Universidad Carlos iii de Madrid

Escuela Politécnica Superior

Máster en multimedia y comunicaciones

Códigos IRA

Luca Martino

Comunicaciones Digitales

Marzo 2008

Contents

1 Introducción **2**

2 Códigos Bloque Lineales **5**

3 Códigos Convolucionales **6**

4 Transmisión en un Canal Gaussiano **8**

5 Gráfico de Tanner **10**

6 Códigos LDPC **11**

7 Códigos RA **12**

8 Código *Irregular* RA **14**

9 Bibliografía **18**

1 Introducción

Supongamos de querer enviar por un canal binario simétrico sin memoria (fig.1) una secuencia de bits. Por cada bit (0 o 1) enviado, tenemos una probabilidad p que se produzca error. Claramente, desearíamos bajar lo más posible el valor de esta probabilidad.

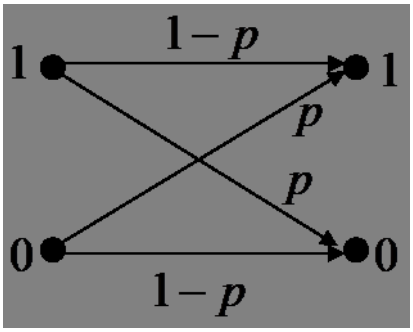

Figure 1: Canal Binario Simétrico.

Códigos de Repetición: la idea más sencilla para proteger de los errores, es repetir varias veces el bits que se quiere enviar, es decir para cada bit se envía una secuencia larga n con el mismo bit repetido (pro ejemplo con $n = 3$, $1 \rightarrow 111$). El decodificador decidirá que el bit transmitido será lo que más aparece el la secuencia recibida (si recibe 101 supondrá que el bit transmitido era 1).
Podemos sacar una observación muy importante, ya mirando los códigos de repetición: se puede lograr una mayor *fiabilidad* en la comunicación, pero el precio que hay que pagar es en términos de *velocidad* (en el ejemplo anterior, hay utilizar 3 veces el canal para enviar solo un bit de información).
En general, la tarea de un codificador de canal consiste en *elegir* 2^k secuencias de n bits (fig.2), llamadas palabras códigos, de tal forma que los errores introducidos por el canal produzcan el menor numero de errores en decodificación.
Para conseguirlo, las 2^k palabras código deben diferenciarse en el mayor número posible de bits; en esta manera, se minimiza la probabilidad que una palabra código se confunda con otra. La *distancia de Hamming*, es propio la cantidad que nos proporciona de cuanto bits difieren dos secuencias.

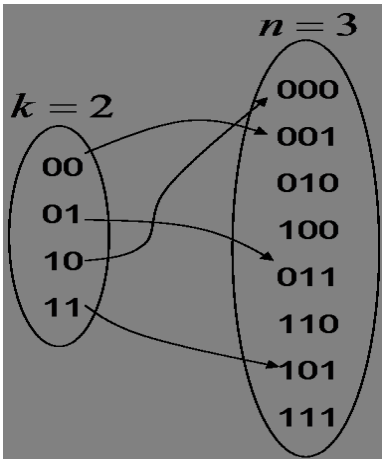

Figure 2: Ejemplo codificador.

Llamaremos *tasa* R de un código la proporción:

$$R = \frac{k}{n} \tag{1}$$

Donde al numerador aparece el número de *bits de información/mensaje* k y al denominador el número totales de n de bits enviados ($n - k$ serán los *bits de redundancia/chequeo*). Esta cantidad R, siempre menor que 1, puede interpretarse como *velocidad de transmisión* por uso del canal. En caso de un alfabeto binario y de un canal sin memoria, nos encontraríamos en el canal esquematizado en figura (1). La *capacidad* de un canal binario C_b (fig.1) vale $C_b = 1 - H_b(p)$, donde $H_b(p)$ es la entropia de una fuente binaria. Para canales binarios la *tasa máxima* a la que se puede transmitir en manera que la probabilidad de error a la salida sea tan pequeña como se quiera, es propio la capacidad C_b. En otras palabras, el **teorema de codificación de canal** (aplicado a un canal binario sin memoria) afirma que si $R \leq C_b$, es decir si la tasa del código es menor que la capacidad, es posible lograr una comunicación fiable: la probabilidad de error tiende a cero, si aumenta la longitud del bloque de datos trasmitidos (compuesto por bits de información más las redundancias). En general, este teorema proporciona un limite máximo C para la velocidad de transmisión (hay que repetir un cierto numero de veces la información para garantizar que venga recibida correctamente).
Dado un bloque de n bits, k mensaje y $n - k$ redundancia, la demostración de teorema de codificación de canal consiste en hacer tender a infinito la

longitud del bloque $n \to \infty$, manteniendo constante la tasa $R = k/n$ del código. De allí, *se puede ver que si $R \leq C_b$ es posible reducir la probabilidad de error aumentando la longitud del bloque n.*
Codificador Aleatorio: propio la demostración del teorema de codificación de canal se basa en la elección aleatoria de 2^k palabras códigos de longitud n (fig.2). Como hemos dicho, si $R \leq C_b$, aumentando n y con R constante la probabilidad de error tiende a cero. Surge la pregunta si este esquema puede haber interés practico. Por mala suerte, este tipo de códigos tiene una complejidad creciente exponencial con k sea en codifica que en decodifica: de hecho para transmitir hay que almacenar 2^k palabras código, y para la detección el único método posible es comparar la secuencia recibida con todas las posibles.

Idealmente un código de protecciones contra errores debe tener las propiedades siguientes:

- complejidad *lineal* en codificar.
- complejidad *lineal* en decodificar.
- probabilidad de error mas baja posible ($P_e \to 0$) para una determinada tasa R, y que tienda a cero por grandes bloques ($n \to \infty$).
- una tasa R más cercana posible a C_b; es decir una tasa R más alta posible menor del limite máximo permitido C_b, para obtener una P_e baja en recepción.

Con un *código de repetición* podemos lograr P_e muy baja aumentando n, pero tenemos una tasa $R = 1/n$ también decreciente con n (para enviar un bits de mensaje tardaríamos mucho). Se han pensado otras ideas para mejorar está situación.
En realidad, se podrían encontrar código aleatorios con complejidad lineal en codificación, con una tasa R igual a la capacidad de canal, con probabilidad de error P_e decreciente con la longitud de bloque n. El problema que la complejidad del decodificador sigue siendo alta (hay que comparar siempre con todas las palabras códigos posibles). Así que el utilizo de un código aleatorio, es posible solo por k y n pequeños (suponiendo $R = k/n$ constante), comportando una P_e alta.

Decodificador óptimo: se puede demostrar que el mejor decodificador elije como palabra código transmitida aquella que difiere el menor número de bits de la secuencia recibida.

Así que idealmente nos gustaría tener un decodificador que proporcione el mismo resultado del decodificador óptimo, pero con una carga computacional que crezca *linealmente* con el numero de bits de información k. Realmente, el los últimos años se han encontrado muchos esquemas de codificación con decodifica con carga computacional *lineal* pero *no óptima*. Una excepción son los *código bloque perfectos* y los *códigos convolucionales* donde efectivamente el algoritmo de decodifica proporciona la decodificación es óptima.
Si cada k bits de información vienen codificados de forma independientes, hablaremos de *códigos bloque*. No siempre es así, por ejemplo en los códigos convolucionales la codificación se realiza sobre un flujo continuo de bits de mensaje. Un código se dice *sistemático* si los bits de mensaje son parte de la palabra código, separados de los bits de redundancia.

2 Códigos Bloque Lineales

La característica principal es que cada palabra código se puede expresar como combinación lineal de k palabras base $\vec{c}_j$, donde los coeficientes son 0 y 1. En formula:

$$\vec{c} = a_1\vec{c}_1 + a_2\vec{c}_2 + \ldots + a_k\vec{c}_k \tag{2}$$

Definiendo una matriz generadora G binaria $k \times n$, las palabras código resultan:

$$\vec{c} = \vec{a} \cdot G \tag{3}$$

La decodifica utiliza *la matriz de paridad* H $(n-k) \times n$, que cumple la relación:

$$G \cdot H^T = 0 \tag{4}$$

Esta matriz tiene interés porque, si llamamos con $\vec{r} = \vec{c} + \vec{e}$ la secuencia recibida contaminada por el ruido $\vec{e}$, el vector $\vec{s}$ llamado *síndrome*:

$$\vec{s} = \vec{r} \cdot H^T \tag{5}$$

se anulará si la secuencia recibida ($\vec{r} = \vec{c} = \vec{a} \cdot G$) es exactamente una palabra código:

$$\vec{s} = \vec{r} \cdot H^T = \vec{a} \cdot G \cdot H^T = \vec{0} \tag{6}$$

De esta manera, si el síndrome $\vec{s}$ no se anula, se podrá afirma que se ha producido algún error y a lo mejor detectar en que bit se ha producido.

3 Códigos Convolucionales

Se diferencian de los códigos bloque en su forma estructural y las propiedades para corregir errores. Los códigos convolucionales son *lineales*, es decir una combinación de dos palabras código cualquiera es también una palabra código. El sistema tiene *memoria*: la codificación actual depende de los datos pasados.
Un código convolucional queda especificado por tres parámetros: el número de salidas n, k entradas y m memoria del código. El número de salidas y de entradas tienen la misma tarea respectivamente que la longitud del bloque y del numero de bits de información, definiendo la tasa del código $R = k/n$. El funcionamiento del convolucional está basado en m registros

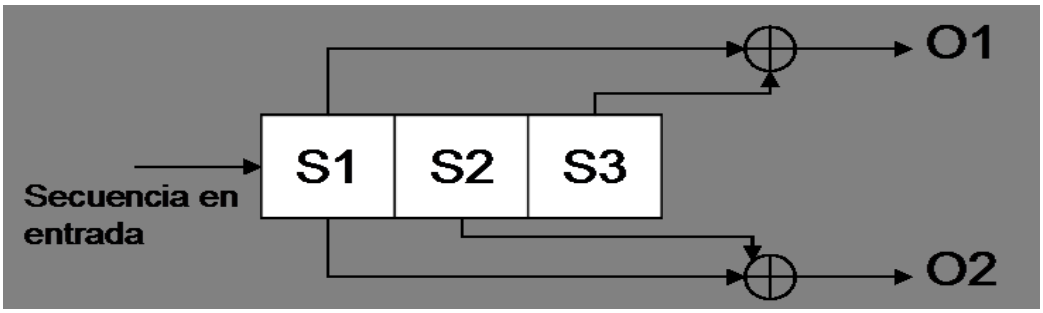

Figure 3: Registros de Desplazamiento de un Convolucional(2,1,3).

de desplazamiento ("memoria" de capacidad un bit) y n sumas módulo dos. La codificación de los bits se realiza a partir del valor del bit presente a la entrada y los valores de los m bits anteriores que están guardados en los registros.
La descripción de un código convolucional puede hacerse de varias maneras: conexión de vectores o polinomios, diagrama de estado, diagrama de árbol y

diagrama Trellis son los esquemas más utilizados. En figura (6) se muestra el esquema en registros de desplazamiento de un código convolucional, con un tasa de $R = 1/2$ y $m = 3$ registros de memorias.
Como ejemplo del funcionamiento de este codificador, supongamos que se quiere enviar la secuencia 0101 donde los bits a la derecha son los más antiguos,así que el primer bit en entrada será un 1. Teniendo en cuenta que al principio todos los registros están inicializados a cero $S_i = 0$, entonces introduciendo el primer bit:

$$S_1 = 1, S_2 = 0, S_3 = 0 \rightarrow O_1 = 1, O_2 = 1 \tag{7}$$

Introduciendo el segundo, el tercero y el cuarto:

$$S_1 = 0, S_2 = 1, S_3 = 0 \rightarrow O_1 = 0, O_2 = 1 \tag{8}$$
$$S_1 = 1, S_2 = 0, S_3 = 1 \rightarrow O_1 = 0, O_2 = 1 \tag{9}$$
$$S_1 = 0, S_2 = 1, S_3 = 0 \rightarrow O_1 = 0, O_2 = 1 \tag{10}$$

Al final del proceso la secuencia 0101 será codificada con la secuencia 01010011. Se puede crear también una tabla que asocia a cada secuencia de m bits dentro de los registros, una secuencia de n bit de salida. Prácticamente **se convoluciona la secuencia de bits de mensaje con un filtro FIR binario** (la redundancia se obtienen filtrando). La representación más utíl para

S1,S2,S3	01,02
0,0,0	0,0
0,0,1	1,0
0,1,0	0,1
0,1,1	1,1
1,0,0	1,1
1,0,1	0,1
1,1,0	1,0
1,1,1	0,0

Figure 4: Relación entradas-salidas.

la decodificación es el diagrama de Trellis: es un diagrama a forma de red donde a cada línea vertical se encuentran los estados del codificador. Para realizar la decodificación se utiliza el famoso *Algoritmo de Viterbi*, que busca el camino de mínima distancia dentro del diagrama Trellis.

4 Transmisión en un Canal Gaussiano

En las secciones siguientes, supondremos que las palabras códigos sean trasmitida en un canal gaussiano. Indicamos con $\vec{c}$ un vector binario que, antes de ser enviado, será transformado en el vector de señal $\vec{t}$ que asume el valor de tensión a si $c_k = 1$ y $-a$ si $c_k = -1$. La señal recibida será:

$$\vec{r} = \vec{t} + \vec{\eta} \tag{11}$$

Donde el vector $\vec{\eta}$ está compuesto por muestras independientes de ruido gaussiano de media cero y varianza $\sigma^2 = N_0/2$. Dado el vector recibido $\vec{r}$, nos gustaría conocer la probabilidad *a posteriori* de detección:

$$p(c_n|r_n) = \frac{p(r_n|c_n) \cdot p(c_n)}{p(r_n)} = \frac{p(r_n|t_n) \cdot p(t_n)}{p(r_n)} \tag{12}$$

La segunda igualdad vale porque entre $\vec{t}$ y $\vec{c}$ hay una relación determinista. Por ejemplo, la probabilidad que $c_n = 1$ habiendo recibido un cierto valor de r_n es:

$$p(c_n = 1|r_n) = \frac{p(r_n|t_n = a) \cdot p(t_n = a)}{p(r_n)} \tag{13}$$

$$p(c_n = 1|r_n) = \frac{p(r_n|t_n = a) \cdot p(t_n = a)}{p(r_n|t_n = a) \cdot p(t_n = a) + p(r_n|t_n = -a) \cdot p(t_n = -a)} \tag{14}$$

Donde se ha utilizado el teorema de la probabilidad total. Suponiendo $p(c_n = 1) = p(c_n = 0) = 1/2$ (lo mismo valdrá por t_n) y desarrollando más:

$$p(c_n = 1|r_n) = \frac{p(r_n|t_n = a)}{p(r_n|t_n = a) + p(r_n|t_n = -a)} \tag{15}$$

Ahora siendo ruido gaussiano, podemos escribir:

$$p(r_n = 1|t_n = a) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(r_n - a)^2\right\} \tag{16}$$

Y sustituyendo en la (15):

$$p(c_n = 1|r_n) = \frac{\exp\left\{-\frac{1}{2\sigma^2}(r_n - a)^2\right\}}{\exp\left\{-\frac{1}{2\sigma^2}(r_n - a)^2\right\} + \exp\left\{-\frac{1}{2\sigma^2}(r_n + a)^2\right\}} \tag{17}$$

dividiendo arriba y abajo por el numerador:

$$p(c_n = 1|r_n) = \frac{1}{1 + \frac{\exp\left\{-\frac{1}{2\sigma^2}(r_n+a)^2\right\}}{\exp\left\{-\frac{1}{2\sigma^2}(r_n-a)^2\right\}}} = \frac{1}{1 + \exp\left\{-\frac{2ar_n}{\sigma^2}\right\}} \tag{18}$$

Vamos a mostrar un ejemplo bastante ilustrativo para mejor entender no solo cuanto escrito, sino también como surgieron unas ideas de decodificación que presentaremos luego. Supongamos que, dado el mensaje $\vec{m} = [1\ 0\ 1\ 0\ 1]$, a través de un determinado método de codificación tengamos la siguiente palabra código $\vec{c} = [0\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 1]$. Si la nuestra constelación tiene amplitud $a = 2$, el vector $\vec{t}$ será:

$$\vec{t} = [-2; -2; -2;\ 2; -2;\ 2; -2;\ 2; -2;\ 2] \tag{19}$$

Al enviar por un canal gaussiano de varianza $\sigma^2 = 2$, recibimos:

$$\vec{r} = [-0.63; -0.83; -0.73; -0.04;\ 0.1;\ 0.95; -0.76;\ 0.66; -0.55;\ 0.58] \tag{20}$$

Ahora utilizando la (18), podemos calcular las probabilidades que el bit enviado sea un 1:

$$p(\vec{c} = 1|\vec{r}) = [0.22;\ 0.16;\ 0.19;\ 0.48;\ 0.55;\ 0.87;\ 0.18;\ 0.79;\ 0.25;\ 0.76] \tag{21}$$

Si tomamos como umbral de decisión el valor de probabilidad 0.5 (si es > 0.5 supongo un 1, si es < 0.5 tomo un 0), podemos tratar de reconstruir el vector $\vec{c}$ enviado:

$$\vec{c}_{est} = [0\ 0\ 0\ \bar{0}\ \bar{1}\ 1\ 0\ 1\ 0\ 1] \tag{22}$$

Donde hemos evidenciado los dos bits, cuarto y quinto, donde se producen los errores. Además se puede notar como los valores de probabilidad a posteriori para estos dos bits (0.48 y 0.55), son muy cercanos al valor del umbral 0.5 así que, se podría decir, que nuestra decisión ha sido más *débil* respecto a los demás bits. Este tipo de indicación (decisión fuerte o débil), será aprovechada por el *Soft Decision Decoder* (lo que hemos hecho es equivalente a tomar como umbral 0 y decidir a través de los signos de los r_n recibidos; pero desde el punto de vista teórico será más útil).

5 Gráfico de Tanner

A cada matriz de paridad $H_{m\times n}$ está asociado un gráfico, llamado *Gráfico de Tanner*, compuesto por 2 conjuntos de nodos. El primer conjunto de n nodos representa los bits c_k de la palabra código (de información y de paridad, juntos). El segundo conjunto de m nodos, llamado *nodos de control* z_i, representa las condiciones de paridad que tendrán que respectar los bits de la palabra código. Por ejemplo, la matriz:

$$H_{3\times 4} = \begin{bmatrix} 1\ 1\ 0\ 1 \\ 0\ 1\ 1\ 1 \\ 1\ 0\ 1\ 0 \end{bmatrix} \tag{23}$$

corresponde a las siguientes condiciones (en modulo 2) que tendrán que cumplir nuestras palabras códigos de 4 bits, serian ($\mathbf{c}\cdot H = \mathbf{0}$):

$$\begin{cases} c_1 + c_2 + c_4 = 0 \\ c_2 + c_3 + c_4 = 0 \\ c_1 + c_3 = 0 \end{cases} \tag{24}$$

Se ve claramente, por ejemplo, que el primer bit c_1 interviene en el nodo de control z_1 y z_3 y que al nodo z_2 llegaran tres ramas provenientes de c_2, c_3, c_4. Gráficamente:

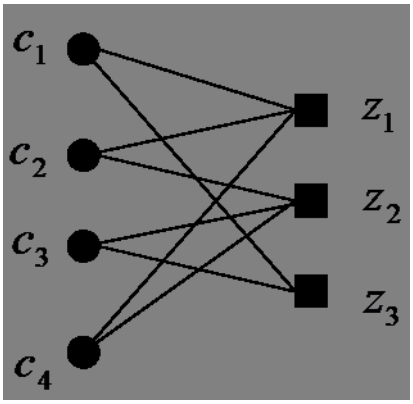

Figure 5: Gráfico de Tanner (*círculos*: bits palabra código; *cuadrados*: nodo de chequeo) .

Además se suelen definir los valores $N(i)$ y $M(j)$ como respectivamente el

conjunto de entradas no nulas de la fila i y de la columna j de H, por el ejemplo anterior:

$N(1) = \{1\ 2\ 4\}$
$N(2) = \{2\ 3\ 4\}$
$N(3) = \{1\ 3\}$
$M(1) = \{1\ 3\}$
$M(2) = \{1\ 2\}$
$M(3) = \{2\ 3\}$
$M(4) = \{1\ 2\}$

Además con $N(i)\backslash j'$ se indica un conjunto igual a $N(i)$ pero quitando j' y la misma cosa con $M(j)/i'$.

6 Códigos LDPC

Los códigos LDPC fueron propuestos por Gallager en los años 60, y después olvidados, se redescubrieron a finales de los años noventa por D. Mackay y R. Neal. A nivel de prestaciones superan los Turbo código (concatenación de dos códigos convolucionales sistemáticamente recursivos). Los códigos LDPC son un tipo de códigos de bloque lineal, caracterizados por una matriz de chequeo de paridad H *dispersa*, es decir con pocos unos en relación al número de ceros.
Pueden ser *regulares* o *irregulares*: un LDPC es regular si las filas y columnas de H tienen un *peso uniforme*, es decir todas las filas tienen el mismo número de unos y todas las columnas tienen el mismo número de unos, mientras será irregular si las filas y columnas tienen pesos no uniformes (típicamente las filas tienen un peso similar mientras las columnas tienen pesos muy diferentes). Los LDPC irregulares superan a los códigos turbo para grandes bloques. Además en 2000, Richardson mostró que un LDPC irregular podía llegar a 0,1 dB de la capacidad Shannon.
Como sabemos la capacidad correctora de un código está determinada da la *distancia mínima* entre las palabras códigos. Se puede demostrar que para la distancia mínima d_{min} asociada a un código *bloque*, valen las siguientes propriedades:

- d_{min} es igual al *peso mínimo* (número de unos mínimo) de las filas de la matriz generadora **G**.

- d_{min} es igual al número mínimo de columnas de la matriz de chequeo **H** que suman más que el vector nulo.

Así que la idea fundamental sobre que se basa los códigos LDPC, es que a una matriz H dispersa estará asociada una distancia mínima elevada, debido al escaso número de unos presentes respecto al número de ceros.
Para hallar la matriz generadora G a partir de H, se aplica el método de eliminación de Gauss (permutaciones de filas/columnas y substituciones con combinaciones lineales de filas/columnas) a la matriz de chequeo, para expresarla de la forma $H = [I_k \ P]$ donde I_k es una matriz unidad de dimensión k. En este modo la matriz generadora tendrá la forma $G = [P^T \ I_r]$. En general, G será grande pero no dispersa.
Siendo códigos bloques se podrían utilizar las técnicas generales de código bloque. Por ejemplo, la decodificación se podría hacer:

- con síndrome $\vec{c} \cdot H = 0$.

- Algoritmos basados en grafos: el BCJR que utiliza esquema de los grafos Trellis, y el *Message passing algorithms* para grafos bipartidos. Una subclase importante de Message passing algorithms es el *Belief Propagation* que se puede ver como una extensión del Algoritmo de Viterbi para obtener probabilidades a posteriori.

El algoritmo *BP* utiliza la formula (18) como inicialización. Una interesante subclase de los código LPDC son códigos RA e IRA.

7 Códigos RA

Los códigos RA consisten en tres pasos de codificación triviales:

- Primer bloque: Código de repetición; es el más simple código de corrección, con buenas propriedad de distancias pero baja tasa de codificación (R).

- Un paso de permutación (conocida).

- Un código convolucional con generador $G(x) = \frac{1}{1+x}$. Esto seria como "acumular", es decir va sumando los bits en entrada (en modulo 2).

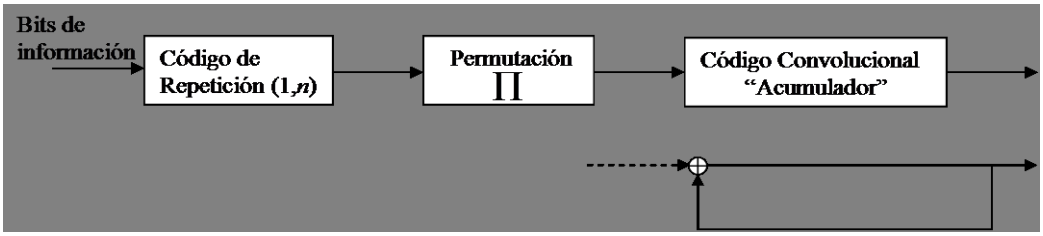

Figure 6: Código RA.

Ejemplo: RA con 2 palabras-mensaje m_1, m_2 y con un código de repetición (1,3); la salida del código de repetición:

$$m_1, m_1, m_1, m_2, m_2, m_2 \tag{25}$$

Y suponiendo una permutación $\prod = (1, 4, 6, 2, 3, 5)$ la salida del segundo bloque seria:

$$m_1, m_2, m_2, m_1, m_1, m_2 \tag{26}$$

Y el "acumulador" produce las salidas finales:

$$p_1 = m_1, \quad p_2 = m_2 + p_1, \quad p_3 = m_3 + p_2 \ \ldots \tag{27}$$

La palabra código final estaría compuesta por 8 bits (6 de redundancia); en modo *sistemático* se podría escribir así:

$$c_1 = m_1, \ c_2 = m_2, \ c_3 = p_1, \ c_4 = p_2, \ c_5 = p_3 \ldots c_8 = p_6 \tag{28}$$

Podemos representarlo con un gráfico de Tanner, donde por claridad vamos a mantener la distinción entre bit de paridad y de mensaje: en la figura (7) se pueden ver, desde la izquierda a la derecha, que el primer nodo de chequeo z_1 impone que $p_1 + m_1 = 0$, es decir $p_1 = m_1$. Luego en z_2 la condición de paridad sería $p_1 + p_2 + m_2 = 0$, que equivale a $p_2 = p_1 + m_2$. En este gráfico de Tanner, hemos añadido una división entre los bits de mensaje (abajo) y los bits de paridad (arriba).
Los bit de información m_1 y m_2 están conectados a los nodos de control según la permutación Π elegida. Ambos bit m_1 y m_2 participan a tres nodos de chequeo según Π (3 conexiones), dado que cada en este ejemplo se ha utilizado un código de repetición (1,3).

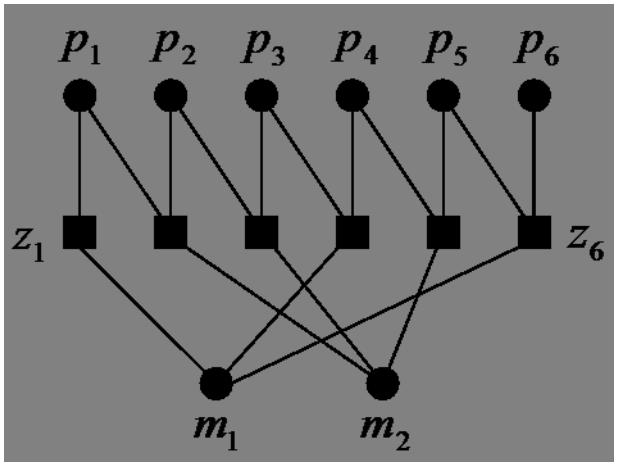

Figure 7: Gráfico de Tanner del código AR del ejemplo (se han divido los bit de mensaje, abajo, de los bits de paridad, arriba).

El paso de permutación (interlazado) permite evitar ráfagas de errores. Los código AR podría ser decodificado como un **Turbo Código** (concatenación de dos códigos convolucionales sistemáticamente recursivos), pero se puede utilizar los algoritmos que utilizan el gráfico de Tanner, como para los LDPC.

8 Código *Irregular* RA

Podemos generalizar la estructura anterior, dividiendo en bloques los k bits de información/mensaje (en el ejemplo anterior había $k = 2$ bits de información) y repitiendo los bits en cada bloque un numero de veces distintos. En concreto, se eligen q fracciones f_i en manera que:

$$\sum_{i=1}^{q} f_i = 1 \tag{29}$$

Así que los primeros $f_1 \cdot k$ bits serán repetidos 2 veces, en el bloque siguiente de $f_2 \cdot k$ bits serán repetido 3 veces, y así hasta el ultimo bloque compuesto por $f_q \cdot k$ bits que vendrán repetidos $q+1$ veces. En otras palabras, finalmente habrá $f_i \cdot k$ nodos de información conectado a $i+1$ nodos de chequeo. Se puede ver desde la figura (8) en numero de conexiones entre los bits de mensaje m_i

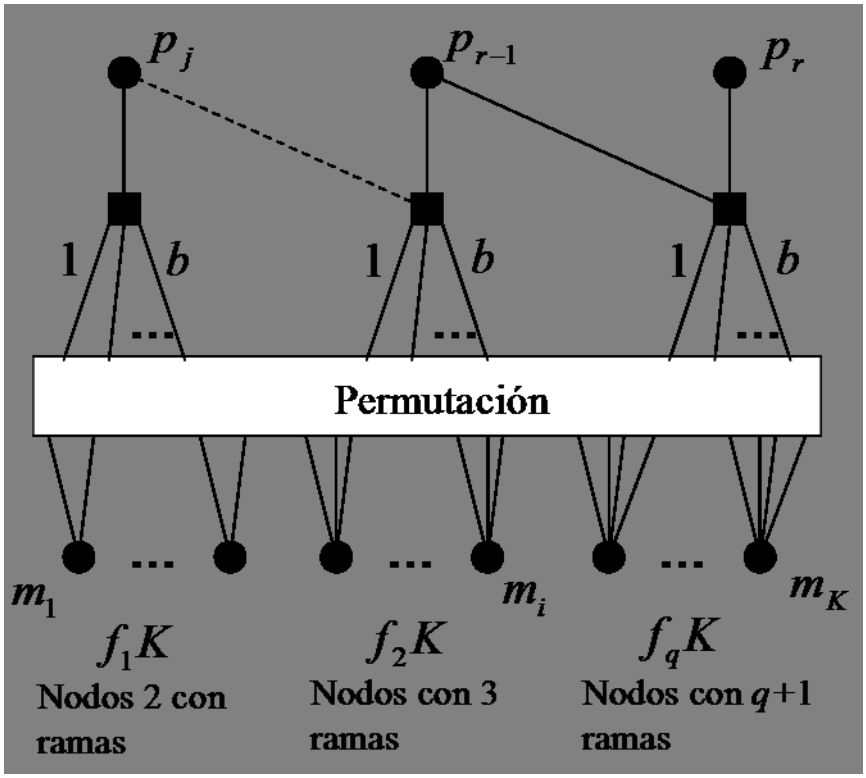

Figure 8: Gráfico de Tanner del código IRA.

(abajo) y los nodos de chequeo, va aumentando según a que bloque pertenece el bit de mensaje considerado.
Como la figura (8) hace referencia a un caso genérico, hemos añadido un bloque de permutación; este bloque quiere simbolizar solamente que cada bit de mensaje m_j del bloque i-ésimo está conectado a $i+1$ nodos de chequeo según una permutación Π.
Además cada nodo de chequeo estará conectado exactamente a b nodos de mensaje (otro parámetro del código). Así que el número de nodos de chequeo y de los bits de paridad, será igual:

$$r = \frac{k}{b} \cdot \sum_{i=1}^{q} (i+1) \cdot f_i \tag{30}$$

Para entender, volvemos un paso atrás a como se construyen los códigos AR, porque pueden ser interpretados como caso *particular* de los códigos IRA.

Por ejemplo, mirando el gráfico de Tanner (7) se puede ver que cada nodo de chequeo (nodos cuadrados) está conectado siempre solo a un nodo de mensaje (abajo), es decir para los código AR $b = 1$ siempre. Además en el casos de los AR q sería igual a k, es decir habría k bloques con solo 1 bit de información y, indicando con n la tasa de repetición del código interno a la estructura AR, las fracciones f_i serán todas nula menos una ($f_1 = 0, \cdots, f_{n-1} = 1, \cdots, f_q = 0$):

$$f_{i-1} = \delta_{i-n} \tag{31}$$

Donde con δ_{i-n} indicamos una función delta discreta que asume valor unitario en $i = n$, y es nula en caso contrario $i \neq n$. La (31) quiere decir que todos los nodos de información están conectado a n nodos de chequeo, o de otra forma, que todos los bits de mensaje vienen replicados con la misma tasa n (la fracción f_{i-1} de bits que serán repetidos i veces será cero, en general; la fracción será 1, es decir todos los bits, solo si $i = n$).
Así que, en el caso del ejemplo del párrafo anterior, repitiendo 3 veces los 2 bits de información vamos a tener $r = 6$ nodos de control. Esto porque, como hemos dicho, cada nodo de chequeo está conectado solo a un bit de mensaje ($b = 1$); substituyendo en la formula general (30) los valores $q = k$, $f_1 = 0$ y $f_{n-1} = f_2 = 1$ ($n = 3$), $b = 1$, $k = 2$, hallaríamos el mismo resultado.
En el caso de los códigos IRA con $b = 1$, teniendo $f_i \cdot k$ bits de información replicados $i + 1$ veces, los nodos de control para este bloque serían $(i+1)f_i k$. Así que el numero total de nodos de chequeo con $b = 1$, sería $\sum_{i=1}^{q}(i+1)f_i k$. Para un valor de b cualquiera, logramos la (30).
El número de bits de paridad será también r, como los nodos de control (círculos superiores de la fig.8). Así que, para un esquema sistemático donde las palabras códigos toman la forma $\{m_1, \cdots, m_k, p_1, \cdots, p_r\}$, la tasa del código será el numero de bits de mensaje k dividido por el numero total de bits $k + r$:

$$R_s = \frac{k}{k+r} = \frac{k}{k + \frac{k}{b}\sum_{i=1}^{q}(i+1)f_i} = \frac{b}{b + \sum_{i=1}^{q}(i+1)f_i} \tag{32}$$

Otro posibilidad sería considerar como palabras códigos solo $\{p_1, \cdots, p_r\}$; en este caso tendríamos un código no sistemático con tasa:

$$R_{nos} = \frac{k}{r} = \frac{k}{\frac{k}{b}\sum_{i=1}^{q}(i+1)f_i} = \frac{b}{\sum_{i=1}^{q}(i+1)f_i} \tag{33}$$

Aunque los código IRA son simplemente un *caso especial de los irregular*

LDPC, tienen la ventaja respecto a los *irregular LDPC* que la codificación muy eficiente y de complejidad menor. En general los RA son códigos con complejidad *lineal* en codificación y en decodificación, y al mismo tiempo logran óptimas prestaciones en decodifica.

9 Bibliografía

[1] Apuntes de Comunicaciones Digitales "Codificación de Canal". Marcelino Lázaro.

[2] "Comunicaciones Digitales". A. Artés, F. Pérez González, J. Cid, R. López, C. Mosquera, F. Pérez Cruz.

[3] "Elements of information Theory". Second Edition. Thomas M. Cover, Joy A. Thomas.

[4] "LDPC Codes: An Introduction". Amin Shokrollahi.

[5] "An Introduction to Low Density Parity Check". Jian Sun.

[5] "Error Correction Coding" Todd K. Mood.