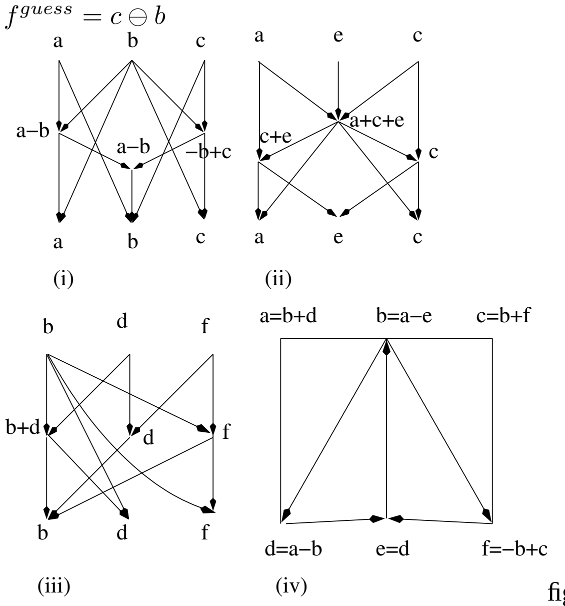

figure 6

Figure 6 (i)-(iii) shows how this strategy corresponds naturally to Network Codings in the three information flow problems in figure 5(i)-(iii). Figure 6 (iv) shows the strategy as a guessing strategy.

IV. Proof of Theorems

Before we prove Theorem 1 and Theorem 2, we need a few formal definitions of information networks. As already pointed out, the translation between information networks and directed graphs is most clean if we represent information networks such that all computations (Network Codings) are placed in the nodes of the network. An information flow network N (in circuit representation) is an acyclic directed graph with all source nodes (input nodes) having in-degree 0 and all receiver nodes (output nodes) having out-degree 0. Each source node is associated with a variable from a set Γ_{var} of variables. In the receiver node there is a demand assigned i.e. variable from Γ_{var}. In each node w that is not a source, a function symbol f_w is assigned. The function symbols in the network are all distinct.

Messages are assumed to belong to an alphabet A. Sometimes we assume that A has an additional structure (e.g. a group, a ring or a vector space). Each outgoing edge from a node transmits the same signal (function value of node).

An actual information flow is given by letting each function symbol f represent an actual function $\tilde{f} : A^d \rightarrow A$ where d is the number of incoming edges to the node that is associated the function symbol f. The information flow is a solution if the functions compose such that each demand is always met.

We let $C_{multiple-unicast}$ denote the class of information networks N for which n messages $m_1, m_2, \ldots, m_n \in A$ (selected from some alphabet A) have to be sent from input (source) nodes $i_1, i_2, \ldots, i_n$ to output nodes $o_1, o_2, \ldots, o_n$.

Let $C_{multiple-unicast}$ be an information network in this model. We define the graph G_N by identifying node i_1 with o_1, node i_2 with o_2, ... and node i_j with o_j in general for $j = 1, 2, \ldots, n$.

Theorem(1) follows directly from Theorem(2). Hence, to prove both theorems it suffices to prove Theorem(2).

Proof of Theorem(2): Let N be an information network with input (source) nodes $i_1, i_2, \ldots, i_n$, output (receiver) nodes $o_1, o_2, \ldots, o_n$ and inner nodes $n_1, n_2, \ldots, n_m$. The network N is acyclic so we can assume that we have ordered the nodes as $i_1 < i_2 < \ldots < i_n < n_1 < n_2 < \ldots < n_m < o_1 < o_2 < \ldots < o_n$ such that any edge (i, j) in N has $i < j$ in the ordering. Any selection of coding functions (whether they form a solution or not) can then be written as

$z_1 = f_1(x_1, x_2, \ldots, x_n)$
$z_2 = f_2(x_1, x_2, \ldots, x_n, z_1)$
$z_3 = f_3(x_1, x_2, \ldots, x_n, z_1, z_2)$
....
$z_m = f_m(x_1, x_2, \ldots, x_n, z_1, z_2, \ldots, z_{m-1})$
$x_1^o = g_1(x_1, x_2, \ldots, x_n, z_1, z_2, \ldots, z_m)$
$x_2^o = g_2(x_1, x_2, \ldots, x_n, z_1, z_2, \ldots, z_m)$
....
$x_n^o = g_n(x_1, x_2, \ldots, x_n, z_1, z_2, \ldots, z_m)$

where for $j = 1, 2, \ldots, n$ x_j is the variable denoting the value assigned to the input node i_j, z_j is the variable denoting the value computed by the inner node n_j, and x_j^o is the variable denoting the output value computed by the node o_j.

Next, consider the corresponding graph G_N we get by identifying nodes i_r and o_r for $r = 1, 2, \dots, n$. We consider the guessing strategy given by the functions above i.e. the strategy given by:

$z_1^{guess} = f_1(x_1^{real}, x_2^{real}, \dots, x_n^{real})$
$z_2^{guess} = f_2(x_1^{real}, x_2^{real}, \dots, x_n^{real}, z_1^{real})$
$z_3^{guess} = f_3(x_1^{real}, x_2^{real}, \dots, x_n^{real}, z_1^{real}, z_2^{real})$
....
$z_m^{guess} = f_m(x_1^{real}, x_2^{real}, \dots, x_n^{real}, z_1^{real}, z_2^{real}, \dots, z_{m-1}^{real})$
$x_1^{guess} = g_1(x_1^{real}, x_2^{real}, \dots, x_n^{real}, z_1^{real}, z_2^{real}, \dots, z_m^{real})$
$x_2^{guess} = g_2(x_1^{real}, x_2^{real}, \dots, x_n^{real}, z_1^{real}, z_2^{real}, \dots, z_m^{real})$
....
$x_n^{guess} = g_n(x_1^{real}, x_2^{real}, \dots, x_n^{real}, z_1^{real}, z_2^{real}, \dots, z_m^{real})$.

Here x_j^{real} (or z_j^{real}) denotes the actual value of the die associated to that node, while x_j^{guess} (or z_j^{guess}) denotes the value being 'guessed' by the node.

Conversely each guessing strategy for G_N can be written on this form. To see this we use the fact that the restriction of G_N to the nodes that correspond to the inner nodes in N forms an acyclic graph (since N is acyclic). Thus the equations can be viewed as an attempt to solve the information flow problem N. To prove Theorem(2) we show that the guessing strategy succeeds with probability $(\frac{1}{s})^m$ if and only if the corresponding information flow functions solves the information network problem. This boils down to showing that the probability that all inner nodes guess their own dice values correctly is $(\frac{1}{s})^m$ (Lemma 3). Assume that we have shown this (lemma 3). Then the probability that all players guess correctly is at most as large as the probability all players corresponding to inner nodes $n_1, n_2, \dots, n_m$ guess correctly. Thus all the players guess simultaneously their own dice values correctly with probability $\leq (\frac{1}{s})^m$. Equality holds if and only if the conditional probability (under the assumption that $z_j^{guess} = z_j^{real}$ for $j = 1, 2, 3, \dots, m$) that $x_j^{guess} = x_j^{real}$ for $j = 1, 2, \dots, n$ is 1. This happens if and only if the functions $f_1, f_2, \dots, f_m, g_1, \dots, g_n$ form a solution to the information flow problem. So to complete the proof of Theorem(2) it suffices to show:

Lemma (3)

For any set of functions $f_1, f_2, \dots, f_m$ and $g_1, g_2, \dots, g_n$ the probability that players $n_1, n_2, \dots, n_m$ (i.e. players in nodes corresponding to inner nodes in the information network) guess their own dice values correctly is $(\frac{1}{s})^m$ (i.e. independent of the chosen guessing functions).

Proof: We are asking for the probability $z_j^{guess} = z_j^{real}$ for $j = 1, 2, \dots, m$ where $z_1^{guess} = f_1(x_1^{real}, x_2^{real}, \dots, x_n^{real})$
$z_2^{guess} = f_2(x_1^{real}, x_2^{real}, \dots, x_n^{real}, z_1^{real})$
$z_3^{guess} = f_3(x_1^{real}, x_2^{real}, \dots, x_n^{real}, z_1^{real}, z_2^{real})$
....
$z_m^{guess} = f_m(x_1^{real}, x_2^{real}, \dots, x_n^{real}, z_1^{real}, z_2^{real}, \dots, z_{m-1}^{real})$.

The number of choices of $x_1^{real}, x_2^{real}, \dots, x_n^{real}$ and $z_1^{real}, z_2^{real}, \dots, z_m^{real}$ is s^{n+m}. We want to count the number of "successful" choices for which $z_j^{guess} = z_j^{real}$ for $j = 1, 2, \dots, m$. That is, the number of choices for which:

$z_1^{real} = f_1(x_1^{real}, x_2^{real}, \dots, x_n^{real})$
$z_2^{real} = f_2(x_1^{real}, x_2^{real}, \dots, x_n^{real}, z_1^{real})$
$z_3^{real} = f_3(x_1^{real}, x_2^{real}, \dots, x_n^{real}, z_1^{real}, z_2^{real})$
....
$z_m^{real} = f_m(x_1^{real}, x_2^{real}, \dots, x_n^{real}, z_1^{real}, z_2^{real}, \dots, z_{m-1}^{real})$

But for each choice of $x_1^{real}, x_2^{real}, \dots, x_n^{real}$ there is exactly one choice of $z_1^{real}, z_2^{real}, \dots, z_m^{real}$. Thus the number of successful choices is s^n. The probability is $\frac{\text{number of successful choices}}{\text{number of choices}} = \frac{s^n}{s^{n+m}} = \frac{1}{s^m}$. ♣

Informally we can explain the validity of Lemma 3 by noticing that the restriction of the graph G_N to the set of nodes that corresponds to inner nodes of N forms an acyclic graph, and as we already noticed in an acyclic graph, the players cannot do any better than uncoordinated guessing.

A. Standard representation

There are a few slightly different ways to represent flow in information networks. In the previous section we considered the Circuit Representation. We call the standard (and "correct") way of representing information flows in Network Coding for the **Standard Representation**. If we use the Standard Representation we get slightly different versions of Theorem(1) and Theorem(2). The actual theorems can be stated the same (verbatim)! The Theorems are modified to fit the standard representation in the way the graph G_N is defined.

An information Network N is a directed acyclic multi-graph. Each source node has in-degree 0, while each receiver node has out-degree 0. Associated with each source node is a variable from a set Γ_{var} of variables. Each outgoing edge is associated with a distinct function symbol with an argument for each incoming edge. Each receiver node has a list of demands which is a subset of variables from Γ_{var}. In the receiver node there is assigned a function symbol for each demand. All function symbols are distinct.

Messages are assumed to belong to an alphabet A. An actual flow (using Network Coding) is given by letting each function symbol f represent an actual function $\tilde{f} : A^d \rightarrow A$ where d is the number of incoming edges to the node that is associated with the function symbol f. The flow (that might utilise Network Coding) is a solution if the functions compose such that each demand is given by the functional expression of the involved terms.

We let $C_{multiple-unicast}$ denote the class of information networks N for which n messages $m_1, m_2, \dots, m_n \in A$ (selected from some alphabet A) has to be sent from source nodes $i_1, i_2, \dots, i_n$ to output nodes $o_1, o_2, \dots, o_n$.

We convert a given information network $N \in C_{multiple-unicast}$ to a directed graph G_N as follows:

Step 1: For each variable or function symbol assigned to an edge or a node we introduce a node in the new graph G_N.

Step 2: We identify nodes i_i with o_1, i_2 with o_2, ... and i_j with o_j in general for $j = 1, 2, \dots, n$.

With this translation of N to G_N Theorem(1) and Theorem(2) remain valid (verbatim).

V. General results for information networks

Theorem 1 and Theorem 2 only apply for information networks $N \in C_{multiple-unicast}$. In this section we generalise the results so they essentially cover all (!) instantaneous information networks. As a price of the increase in generality we lose some of the elegance of Theorem 1. The proof of Theorem 4 (that generalises Theorem 1) provides us, in fact, with a new (and different) proof of Theorem 1.

Let N be a information network, and let A be an (finite) alphabet with s elements. For a selection of fixed network functions $\bar{f}$, we define the networks N's *global success rate* $p(N, s, \bar{f})$ of a specific network coding flow (with coding functions $\bar{f}$) as the probability that *all* outputs produce the required outputs if all inputs are selected randomly with independent probability distribution. The *maximal global success rate* $p(N, s)$ of the information flow network N (over alphabet of size s) is defined as the supremum of all global success rates $p(N, s, \bar{f})$ that can be achieved by any choice of coding functions $\bar{f}$. Since the set of functions $\bar{f}$ (for a fixed finite alphabet A) is finite, $p(N, s)$ is the maximal global success rate $p(N, s, \bar{f})$ for some specific choice of coding functions $\bar{f}$.

Assume that N is an information network over an alphabet A with s elements. Assume that N has n source nodes (input nodes), and that each of these is required by at least one receiver node (output node).

Definition

We define the *source transmission bandwidth* $k = k(N, s)$ of the information network N (over alphabet of size s) as $k(N, s) = \log_s(p(N, s)) + n$.

The notion is motivated by the Theorem(4) below, and can be viewed as a generalisation of the guessing number of a graph.

Notice, that a network has source transmission bandwidth k if all output nodes can simultaneously calculate their required messages with probability s^k higher than what can be achieved by the "channel free" network. An information network N is solvable if and only if $p(N, s) = 1$ i.e. if and only if $k(N, s) = \log_s(p(N, s)) + n = n$. It other words, an information network N with n sources (that send n distinct source messages that each message is required at one or more receiver nodes), has source transmission bandwidth $k(N, s) = n$ if and only if it is solvable (in the sense of network coding) over an alphabet of size s.

For each (directed) graph $G = (V, E)$ we want to define an information flow problem $N_G = (W, F)$ with $|V|$ source nodes (input nodes) and $|V|$ receiver nodes (output nodes). Expressed slightly informally, we define N_G by splitting each node $w \in V$ into two nodes w_{input} and w_{output} (thus the vertex set W consists of two copies of V). For each edge $(w, v) \in E$ we add an edge $(w_{input}, v_{output}) \in F$. Let $N_G = (W, F)$ denote the flow problem that appears through this transformation where each output node v_{output} requires the message assigned to v_{input}. Notice that the information network N_G is usually very far from being solvable, since most source (input) nodes have no path to its corresponding receiver (output) node.

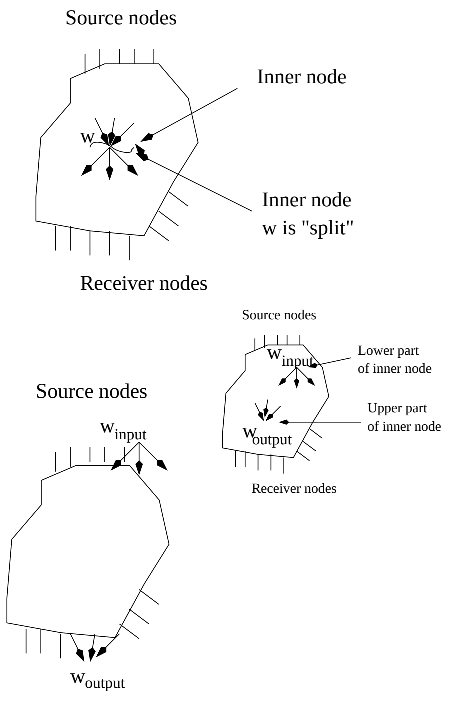

figure 7

Observation

Let G be a directed graph. Then $N_G \in C_{multiple-unicast}$ and G has guessing number $k = k(G, s)$ if and only if N_G has source transmission bandwidth $k = k(N_G, s)$.

For each $p \in [0, 1]$ there is a one-to-one correspondence between guessing strategies $\bar{f}$ in the GuessingGame(G, s) that achieve success with probability p and information flows $\bar{f}$ in N_G that have global success rate p.

The Observation is too trivial to deserve to be called a theorem. It is, however, quite interesting since it shows (together with the remark in the end of this section) that the notion of source transmission bandwidth generalises the guessing number of a directed graph.

We now introduce a simple move we call "split". Given an information network $N = (V, E)$ (with E being a multiset) the move "split" can be applied to any inner node $w \in V$ in N (a node is an inner node if it is not a source or a receiver node). The move "split" copies the inner node w into two nodes w_{input} and w_{output}. In other words, the move converts the vertex set V to the set $V' = V \cup \{w_{input}, w_{output}\} \setminus \{w\}$ containing all points in V but with two copies (w_{input} and w_{output}) of w. For each outgoing edge $(w, u) \in E$ from w we introduce an edge $(w_{input}, u) \in E'$ (with the same multiplicity as (w, u)). For each incoming edge $(u, w) \in V$ we introduce an edge $(u, w_{input}) \in E'$ (with the same multiplicity as (w, u)).

The information network $N' = (V', E')$ has as source (input) nodes all source (input) nodes in V together with $\{w_{input}\}$. The set of receiver (output) nodes consists of the receiver (output) nodes in V together with $\{w_{output}\}$. We associate a new variable z with the node w_{input} and node w_{output} is associated with the demands z. All other nodes keep their demands.

In figure 7, we see how the split move can be applied. We say that the information network N' appears from the information network N by a "reverse split move", if N appears from N' using a split move.

The split move always results in an information network that has no solution (since there is no path from the source node w_{input} to the receiver node w_{output}).

The next Theorem can be considered as a generalisation of Theorem 1 and Theorem 2.

Theorem(4)

Let N and N' be two information networks that appear from each other by a sequence of split and inverse split moves (in any order). The networks N and N' have the same source transmission bandwidth (i.e. $k(N,s) = k(N',s)$)

More specifically, let N be an information flow network, let A be an alphabet with $|A| = s$ letters, and assume that $\bar{f}$ is a selection of coding functions over this alphabet. Assume that N has source messages $x_1, x_2, \ldots, x_r$ (they might be transmitted from more than one input edge). Assume that the coding functions have a global success rate $p = p(N,s,\bar{f}) \in [0,1]$. Let N' be any information network that appears from N by application of the split move. Then N' with the coding functions $\bar{f}$) has global success rate $p(N',s,\bar{f}) = \frac{p}{s}$.

In general if N has global success rate p (over alphabet A) any network N' that appears from N by application of r split moves and t reverse split moves (in any order) has global success rate $p \times s^{t-r}$.

Proof: The first part follows from the more detailed second part, since each application of the split rule increases the value of n (the number of input nodes by one), and each application of the inverse split rule decreases the value of n by one.

Assume that the information network $N = (V,E)$ has global success rate $p = p(N,s,\bar{f}) \in [0,1]$ with respect to the coding functions $\bar{f}$. Let $w \in V$ be any inner node in N. Replace w with (split w into) two nodes w_{input} and w_{output} as already explained. The incoming coding function to node w_{output} is the same function as the inner coding function to node w in the network N. Each outgoing coding function of w_{input} is the same as each outgoing function for node w. The network N' has got a new input node. Let us calculate the probability $p(N',s,\bar{f})$ that all output nodes produce the correct outputs. The probability that node w_{output} produces the correct output is exactly $\frac{1}{s}$. Assume now $w_{output} = w_{input}$. The conditional probability (i.e. the probability given $z_{output} = z_{input}$) that all output nodes in the network N produce the correct output is $p = p(N,s,\bar{f})$. But, then the probability that all output nodes in N' produce the correct output is exactly $\frac{p}{s}$.

The second part of the theorem follows from the first part. Assume that $\bar{f}$ is a selection of coding functions such that $p(N,s,\bar{f}) = p(N,s)$ (the alphabet is finite so there are only finitely many functions $\bar{f}$, and thus there exist functions that achieve the maximum value $p(N,s)$). We already showed that $p(N',s,\bar{f}) = \frac{p(N,s,\bar{)}}{s}$. We claim that $p(N',s) = p(N',s,\bar{f})$. Assume that $p(N',s,\bar{g}) > p(N',s,\bar{f})$. But then $p(N,s,\bar{g}) = s \times p(N',s,\bar{g}) > s \times p(N',s,\bar{f}) = p(N,s,\bar{f})$ which contradicts the assumption that $\bar{f}$ was an optimal coding function for the information network N (over alphabet of size s). ♣

Remark: Notice that if N' can be derived from N by a split move the graphs G_N and $G_{N'}$ are identical. Thus starting with any $N \in C_{multiple-unicast}$ any sequence of split and inverse split moves leave G_N unchanged. This shows that G_N is invariant under split and unsplit moves on N.

If $N \in C_{multiple-unicast}$ has n input nodes, n output nodes, and m inner nodes we can 'split' all inner m nodes producing a bi-parte graph $\hat{N}$ with $n+m$ input nodes and $n+m$ output nodes. Notice that there is a one-to-one correspondance between information flows in $\hat{N}$ and guessing strategies in G_N (actually, from a mathematical perspecitive the two are essentially identical since the coding functions in the output nodes are the guessing functions in G_N).

Now assume that the $n+m$ input values of $\hat{N}$ are chosen randomly (and independently). The probability $p(\hat{N},s)$ that all outputs are correct is identical to the probability p that each player in the guessing game G_N correctly guesses his/her own die value.

But, now according to Theorem 4, $p(\hat{N},s) = (\frac{1}{s})^m p(N,s)$ (since $\hat{N}$ appear from N using m split moves). In other words N is solvable (i.e. $p(N,s)=1$) if and only if $p(\hat{N},s) = (\frac{1}{s})^m$. This happens if and only if the players in the guessing game played on G_N (=$G_{\hat{N}} \equiv \hat{N}$) have a strategy that succedes with probability $(\frac{1}{s})^m$. Thus the guessing number of G_N is $(n+m)-m = n$. This shows that Theorem 4 implies Theorem 1 (and that the proof of Theorem 4 together with the remark provide an alternative proof of Theorem 1).

VI. Utilising Public information.

A. Another Game

Consider the directed graph in figure 8(i) (introduced in 4 (iv)). Each node has to derive their own message. This is, of course, impossible and we know that the best the players can hope for (if they use a suitable coordinated guessing strategy) is that they are all correct on $s3$ distinct inputs (out of the $s6$ different inputs). If the players have access to $s3$ public messages and these are carefully chosen, it is possible for the players (through a cooperative strategy) to ensure that each player can derive his/her own message.

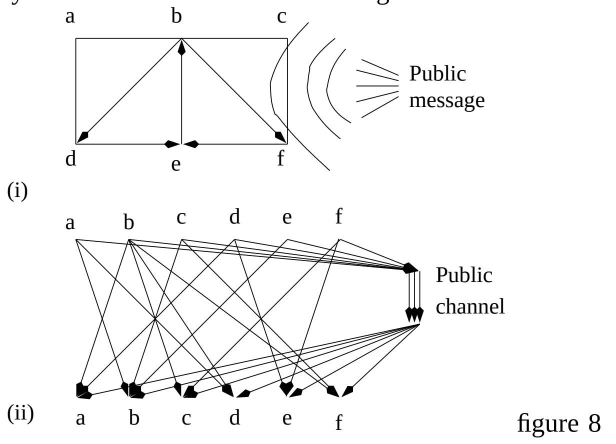

figure 8

If, for example, the values $a \ominus b \ominus d \in A, c \ominus b \ominus f \in A$ as well as $e \ominus d \in A$ are common knowledge (broadcast through public messages) each node can derive its own message (since $a = (a \ominus b \ominus d) \oplus b \oplus d, b = (e \ominus d) \ominus (a \ominus b \ominus d) \oplus (a \ominus e), c = (c \ominus b \ominus f) \oplus b \oplus f$, $d = (a \ominus b) \oplus (a \ominus b \ominus d)$, $e = (e \ominus d) \oplus d$ and $f = (c \ominus b) \ominus (c \ominus b \ominus f)$).

Another equivalent way of stating this is to consider the biparte flow problem in figure 8 (ii), with public channel of bandwidth 3. Notice that figure 8 (i) and figure 8 (ii) are different representations of the problems that are mathematically equivalent.

Are the solutions (public messages $a \ominus b \ominus d \in A, c \ominus b \ominus f \in A$ as well as $e \ominus d \in A$) in figure 8 (i) and figure 8 (ii) optimal? Is it possible to send fewer than s^3 message through the public channel (and still have all players being able to deduce their own message)? From the analysis of the guessing game in figure 4 (iv) we know that the probability that the players in nodes a, c and e guess their own messages is independent (for any guessing strategy) and thus nodes a, c and e guess correctly their own message with probability $(\frac{1}{s})^3$. We claim that if node a, c and e in general are able to derive their own message they must have access to at least s^3 distinct messages in the public channel. To see this assume that it were possible for the players in figure 8 (i) to deduce their own messages from a public channel that sends $< s^3$. The players could then all agree to guess *as if* the public channel is broadcasting a specific message m they agreed on in advance. Since there are less than $s3$ public messages there is a message m that is broadcast with probability $> (\frac{1}{s})^3$). This contradicts the fact that the players (especially the players in nodes a, c and c) cannot do better than $(\frac{1}{s})^3$. Thus the solutions in figure 8 (i) (and in figure 8 (ii)) are optimal.

Let $G = (V, E)$ be a directed graph. Assume like before that each node is being assigned a message x randomly chosen from a fixed finite alphabet A containing $s = |A|$ elements. Like in the guessing game each node transmit their message (dice value) along all outgoing edges. In other words each node j know the messages (dice values) of exactly all nodes i with $(i, j) \in E$.

The task of the players is to deduce their own message. This is of course impossible (unless the graph is reflexive) since in general the players have no direct access to their own message (dice values). The task of the players is to cooperate and agree on a protocol and a behaviour of a public channel that ensure that all players are always able to derive their own messages.

Definition

Let $G = (V, E)$ be a directed graph and let A denote an alphabet with s letters. Let P be a finite set of public messages. Consider the following game PublicChannelGame(G, A, P). The game is played as follows. Each node $j \in V$ is assigned a message $x_j \in A$. A public message $p = p(x_1, x_2, \ldots, x_n) \in P$ (given by a function $p : A^n \to P$) is broadcast to all nodes. Each node j have access to the message $p \in P$ as well as x_i for each i with $(i, j) \in E$. In the game each player j needs to deduce the content of their own message x_j.

Each player (node) $v \in \{1, 2, \ldots, n\}$ send their message to each player $w \in \{1, 2, \ldots, n\}$ with $(v, w) \in E$. Or in other words each node w receive messages from a set $A_w := \{v \in V : (v, w) \in E\}$. The task is do design the function $p(x_1, x_2, \ldots, x_n)$ such that each player always (i.e. for any choice of $x_1, x_2, \ldots, x_n \in A$) can deduce their own message. If this is possible, we say that the game PublicChannelGame(G, A, P) has a solution.

Definition

A directed graph $G = (V, E)$ has *(general) linear guessing number* $k = k_s$ if the game PublicChannelGame(G, A, P) has solution for some A with $|A| = s$ and with $P = s^{|V|-k}$.

This definition anticipates Theorem 6.

Now consider, for example, the case of figure 3(iii) with 4 players holding messages (dice values) x_1, x_2, x_3 and x_4. In this case each player is able to calculate their own dice value if, for example, $x_1 \oplus x_4, x_2 \oplus x_4$ and $x_3 \oplus x_4$ modulo s were know public information. [To see this, notice that node 1 receives x_4 from which it can calculate $x_1 = (x_1 \oplus x_4) \ominus x_4$, node $i = 2, 3$ receives x_{i-1} from which it can calculate $x_i = (x_i \oplus x_4) \ominus (x_{i-1} \oplus x_4) \oplus x_{i-1}$. Finally, node 4 receives x_3 from which it can calculate $x_4 = (x_3 \oplus x_4) \ominus x_3$].

For any information network N we can apply the split move until all inner nodes have been spilt. In this case N becomes a biparte graph B_N with no inner nodes. Notice that B_N is uniquely determined by N.

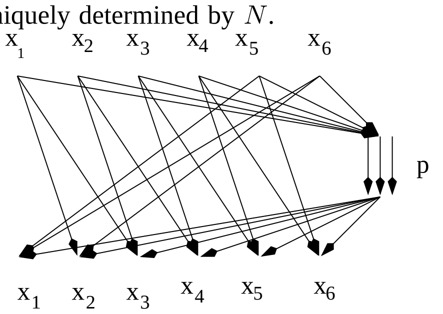

figure 9

This example and the example in figure 4 (iv), suggests that it is always is possible to replace the guessing part of guessing game, and instead let all players have access to a suitable public channel of information. We will show (corollary 10) that this is possible for linear solutions (also sometimes called matrix linear) for the guessing game, but never possible if only non-linear solutions exists. Notice, that the following analysis is only meaningful when the alphabet (i.e. the dice values) can be organised as a vector space U (of dimension d) over a finite field F (with a number q of elements being a prime power). The number $|U|$ of element of U is given by $s := q^d$.

Theorem(5)

Assume that the alphabet U is a vector space of dimension d over a finite field F with q elements (i.e. q is a prime power). Then the following are equivalent:

(1) The players have a linear guessing strategy in GuessingGame(G, U) that succeed with probability $(\frac{1}{q^d})^k$

(2) G has linear guessing number $k = k_{lin}(G, q^d)$.

(3) PublicChannelGame(G, U, U^k) has a solution

(possible non-linear).

(4) The Biparte information Flow problem B_G associated to G has a solution (over U and possible non-linear) that uses a public channel P of bandwidth k.

(5) The Biparte information Flow problem associated to G has a linear solution over a field $\tilde{F}$ of q^d elements, that uses a public channel of bandwidth k.

(6) The Biparte information Flow problem associated to G has a linear solution (over the vector space U) that uses a public channel of bandwidth k.

(7) PublicChannelGame(G, U, U^k) has a linear solution.

From this we get:

Theorem (6)

Assume that the alphabet U is a finite dimensional vector space over a finite field F. The nodes in a directed graph G can calculate their messages (selected from U) if they have access to a public channel of bandwidth $\leq k$ if and only if the (special) linear guessing number of G is $\geq |V| - k$.

Theorem (6) explain the terminology *general linear guessing number*. In the case where the alphabet is a vector-space the linear guessing number (in sense of linear maps) agree with the *general linear guessing number*. The two notions of linear guessing number agree when they are both defined. The general linear guessing number is, however, defined for all $s \in \{2, 3, 4, \dots, \}$, while the linear guessing number only is defined when s is a prime power (since a finite dimensional vector space always has a number of elements being a prime power).

B. Proof of theorem 5

First notice that (1) and (2) are equivalent (by definition). We claim:

Lemma 7

(1) implies (3):

Proof: We are given a graph $G = (V, E)$ and we consider GuessingGame(G, U, U^k), for U being a vector space of dimension d over a field F with q elements (q being a prime power). The number k is given by (1). We assume that the players have a linear guessing strategy, i.e. a strategy with all functions $f_w : U^{r_\omega} \to U$ are linear (i.e. given by a $r_\omega d \times d$ matrix with entries in F). Further more we assume this linear guessing strategy make it possible for the players to guess correctly all their own dice values with probability $(\frac{1}{q^d})^k$.

Consider $\tilde{U} := U^{|V|}$, the linear subspace of vectors $(v_1, v_2, \dots, v_{|V|}) \in U^{|V|}$ with $v_j \in U$ for $j = 1, 2, \dots, |V|$. Let $W \subseteq \tilde{U}$ denote the linear subspace of dice values for which the players all successfully guess their own dice value (while using the linear guessing strategy we assume exists). Since the strategy is successful with probability $(\frac{1}{q})^{dk}$ and since the number of points in $\tilde{U}$ is $q^{d|V|}$ the number of points in W is $q^{d|V|-dk}$. Since W is a linear subspace with $q^{d|V|-kd}$ points over a field of q elements its vector space dimension is $d|V| - dk$ (thus dk must be an integer).

For each vector $u \in \tilde{U}$ we consider the linear "side" space $u + W$. Let $u_0 (= 0), u_1, u_2, \dots, u_l$ denote a maximal family of vectors with $W = u_0 + W, u_1 + W, u_2 + W, \dots, u_l + W$ all being disjoint. It follows that $l = q^{dk} - 1$ i.e. that there are q^{dk} disjoint side spaces of W and that $\bigcup_{j=0}^{l} (u_j + W) = U$.

We can now convert this into a solution to PublicChannelGame(G, U, U^k). We do this by broadcasting a public message as follows: Assume each node in V has been assigned a value from U. The information of all dice values are contained in a vector $u \in \tilde{U}$. There exist exactly one index $j \in \{0, 1, 2, \dots, l\}$ such that $u \in u_j + W$. Broadcast the index $j \in \{0, 1, 2, \dots, q^{dk} - 1\}$ by selecting a bijection from $\{0, 1, \dots, q^{dk} - 1\}$ onto U (this is possible since U contains exactly q^{dk} points). Now each node can calculate its own message by correcting their guess (they would have made had they played the Guessing Game) by the suitable projection of u_j.

This shows that the game PublicChannelGame(G, U, U^k) has a solution (possible non-linear) with the public message being selected from the set U^k public messages. ♣

In this construction, the public channel broadcasts different messages for each index $j \in \{0, 1, 2, \dots, l\}$. In general, this map is not linear. We will show that any non-linear strategy can be turned into a linear strategy.

Lemma 8

(4) implies (5)

Before we prove this implication we make a few general observations and definitions. Assume the bi-parte flow problem in (5) (in theorem 5) has a solution with the public channel broadcasting $p_1(x_1, x_2, \dots x_n), \dots p_w(x_1, x_2, \dots, x_n)$. Since $p_j : A^n \to A$ and A is a field, each function p_j can be expressed as a polynomial $p_j \in A[x_1, x_2, \dots x_n]$. Each output node o_j receive $p_1, p_2, \dots p_w \in A$ as well as $x_{j_1}, x_{j_2}, \dots, x_{j_v} \in A$. The task of output node o_j is to calculate $x_j \in A$. For any polynomial $q \in A[x_1, x_2, \dots x_n]$ we let $L(q) \in A[x_1, x_2, \dots x_n]$ denote the sum of all monomials (with the original coefficients) of q that only contains one variable (e.g. x_j, x_j3, or x_j7). In other words $L(q)$ consists of q where the constant term as well as all monomials containing more than one variable have been removed. If for example $q = 5x_1x_3 - 7x_1x_2 + 3x_1 - 5x_2 + 1$, then $L(q) = 3x_1 - 5x_2$.

In the following lemma we assume that U $(= A)$ is structured as a finite field (containing q^d-elements).

Lemma(9)

A biparte information flow problem B has a solution with public information given by polynomials $p_1, p_2, \dots p_w \in A[x_1, x_2, \dots x_n]$ then B has a solution with public information given by linear expressions $l_1, l_2, \dots l_w \in A[x_1, x_2, \dots, x_n]$.

Remark: It is instructive to notice some of the reasons why in general non-linear flows cannot be eliminated from information networks. In a general network a non-linear solution might for example involve that two nodes send messages $(x + y)$ and $(y + z)$ to a node r where their product $(x + y)(y + z) = xy + xz + yz + y^2 = xy + xz + yz + y$ is being calculated. Removing mixed monomials would lead to $L(x + y) = x + y$ and $L(y + z) = y + z$ to be sent to node r

where $L((x+y)(y+z)) = y2$ must be calculated. Since it is not possible to derive $y2$ (or y) from $x+y$ and $y+z$ the process of removing monomials with mixed variables fails in general. The networks in [18] and [8] show that certain flow problems only have non-linear solutions. For such networks any attempt of removing non-linear terms (not just using local procedures) will fail. The point of lemma 9 is that the network B together with any public channel is structured in such a fashion that allows us to remove mixed terms and then replace the resulting function with linear functions. Information networks in which only two messages are transmitted provide another case where linearisation is always possible [9].

Proof of lemma(9): Assume that we are given a biparte flow problem B that have a solution with public information given by polynomials $p_1, p_2, \ldots p_w \in A[x_1, x_2, \ldots x_n]$. Assume that in this solution the output nodes $o_1, o_2, \ldots, o_n$ have assigned coding functions $f_1, f_2, \ldots, f_n$. Assume the underlying alphabet A has q elements (q is a prime power). Then we can organise A as a field. Each function $f : A^r \rightarrow A$ (where A is a field) can be expresed as a polynomial $p \in A[x_1, x_2, \ldots, x_r]$. Thus without loss of generality we can assume that the functions $f_1, f_2, \ldots, f_n$ are polynomials in $A[x_1, x_2, \ldots, x_n, z_1, z_2, \ldots, z_w]$. Since output node o_j requires x_j for each $j \in \{1, 2, \ldots, n\}$, the polynomial equation

$$f_j(x_1, x_2, \ldots, x_n, p_1, p_2, \ldots, p_w) = x_j$$

Let $\tilde{f}_j(x_1, x_2, \ldots, x_n, z_1, z_2, \ldots, z_w) = Lf_j(x_1, x_2, \ldots, x_n, z_1, z_2, \ldots, z_w)$ and let $l_j(x_1, x_2, \ldots, x_n)$. The polynomials $\tilde{f}_j$ and l_j have no mixed terms. In general for polynomials q, p $L(p+q) = L(p) + L(q)$ and $L(p_1, p_2, \ldots, p_s) = L(L(p_1)L(p_2) \ldots L(p_s))$. From this it is not hard to show that $\tilde{f}_j(x_1, x_2, \ldots, x_n, l_1, \ldots, l_w) = x_j$. In other words if we apply the operator L that removes all monomials with two or more distinct variables the public information then becomes $L(p_1), L(p_2), \ldots L(p_w)$. These functions can be realised (since there are no restrictions on the public channel and all functions $A^n \rightarrow A^w$ can be calculated). Using the same argument we can remove all mixed terms and insure that each output node o_j receive a function $\tilde{f}_j$ of its inputs (the input from input nodes as well as from the public channel). Since each of the equations $\tilde{f}_j(x_1, x_2, \ldots, x_n, l_1, \ldots, l_w) = x_j$ hold for $j = 1, 2, \ldots, n$ B has a solution with public information given by the linear expressions $l_1, l_2, \ldots l_w \in A[x_1, x_2, \ldots, x_n]$ This completes the proof of lemma(9). ♣

Now is it easy to prove Theorem(5). We have shown $(1) \rightarrow (3)$ (Lemma 7) , as well as $(4) \rightarrow (5)$ (Lemma 8).

The implication $(5) \rightarrow (6)$ follows from the fact that a linear map $f : \tilde{F}^r \rightarrow \tilde{F}$ is a (matrix) linear map from $U^r \rightarrow U$ for any vectorspace over F where F is a subfield of $\tilde{F}$.

The implications $(6) \rightarrow (7) \rightarrow (1)$ as well as $(3) \leftrightarrow (4)$ are all almost trivial and are left as easy exercises for the reader. This completes the proof of Theorem (5). Theorem (6) follows as an easy corollary.

VII. Some Corollaries

In general the guessing game GuessingGame(G, s) might only have non-linear optimal guessing strategies. When this happens G has linear guessing number k_{lin} that is strictly smaller than G's guessing number k. We have the following characterisation:

Corollary(10)

Let $G = (V, E)$ be a graph and let U be a finite vector space. The linear guessing number k_{lin} of G over U is smaller or equal to the guessing number k of G. Equality holds if and only if PublicChannelGame$(G, U, U^{|V|-k})$) is solvable.

We have seen that the problem of solving information network flow problems (of class $C_{multiple-unicast}$) can be restated to that of calculating the guessing number of a graph. The linear guessing number of a graph is an important concept. We have the following version of Theorem 1 (that could -with minor modifications - also be proved as Theorem 1).

Corollary(11)

The information flow problem $N \in C_{multiple-unicast}$ with n input/output nodes has a linear solution (i.e. a solution within the "wave paradigm") over an alphabet of size s if and only if G_N has its linear guessing number $k(G, s) \geq n$ (which happens if and only if $k(G, s) = n$).

VIII. Algebraic calculations of linear guessing numbers

Consider a directed graph $G = (V, E)$. In this section we show that the linear guessing number (over a field) has an algebraic definition.

Let $M = (m_{ij})_{i,j}$ be a $n \times n$ 0/1-matrix and let A be a finite field. Then we 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 $m'_{ij} = 0$ whenever $m_{ij} = 0$. Let I denote the $n \times n$ identity matrix and let $\mu(M) := n - min_{M' \in \mathcal{C}_{\mathcal{A}}(\mathcal{M})} rank(I + M')$. Notice that $\mu(M) \in \{0, 1, 2, \ldots, n\}$.

Theorem 12

Let G be a (directed) graph with n nodes and incidence 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 $\mu(G_N) := n - min_{M' \in \mathcal{C}_{\mathcal{A}}(\mathcal{M}_{\mathcal{G}})} rank(I + M') = k$.

Proof: Assume G has linear guessing number k. According the Theorem (5) G has linear guessing number k if and only if the PublicChannelGame(G, A, A^k) has a linear solution S with a public channel of bandwidth k. We say an edge $(v_1, v_2) \in E$ in G is active (with respect to the solution S) if the message in v_1 affects the guessing function in v_2. Let $E' \subseteq E$ consists of all active edges in G. Let $G' = (V, E')$ be the subgraph of G that consists of all active edges in G. For each active edge we assign a value $\alpha \in A \setminus \{0\}$. Consider a node $w \in V$ such that $(v_1, w), (v_2, w), \ldots, (v_d, w)$ are all active incoming edges with assigned values $\alpha_1, \alpha_2, \ldots, \alpha_d \in A \setminus \{0\}$. The (linear) signal being send to node w is $s = \alpha_1 m_{v_1} + \alpha_2 m_{v_2} + \ldots + \alpha_d m_{v_d}$ i.e. the waited sum of all incoming signals, as well as the signals that are send from the public channel. Since node w requires message m_w the public channel nust send a message from which the message $s + m_w$ (i.e.$\alpha_1 m_{v_1} + \alpha_2 m_{v_2} + \ldots + \alpha_d m_{v_d} + m_w$) can be derived.

Next assume that the rank of $ref(G')$ is k for some $G' \subseteq G$. Let $l_1(x_1, x_2, \ldots x_n), l_2(x_1, x_2, \ldots x_n), \ldots l_k(x_1, x_2, \ldots x_n)$ denote the k linearly independent rows of $ref(G')$. Send these signals as public messages. Let w be an arbitrary node. The node receive a signal $m_{v_1} + m_{v_2} + \ldots + m_{v_r}$ from the channels in G'. The node w need to derive m_w so it suffice to show that the node w can derive $m_{v_1} + m_{v_2} + \ldots + m_{v_d} + x_w$ from the public messages. But, the row $m_{v_1} + m_{v_2} + \ldots + m_{v_d} + m_w$ appears in $ref(G')$ and thus it belong to the span of the k vectors $l_1(x_1, x_2, \ldots x_n), l_2(x_1, x_2, \ldots x_n), \ldots l_k(x_1, x_2, \ldots x_n)$ that are send through the public channel. ♣

For a graph G let $Ref(G)$ denote the reflexive closure of G. Let $rank(G)$ denote the rank over the field $\{0, 1\}$ of the incident matrix of G.

Theorem(13)

Assume the alphabet $A = \{0, 1\}$ only contains two elements. Let G be a graph. Then PublicChannelGame$(G, \{0, 1\}, \{0, 1\}^k)$ has a solution if and only if

$$k \geq min_{G' \subseteq G} rank(Ref(G'))$$

IX. More Games

Suppose $N \in C_{multiple-unicasts}$ is an information network where some nodes have in-degree > 2. For each node n with in-degree $d > 2$ we can replace the incoming d edges with a tree with d leaves and a root in n.

Theoretically this replacement restricts the power of the information network since not all functions $f : A^d \to A$ can be written as a composition of $(d-1)$ functions $g_j : A2 \to A$, with $j = 1, 2, \ldots, d-1$.

Let S_d denote the class of d-ary functions $f : A^d \to A$ that can be written as a composition of $d-1$, 2-ary functions.

Given a directed graph $G = (V, E)$ and assume that each node with in-degree d can only compute functions that belong to S_d. How does this affect the guessing number of the graph? How does it affect the set of solutions?

The network in figure 2 corresponds to a type of games that can be described as follows:

- PublicChannelGameVariant(G, s): As before let $G = (V, E)$ be a graph on a vertex set $V = \{1, 2, \ldots, n\}$ of persons. The game is played by n players. Each player is assigned a message selected from some alphabet $\{1, 2, \ldots, s\}$. Each person $w \in \{1, 2, \ldots, n\}$ receive *the function value* (a value in $\{1, 2, \ldots, s\}$) from the set $A_w = \{v \in V : (v, w) \in E\} \subseteq V$. Each player also have access to a public information channel p. How many messages should the public channel p be able to broadcast for all players to be able to deduce there own message? Problem 3 in section II corresponded to the case where G is the complete graph on n nodes.

As we already pointed out there exists graphs G for which the dice guessing game only can achieve maximal probability, if the players uses non-linear functions.

We will show (and this will follow as a corollary of Theorem(16)) that:

Theorem(14)

Assume that the public information is given by a function $p : A^n \to A$. Then PublicChannelGameVariant(K_n, s) is played on the complete graph K_n has a solution if and only if there exists a commutative group $(A, \oplus)$ structure on the alphabet A and there exists n permutations $\pi_1, \pi_2, \ldots, \pi_n \in S_A$ of elements in A such that the public channel broadcast

$$p(x_1, x_2, \ldots, x_n) = \pi_1 x_1 \oplus \pi_2 x_2 \oplus \ldots \oplus \pi_n x_n$$

Roughly, Theorem(14) states that the set of solutions consists of all the "obvious" solutions (where $p(x_1, x_2, \ldots . x_n) = x_1 \oplus x_2 \oplus \ldots \oplus x_n$for a commutative group), together with all "encryptions" $\pi : A \to A$ of these.

X. On the power of Network Coding

In this section we show that the advantage of (linear) Network Coding over any method that does not allows "interference" is as high as one could possible have hoped for. Consider the information networks N in figure 10. The network corresponds to the Guessing Game on the complete graph K_n.

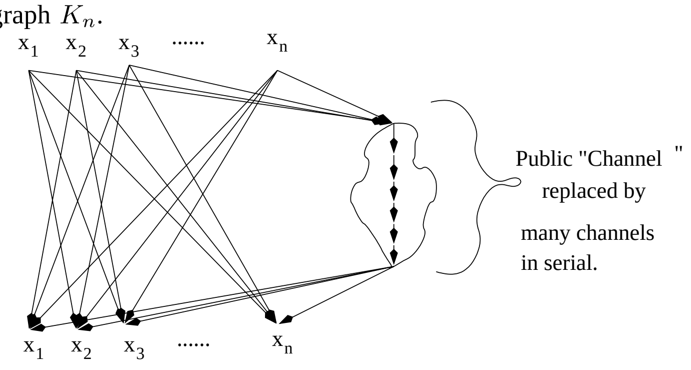

figure 10

Theorem (15)

For each n there is a network N with n input nodes and n output nodes such that the through-put is n times higher than any method that does not allow interference.

For any $n \in N$ and for any $\epsilon > 0$ there exists a network $N(n, \epsilon)$ such that the through-put divided by the number of active channel using Network Coding, is $n - \epsilon$ times as high as the maximal through-put divided by the number of active channels using methods that does not allow interference.

If each inner node is required to have in-degree (and out-degree) ≥ 2 the result remains valid.

Proof: For each $n \geq 2$ (and each $\epsilon > 0$) we base the construction on the network in figure 10. Assume that the public channel consists of m channels in serial. In any "solution" (operating at rate $\frac{1}{n}$) that does not allow mixture of data packets all messages must go through these m channels. Thus the number of active channels is $m+2$. In the Network Coding solution (operating at rate 1) all $n(n-1) + (m+2)$ channels are active. We can choose m such that $n \times (\frac{(m+2)}{n(n-1)+(m+2)}) > n - \epsilon$. For this m the through-put divided by the number of active channel (in the Network Coding solution) is $n-\epsilon$ times as high as the maximal through-put divided by the number of active channels using methods that does not allow interference.

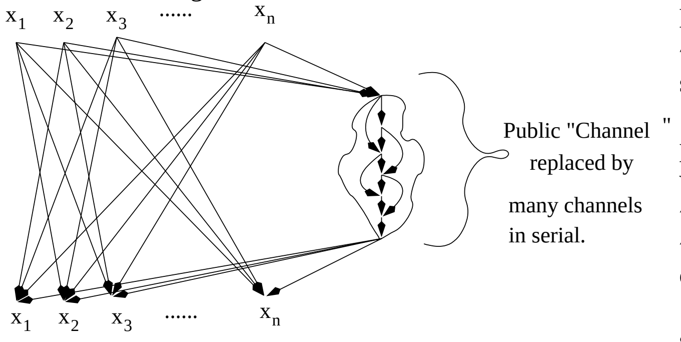

figure 11

The serial construction in this proof might be considered unacceptable. It might be argued that the cost of using the serial channels ought to count as 1 rather than m. To overcome this criticism we can modify the serial channels as indicated in figure 11 and select m so each path through the public channel still must involve $\geq m$ active channels (m chosen as before). .♣

XI. Analysis of specific networks

Consider the information network $N(n)$ sketched in figure 12. Notice that the information network $N(3)$ is displayed in figure 2. Notice also that $N(2)$ essentially is the butterfly network (see figure 1a).

The networks $N(n)$ corresponds to PublicChannelGameVariant(K_n, s) played on the complete graph K_n.

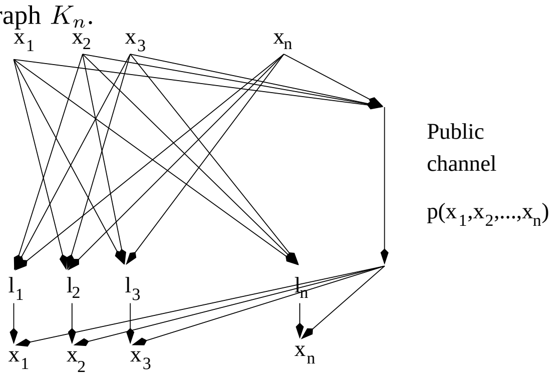

fig. 12

Let us consider this network again. The network $N(3)$ is quite natural to study and I thank Ken Zeger to have pointed out that the very same network was considered in [10] with essentially the same conclusion (Theorem 16). The network $N(3)$ (as well as the networks $N(n)$ in general) can serve as building blocks for various constructions in Network Coding [10].

The three output nodes receive the messages $l_1(x_2, x_3) \in A, l_2(x_1, x_3) \in A$ and $l_3(x_1, x_2) \in A$. Besides this, each output node has access to public message $p = p(x_1, x_2, x_3) \in A$. We notice that a solution to the flow problem associated with N_3 consists of six functions $l_1, l_2, l_3, r_1, r_2, r_3 : A \times A \to A$ as well as one function $p : A \times A \times A \to A$ such that $x_1 = r_1(p(x_1, x_2, x_3), l_1(x_2, x_3))$, $x_2 = r_2(p(x_1, x_2, x_3), l_2(x_1, x_3))$ and $x_3 = r_3(p(x_1, x_2, x_3), l_3(x_1, x_2))$.

The solution we already considered can be achieved (within the framework of linear Network Coding) as follows: Let $(A, \oplus)$ be an abelian group, let $p(x_1, x_2, x_3) := x_1 \oplus x_2 \oplus x_3$, let $l_i(x, y) := x \oplus y$ for $i = 1, 2, 3$ and let $r_i(x, y) := \mathrm{x} \ominus \mathrm{y}$ for $i = 1, 2, 3$. We leave to the reader to check that this defines a solution to the flow problem associated with the network N_3.

Actually, for each abelian group $(A, \oplus)$ and for any three permutations $\pi_1, \pi_2, \pi_3 : A \to A$ the network has a solution with $p(x_1, x_2, x_3) := \pi_1 x_1 \oplus \pi_2 x_2 \oplus \pi_3 x_3$, $l_1(x_2, x_3) := \pi_2 x_2 \oplus \pi_3 x_3$, $l_2(x_1, x_3) := \pi_1 x_1 \oplus \pi_3 x_3$ and $l_3(x_1, x_2) := \pi_1 x_1 \oplus \pi_2 x_2$. We will show that all solutions are essentially of this form. More generally let N_n denote the network:

The network N_n has n input nodes. These transmit messages $x_1, x_2, \ldots x_n \in A$. The messages $x_1, x_2, \ldots x_n$ are independent so we assume that the network cannot exploit hidden coherence in the data. The network N_n has n internal nodes $l_1, l_2, \ldots l_n$. The node l_j is connected to each input node *except* the node that transmits message x_j. The network has n output nodes that are required to receive the messages $x_1, x_2, \ldots x_{n-1}$ and x_n (one message for each output node). The node required to receive x_j is connected to l_j as well as to the public channel p. The public channel broadcasts one message $p = p(x_1, x_2, \ldots x_n) \in A$ to all output nodes. First we notice that:

Observation

The network N_n has a solution over any (finite) alphabet A. Using routing only one message can be transmitted at a time. Thus the through-put using Network coding is n-times as large as the through-put using any type of routing method that does not allow interference. This is optimal since any network problem with n input nodes that is solvable using network coding can be solved using routing if the bandwidth is increased by a factor n.

The next Theorem gives a complete classification of the set of solutions (all utilising Network coding) to the network N_n.

Theorem(16)

Consider the network flow problem N_n over a finite alphabet A. Assume $n \geq 3$. Let $p : A^n \to A$ be any function. The network flow problem N_n has a solution with public information p if and only if for some group composition $\oplus$ on A that makes $(A, \oplus)$ an abelian group, there exist n permutations $\pi_1, \pi_2, \ldots \pi_n : A \to A$ such that $p(x_1, x_2, \ldots x_n) = \oplus_{j=1}^{n} \pi_j x_j$.

Proof: In general if Theorem(16) have been shown for N_r for some $r \geq 3$ the Theorem is also valid for each N_s with $s \geq r$. Thus to prove the theorem it suffice to show that the theorem is valid for N_3.

Let $p : A^3 \to A$ be defined by $p(x_1, x_2, x_3)$. Assume that the network has a solution when the public signal is given by p. The function $p : A^3 \to A$ must be 'latin' (i.e. $f_{a,b}(z) := p(a, b, z)$, $g_{a,c}(y) := p(a, y, c)$ and $h_{b,c}(x) := p(x, b, c)$ for each $a, b, c \in A$ define bijections $f_{a,b}, g_{a,c}, h_{b,c} : A \to A$). Notice that p defines a latin cube of order $|A|$. The functions $l_1.l_2, l_3 : A^2 \to A$ are also forced to be latin i.e. they define three latin squares each of order $|A|$. In order to proceed we need to prove a number of lemmas.

Lemma(17)

Denote one element in A by 1. The network N_3 has a solution for some functions $l_1, l_2, l_3 : A^2 \to A$ if and only the network N_3 has a solution when $l_1(x_2, x_3) := p(1, x_2, x_3)$, $l_2(x_1, x_3) := p(x_1, 1, x_3)$ and $l_3(x_1, x_2) := p(x_1, x_2, 1)$.

Proof of lemma(17): We introduce a new and interesting type of argument that might be useful when reasoning about 'latin' network flow in general. For each output node we draw a triangle with a coding function assigned to each corner. The triangle corresponding to the output node that required output x_1 has assigned $p(x_1, x_2, x_3), l_1(x_2, x_3)$ and x_1 to its corners. If p and l_1 are functions that produce a solution to the network flow problem, $x_1 \in A$ can uniquely be calculated from $p(x_1, x_2, x_3) \in A$ and $l_1(x_2, x_3) \in A$ (i.e. there exists a (latin) function $f : A^2 \to A$ such that $x_1 = f(p(x_1, x_2, x_3), l_1(x_2, x_3)))$. Notice, that any coding function assigned to one of the corners can be calculated uniquely from the two other functions. More specifically $l_1(x_2, x_3) \in A$ is uniquely determined by $x_1 \in A$ and $p(x_1, x_2, x_3) \in A$. And the value $p(x_1, x_2, x_3)$ is uniquely determined by x_1 and $l_1(x_2, x_3)$. We say that a triangle with a coding function assigned to each corner is 'latin' if each of the three coding functions can be calculated from the two other functions. For any solution of the network flow problem N_3 each of the following three triangles are latin:

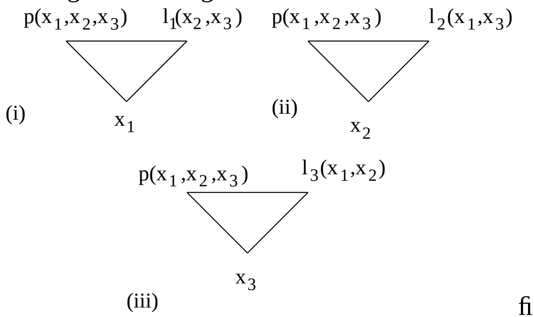

figure 13

Letting $x_1 = 1$ in triangle (i) we notice that $p(1, x_2, x_3)$ can be calculated from $l_1(x_2, x_3)$ and conversely we notice that $l_1(x_2, x_3)$ can be calculated from $p(1, x_2, x_3)$. Thus we can replace the function $l_1(x_2, x_3)$ with the function $l_1(x_2, x_3) := p(1, x_2, x_3)$. Similarly, by letting $x_2 = 1$ in triangle (ii) and letting $x_3 = 1$ in triangle (iii) we obtain a solution with $l_2(x_1, x_3) := p(x_1, 1, x_3)$ and $l_3(x_1, x_2) := p(x_1, x_2, 1)$. This completes the proof of lemma(3).

Lemma(18)

Assume that there is a solution to the flow problem N_3 with public information given by $p : A^3 \to A$. Then the latin function $p(x_1, x_2, x_3)$ determines (uniquely) two latin functions (i.e two latin squares) $l : A^2 \to A$ (l stands for 'left') and $r : A^2 \to A$ (r stands for 'right') defined by the two equations:

- $p(1, l(x_1, x_2), x_3) = p(x_1, x_2, x_3)$
- $p(x_1, r(x_2, x_3), 1) = p(x_1, x_2, x_3)$

Proof of lemma(18): Certainly (since p is latin), there exist uniquely defined functions $l', r' : A^3 \to A$ such that $p(1, l'(x_1, x_2, x_3), x_3) = p(x_1, x_2, x_3)$ and $p(x_1, r'(x_1, x_2, x_3), 1) = p(x_1, x_2, x_3)$. To show lemma (4) it suffices to show that l' is independent of x_3 and that r' is independent of x_1. Consider the two latin triangles:

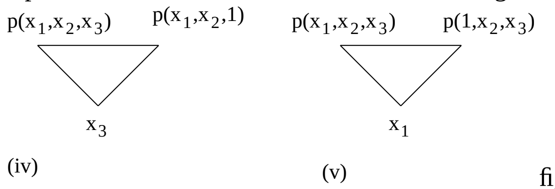

figure 14

In each triangle (iv) and (v) each coding function is uniquely determined by the two other coding functions in the triangle. Thus there exists $f, g : A^2 \to A$ such that $p(x_1, x_2, x_3) = f(p(x_1, x_2, 1), x_3)$ and such that $p(x_1, x_2, x_3) = g(x_1, p(1, x_2, x_3))$. Let $l(x_1, x_2) := l'(x_1, x_2, 1)$ and let $r(x_2, x_3) := r'(1, x_2, x_3)$ and notice that $p(x_1, x_2, 1) = p(1, l(x_1, x_2), 1)$ and $p(1, x_2, x_3) = p(1, r(x_2, x_3), 1)$. But then $p(x_1, x_2, x_3) = f(p(x_1, x_2, 1), x_3) = f(p(1, l(x_1.x_2), 1), x_3) = p(1, l(x_1, x_2), x_3)$ and $p(x_1, x_2, x_3) = g(x_1, p(1, x_2, x_3)) = g(x_1, p(1, r(x_2, x_3), 1) = p(x_1, r(x_2, x_3), 1)$. Thus l and r satisfies the same equations that uniquely determined l' and r' and thus $l'(x_1, x_2, x_3) = l(x_1, x_2)$ and $r'(x_1, x_2, x_3) = r(x_2, x_3)$. This completes the proof of lemma(4).

Lemma(19)

Assume that $p : A^3 \to A$ has a solution and that $p(x_1, x_2, x_3) = p(1, l(x_1, x_2), x_3)$ and assume that $p(x_1, x_2, x_3) = p(x_1, r(x_2, x_3), 1)$. Then the functions $l, r : A^2 \to A$ satisfy the equation $r(l(x_1, x_2), x_3) = l(x_1, r(x_2, x_3))$.

Proof: Since p is latin and $p(x_1, x_2, x_3) = p(1, r(l(x_1, x_2), x_3), 1) = p(1, l(x_1, r(x_2, x_3)), 1)$.

The next three lemma are straight forward to prove.

Lemma(20)

Assume $p(x_1, x_2, x_3)$ allows a solution and that $l(x_1, x_2)$ and $r(x_2, x_3)$ are defined such that $p(1, l(x_1, x_2), x_3) = p(x_1, x_2, x_3)$ and $p(x_1, r(x_2, x_3), 1) = p(x_1, x_2, x_3)$. Then for each pair $\pi_1, \pi_3 : A \to A$ of permutations $p'(x_1, x_2, x_3) := p(\pi_1 x_1, x_2, \pi_3 x_3)$ allows a solution and $l'(x_1, x_2) = l(\pi_1 x_1, x_2)$ and $r'(x_2, x_3) = r(x_2, \pi_3 x_3)$ satisfies the equations $p'(1, l'(x_1, x_2), x_3) = p'(x_1, x_2, x_3)$ and $p'(x_1, r'(x_2, x_3), 1) = p'(x_1, x_2, x_3)$.

Lemma(21)

There exists permutations $\pi_1, \pi_3 : A \to A$ such that $l(\pi_1 x_1, 1) = x_1$ and such that $r(1, \pi_3 x_3) = x_3$.

Lemma(22)

If $p(x_1, x_2, x_3)$ is a solution, there is another solution $p'(x_1, x_2, x_3) = p(\pi_1 x_1, x_2, \pi_3 x_3)$ such that the two functions $l'(x_1, x_2)$ and $r'(x_2, x_3)$ that satisfy the equations $p'(1, l'(x_1, x_2), x_3) = p'(x_1, x_2, x_3)$, $p'(x_1, r'(x_2, x_3), 1) = p'(x_1, x_2, x_3)$ as well as $l'(x_1, 1) = x_1$ and $r'(1, x_3) = x_3$.

Without loss of generality (possibly after having replaced x_1 and x_3 by $\pi_1 x_1$ and $\pi_3 x_3$) we can assume that we are given a latin function $p(x_1, x_2, x_3)$ and two latin functions $l(x_1,x_2)$ and $r(x_2,x_3)$ that satisfies $l(x_1,1)=x_1$, $r(1,x_3)=x_3$, and have $l(x_1,r(x_2,x_3))=r(l(x_1,x_2),x_3)$ for all $x_1,x_2,x_3\in A$. But, then $r(x_1,x_3)=r(l(x_1,1),x_3)=l(x_1,r(1,x_3))=l(x_1,x_3)$ and thus $l=r$. But then l is transitive i.e. $l(x_1,l(x_2,x_3))=l(l(x_1,x_2),x_3)$. Furthermore since $l(x,1)=x$ and $l(1,x)=r(l,x)=x$ we notice that l defines a group operation on A. Thus we have shown that for any function $p(x_1,x_2,x_3)$ that allows a solution to the network flow problem N_3, there exist permutations $\pi_1,\pi_3:A\to A$ such that if we let $p'(x_1,x_2,x_3):=p(\pi_1x_1,x_2,\pi_3x_3)$ then there is a group structure $*$ on A such that $p'(x_1,x_2,x_3)=p'(1,x_1*x_2*x_3,1)$ for all x_1,x_2,x_3. But then there is a permutation $\pi:A\to A$ such that if we let $p''(x_1,x_2,x_3)=\pi(p'(x_1,x_2,x_3))$ then $p''(1,b,1)=b$ for all $b\in A$. Notice, that $p''(x_1,x_2,x_3)=\pi(p'(x_1,x_2,x_3))=\pi(p'(1,x_1*x_2*x_3,1))=p''(1,x_1*x_2*x_3,1)=x_1*x_2*x_3$. This shows:

Lemma(23)

Let $p:A^3\to A$ be the public information in the network N_3. Then, if there is a solution to the network flow problem N_3, there exists a group composition $*$ on A such that ‘essentially’ $p(x_1,x_2,x_3)=x_1*x_2*x_3$ (modulo the application of suitable permutations to x_1,x_3 and p (or x_2)).

Lemma(24)

Let $(A,*)$ be a group and let $p(x_1,x_2,x_3):=x_1*x_2*x_3$. Then the flow problem N_3 has a solution if and only if $(A,*)$ is a commutative group.

Proof: Assume that $p(x_1,x_2,x_3):=x_1*x_2*x_3$ (or just $x_1x_2x_3$ for short) allows a solution. Then we have the following ‘derivation’ from latin triangle with coding functions $p(a,b,c)=abc$, $p(a,1,c)=ac$ and b.

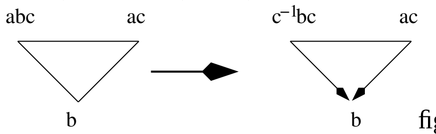

figure 15

Figure 7, represents the fact that b can be uniquely determined from abc and ac. But, then given $c^{-1}bc$ and ac we can calculate $abc=(ac)c^{-1}bc$ and thus we can determine b. Now ac can take any value (depending on a) and thus this equation is useless in calculating b. This shows that b is uniquely determined from $c^{-1}bc$. The expression $c^{-1}bc$ must be independent of c and thus $c^{-1}bc=1^{-1}b1=b$. But, then $bc=cb$ for all $a,b,c\in A$ which shows that the group $(A,*)$ must be a commutative group. The converse is rather obvious, since if $(A,*)$ is an abelian group and $p(x_1,x_2,x_3)=x_1x_2x_3$, we get a solution by letting $l_1(x_1,x_2):=x_1x_2$, $l_2(x_1,x_3)=x_1x_3$ and $l_3(x_1,x_2)=x_1x_2$. This completes the proof of lemma(10) which in turn clearly implies the theorem for N_3. This in turn easily implies the validity of theorem(2) for general N_n with $n\ge 3$. ♣

A. Constructions using the networks $N(n)$ as building blocks

A very natural construction is seen in figure 16. The network $M(3)$ appear by adding a new node to $N(3)$ that requires x_3 and receive its input from l_1,l_2 and l_3. The idea behind the construction of $M(3)$ is to ensure that the network only has solutions for certain alphabet sizes.

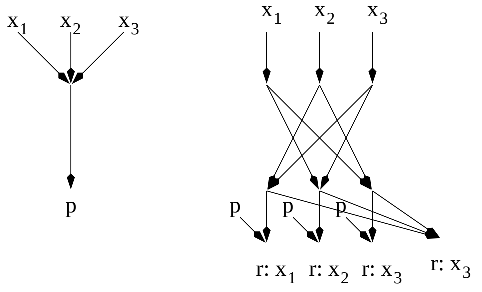

figure 16

The new node is added to $N(3)$ to ensure that the node only can reconstruct x_3 by essentially computing x_3 from $x_3\oplus x_3$. Heuristically, node l_1 receives $x_2\oplus x_3$, node l_2 receives $x_1\oplus x_3$ and l_3 receives $x_1\oplus x_2$. From these three messages the new node needs to derive the message x_3. The natural way to do this is to calculate $2x_3=(x_1\oplus x_3)\oplus(x_1\oplus x_3)\ominus(x_1\oplus x_2)$. Now, intuitively, whis is only possible if the abelian group $(A,\oplus)$ has the linear map $l:A\to A$ given by $l(a)=2a$ being invertible. This holds if and only if $|A|$ is odd (for a slightly more detailed explanation of this see [10]).

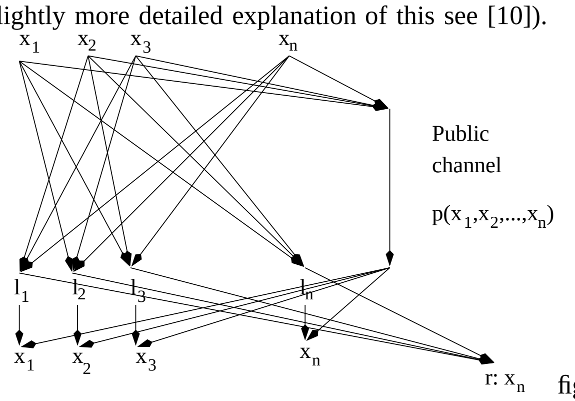

fig. 17

Next, consider the network $M(n)$ in figure 17. Again, heuristically node l_j receive $p\ominus x_j$ and thus the canonical way of calculating x_n is to calculate $(n-1)x_n=(p\ominus x_1)\oplus(p\ominus x_2)\oplus\ldots(p-\ominus x_{n-1})\ominus(n-2)(p\ominus x_n)$. This calculations can always be carried out if the map $l:A\to A$ given by $l(a)=(n-1)a$ is invertible. Intuitively, if there is no such invertible map l we would (keeping Theorem 16 in mind) not expect that there are any solutions to the corresponding information flow problem. We will now prove this more formally:

Lemma(25)

The information flow problem $M(n)$ has a solution (that might utilise network coding) over alphabet A of size s if and only if the linear map $l:A\to A$ given by $l(a)=(n-1)a$ is invertible for some abelian group $(A,\oplus)$ of size s.

Proof: According to Theorem(16) the public channel broadcast the message $p=\pi_1(x_1)\oplus\pi_2(x_2)\oplus\ldots\oplus\pi_n(x_n)$. Without loss of generality we can assume that the public information is given by $p=x_1\oplus x_2\oplus\ldots\oplus x_n$ (since we can replace the input variables x_j with $x_j':=\pi_j^{-1}(x_j)$ and then the output requirement x_j can be achieved if and only if x_j' can be achieved). First notice that x_j,l_j and p form a latin triangle in any solution. Since $x_j,p\ominus x_j$ and p also forms a latin triangle and since l_j and $p\ominus x_j$ are independent of x_j we notice that l_j and $p\ominus x_j$ can be computed from each other. Thus, any solution to the information flow problem $M(n)$ can without loss of generality be assumed to have $l_j := p \ominus x_j$ for $j = 1, 2, \ldots, n$.

First, assume that l is not invertible over the group $(A, \oplus)$. Then, exists $a \in A$ with $a \neq 0$ with $(n-1)a = 0$. Consider, the input messages $x_1 = x_2 = \ldots = x_n = a$ and compare them with the input messages $x_1 = x_2 = \ldots = x_n = 0$. In the both cases $l_1 = l_2 = \ldots = l_n = 0$ and thus, since $a \neq 0$, the new added node is in general not able to recover x_n (since it cannot in general distinguish a from 0). This argument is essentially, identical to an argument in [10].

Conversely, assume l is invertible for some ablean group $(A, \oplus)$ of size s. Let $p = x_1 \oplus x_2 \oplus \ldots \oplus x_n$ and let $l_j = p \ominus x_j$ for $j = 1, 2, \ldots, n$. Notice that $(n-1)x_n = (p \ominus x_1) \oplus (p \ominus x_2) \oplus \ldots \oplus (p \ominus x_{n-1}) \ominus (n-2)(p \ominus x_n)$. If l is invertible $x_n = l^{-1}((n-1)x_n)$ can be recovered.

♣

From this we get:

Theorem(26)

The information network $M(n)$ (for $n \geq 3$) has a solution over an alphabet A if and only if $n-1$ and $|A|$ are relative prime (i.e. $gcd(n-1, |A|) = 1$).

B. A neat application of Theorem 16

Consider the information network L in figure 18. This network is related (but non-isomorphic to) the network $N1$ constructed in [8].

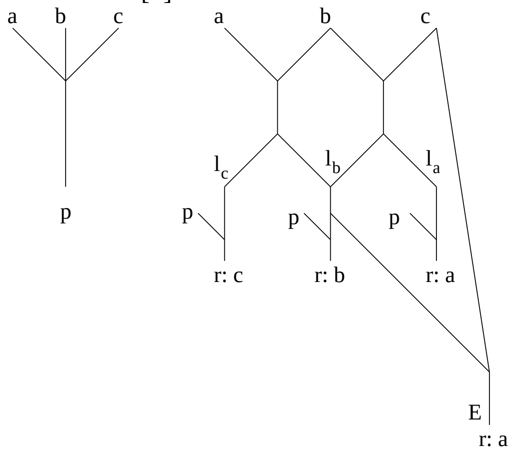

figure 18

We show:

Theorem(27)

The information flow problem L has a solution over an alphabet A if and only $|A|$ is a power of two (i.e.$|A| = 2^k$ for some $k \in N$).

Proof: First notice that node E that requires a has no access to the public channel. Thus the message passing through l_b cannot depend on message b. But then node l_c receives a message that is a function of a and b, l_b receives a message that is a function of a and c, while node l_a receives a message that is a function of b and c. Thus, we can apply Theorem 16, and deduce that there exists an abelian group structure $(A, \oplus)$ on A such that the public channel broadcast a message on the form $p = \pi_a(a) \oplus \pi_b(b) \oplus \pi_c(c)$ where π_a, π_b and π_c are permutations (encryptions!!) of the messages a, b and c.

Without loss of generality we can assume $p = a \oplus b \oplus c$ since otherwise we can simply "encrypt" the messages a, b and c with $a' := (\pi_a)^{-1}(a), b' := (\pi_b)^{-1}(b)$ and $c' := (\pi_c)^{-1}(c)$ and then "decrypt" the messages a, b and c in the receiver nodes. This point needs a bit care since the decryption might not be a linear function (see comment belov).

Anyway first assume that $l_c = a \oplus b$ and $l_b = a \oplus c$ and $l_a = b \oplus c$. Thus $l_b = a \oplus c$ can be derived from $l_c = a \oplus b$ and $l_a = b \oplus c$.

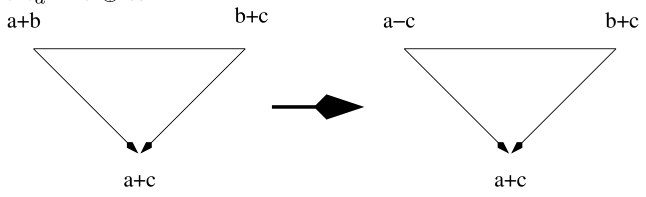

figure 19

As indicated in figure 19, we then infer that $a \oplus c$ always can be derived from $a \ominus c$ and $b \oplus c$. But the message $b + c$ depend on (is "infected" by) b and so we conclude that $a \oplus c$ always can be derived from $a \ominus c$ alone. But then there exists $f : A \to A$ such that $a \oplus c = f(a \ominus c)$ (Had we only considered encoded signals we would have reached the same conclusion since a suitable decoding could be build into the definition of f).

If we let $c = 0$ we notice that $a = f(a)$ and so f is the identity map. Thus $a \oplus c = a \ominus c$ i.e. $2c = 0$. From this we conclude that the only possibly solutions can appear if $(A, \oplus)$ is an abelean group for which all elements have order 2. This is only possible if $|A| = 2^k$ for some k.

On the other hand for any alphabet A with 2^k elements the network L has a (linear) solution given by $l_c = a \oplus b, l_b = a \oplus c, l_a = b \oplus c$ and $p = a \oplus b \oplus c$. ♣

XII. Multiple unicast Networks

I will finish the paper by introducing a simple construction that allows us to convert the results in the previous section to similar results for multiple unicast networks.

Consider the information network C in figure 20 (i). The source nodes send messages $x_1, x_2, \ldots, x_n$ to a collection of receiver nodes. Assume two receiver nodes requires x_1. To be a multiple-unicast network we require that different receiver nodes require different messages.

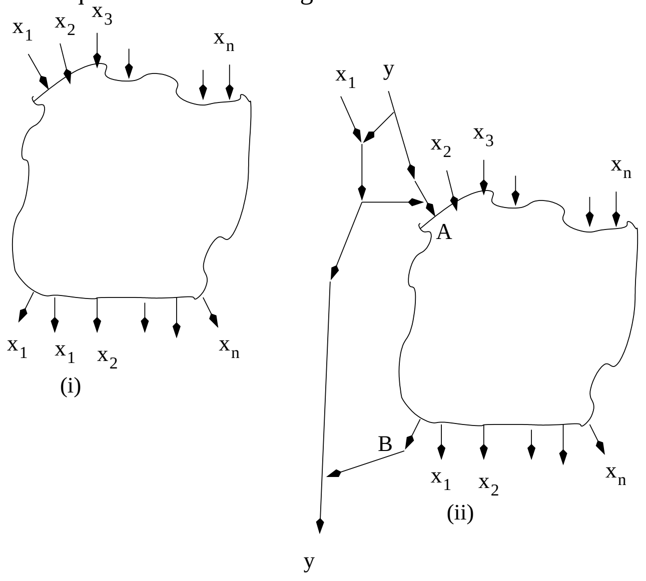

figure 20

In figure 20 (ii) we have modified the information network in figure 20 (i). New source and receiver nodes sending and receiving y have been added. Notice that any solution must have message x_1 enter at node A and have message x_1 leaving the network at point B. This ensures that the network C' in figure 20 (ii) has a solution over an alphabet A if and only if C has a solution over A. Actually,we notice (using the analysis of the butterfly network in [19]) that the set of solutions to the information flow problem C' is a direct product of the set of latin squares of order $|A|$ and the set of solutions to C. Thus, the class of functions needed to solve C is essentially the same as the class of functions needed to solve C'.

Theorem(28)

There exists a multiple unicast information flow problem U , that is solvable, but have no linear solutions (over any vector space).

Proof: In [8] the authors construct an information flow problem N_{nonlin} that is solvable, but have no linear solutions over any alphabet organised as a vector space. Using the idea just introduced we can modify this problem to a multiple unicast problem U that has a set of solutions that is a direct product of the set of latin squares of order $|A|$ and the set of solutions to N_{nonlin}. ♣

Theorem(29)

There exists a directed graph G with guessing number k that can only be achieved if the players uses non-linear guessing strategies.

There exists a directed graph G with guessing number $k_G(s)$ depending on s. Furthermore there exists k such that $k_G(s) < k$ for infinitely many values of s while $k_G(s) = k$ for all remaining values of s (also infinitely many).

Proof: The first part of the theorem follows by combining Theorem 2 and Theorem 28. The second part follows by combining Theorem 2 with Theorem 27 (or Theorem 26) and the general conversion method ♣.

Open Question

Is the guessing number of an undirected graph always an integer?

XIII. Acknowledgements

To end I would like to thank Raymond Yeung, Yun Sun, Taoyang Wu and Ken Zeger for their valuable feedback and general comments on an earlier draft version of this paper.

References

[1] S. Medard M.-Koetter R. Acedanski, S. Deb. How good is random linear coding based distribued network storage? To appear.

[2] A Aggarwal and J S Vitter. The input/output complexity of sorting and related problems. *Communications of the ACM*, 31(9):1116–1126, September 1988.

[3] R Ahlswede, N Cai, Li, and R Yeung. An algebraic approach to network coding. page 104, 2001.

[4] K.R. Bhattad, K. Narayanan. Weakly secure network coding. To appear.

[5] N. Cai and R.W. Yeung. Network coding and error correction. In *ITW 2002 Bangalore*, pages 119–122, 2002.

[6] J. Cannons, R Dougherty, C Freiling, and K Zeger. Network routing capacity. *IEEE/ACM TRANSACTIONS ON NETWORKING*, October 2004. Submitted.

[7] Deb, Choute, Medard, and Koetter. Data harvesting: A random coding approach to rapid dissemination and efficient storage of data. In *INFOCOM*, 2005. Submitted.

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

[9] R Dougherty, C Freiling, and K Zeger. Linearity and solvability in multicast networks. *IEEE Transactions on Information Theory*, 50(10):2243–2256, October 2004.

[10] R Dougherty, C Freiling, and K Zeger. Unachievability of network coding capacity. *IEEE Transactions on Information Theory and IEEE/ACM Transactions on Networking (joint issue)*, 2005. submitted March 2005.

[11] C. Fragouli and E Soljanin. A connection between network coding and convolutional codes. In *IEEE International Conference on Communications*, 2004.

[12] T Ho, M Medard, and R Koetter. An information theoretic view of network management. In *Prooceeding of the 2003 IEEE Infocom*.

[13] R Koetter and M Medard. An algebraic approach to network coding. In *Proocedings of the 2001 IEEE International Symposium on Information Theory*.

[14] R Koetter and M Medard. Beyond routing: An algebraic approach to network coding. In *Proceedings of the 2002 IEEE Infocom*, 2002.

[15] Li, Yeung, and Cai. Linear network codes. *IEEE Trans. v.49,371-381*.

[16] K. Rabaey J Petrovic, D. Ramchandran. Overcomming untuned radios in wireless networks with network coding. To appear.

[17] L.G. Pippenger, N. Valiant. Shifting graphs and their applications. *JACM*, 23:423–432, 1976.

[18] S. Riis. Linear versus non-linear boolean functions in network flow. In *Proceeding of CISS 2004*.

[19] S. Riis and R Ahlswede. Problems in network coding and error correcting codes. NetCod 2005.

[20] M Thorup and S Riis. Personal communication. March 1997.

[21] L. Valiant. Graph-theoretic arguments in low-level complexity.

[22] L Valiant. On non-linear lower bounds in computational complexity. In *Proc. 7th ACM Symp. on Theory of Computing*, pages 45–53, 1975.

[23] L. Valiant. Why is boolean circuit complexity theory difficult? In M.S. Pattorson, editor, *Springer Lecture Series*, pages 84–94, 1992.

[24] C. Boudec J-Y. Widmer, J. Fragouli. Low-complexity energy-efficient broadcasting in wireless ad-hoc networks using network coding. To appear.

[25] Wu, Chou, and Kung. Information exchange in wireless networks with network coding and physical-layer broadcast. Technical Report MSR-TR-2004-78, Microsoft Technical Report, Aug. 2004.

[26] Yeung and Zhang. Distributed source coding for satellite communications. *IEEE Trans. Inform. Theory*, (IT-45):1111–1120, 1999.