Codificación de Canal

Códigos Convolucionales

Manuel A. Vázquez
Jose Miguel Leiva
Joaquín Míguez

7 de marzo de 2024

Índice

Índice

Códigos bloque lineales vs. códigos convolucionales

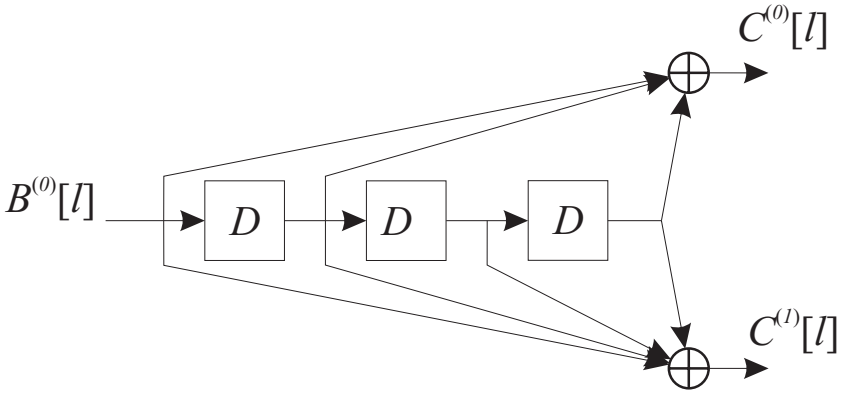
Algunas diferencias (relacionadas entre sí)...

Códigos convolucionales	Códigos bloque lineales
• La codificación es continua	• La codificación es *bloque a bloque*
• El sistema tiene memoria	• El sistema **no tiene memoria**
• *Mapping* secuencia-a-secuencia	• *Mapping* mensaje-a-palabra código

En ambos esquemas, todas las operaciones (e.g., la convolución), son en $GF(2)$.

Especificación

A menudo los códigos se definen mediante un diagrama de bloques que ilustra como la entrada se transforma en la salida, e.g.

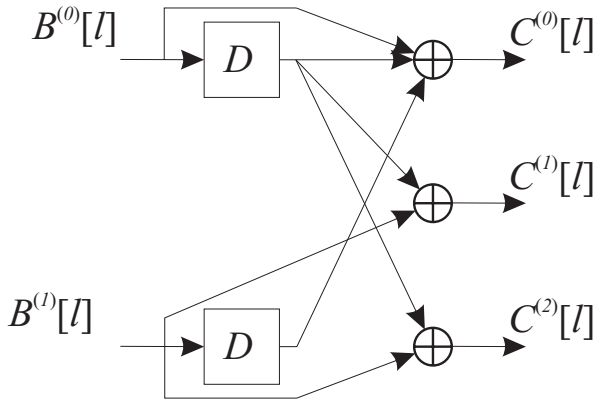


- $B^{(j)}[l]$ es el l-ésimo bit de la j-ésima entrada
- $C^{(j)}[l]$ es el l-ésimo bit de la j-ésima salida
- D es un elemento de retardo

El sistema tiene memoria

Los bits de salida dependen de los bits de entrada anteriores (y actuales): es una **máquina de estados**.

Especificación: ecuaciones



- Relación entre las entradas y las salidas

$$
\begin{aligned}
C^{(0)}[l] &= B^{(0)}[l] + B^{(0)}[l-1] + B^{(1)}[l-1] \\
C^{(1)}[l] &= B^{(0)}[l-1] + B^{(1)}[l] \\
C^{(2)}[l] &= B^{(0)}[l-1] + B^{(1)}[l]
\end{aligned}
$$

- expresaremos estas relaciones en el dominio D...

Convolución

...pero, dónde está la convolución en un código *convolucional*?

Convolución de señales en tiempo discreto

$$x[n] * h[n] = h[n] * x[n] = \sum_{k=-\infty}^{\infty} h[k]x[n-k]$$

y suponiendo que la respuesta al impulso, $h[k]$, es no nula entre los instantes de tiempo, e.g., 0 y 1

$$= h[0]x[n] + h[1]x[n-1]$$

Para la primera salida, por ejemplo, tenemos

$$C^{(0)}[l] = \underbrace{B^{(0)}[l] + B^{(0)}[l-1]}_{B^{(0)}[l]*\begin{bmatrix}1 & 1\end{bmatrix}} + \underbrace{B^{(1)}[l-1]}_{B^{(1)}[l]*\begin{bmatrix}0 & 1\end{bmatrix}}$$

Cada salida es una suma de convoluciones!!

Transformada D

Definición: La transformada D...

...de una *secuencia* binaria $B^{(i)}[l]$ es

$$B^{(i)}(D) = \sum_u B^{(i)}[u] \cdot D^u$$
$$= \cdots B^{(i)}[-1] \cdot D^{-1} + B^{(i)}[0] + B^{(i)}[1] \cdot D^1 + \cdots$$

...y decimos $B^{(i)}[l] \overset{D}{\longleftrightarrow} B^{(i)}(D)$

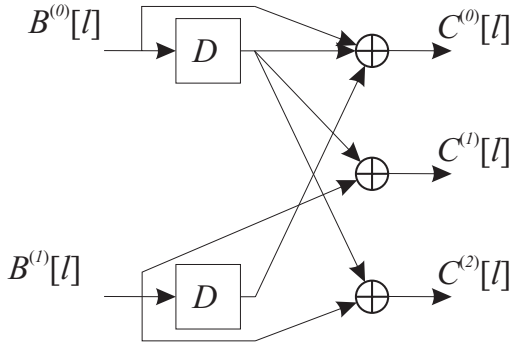
con la propiedad

$$\boxed{B^{(i)}[l-d] \leftrightarrow B^{(i)}(D) \cdot D^d}$$

de manera que, e.g.,

$$B^{(i)}[l] + B^{(i)}[l-1] + B^{(i)}[l-2] + B^{(i)}[l-3] \overset{D}{\longleftrightarrow} B^{(i)}(D)(1 + D + D^2 + D^3).$$

Matriz generadora



- En forma polinomial

$$C^{(0)}(D) = (1+D)B^{(0)}(D) + DB^{(1)}(D)$$
$$C^{(1)}(D) = DB^{(0)}(D) + B^{(1)}(D)$$
$$C^{(2)}(D) = DB^{(0)}(D) + B^{(1)}(D)$$

- En forma matricial: $\mathbf{C}(D) = \mathbf{B}(D)\mathbf{G}(D)$, con

$$\mathbf{C}(D) = \begin{bmatrix} C^{(0)}(D) & C^{(1)}(D) & C^{(2)}(D) \end{bmatrix} \quad \mathbf{B}(D) = \begin{bmatrix} B^{(0)}(D) & B^{(1)}(D) \end{bmatrix}$$

$$\mathbf{G}(D) = \begin{bmatrix} 1+D & D & D \\ D & 1 & 1 \end{bmatrix}_{k \times n}$$

$\mathbf{G} \equiv$ matriz generadora
$g_{ij} \equiv$ contribución de la i-ésima entrada a la j-ésima salida

Definiciones

Definición: Memoria total...

...del código, M_t, es el número de unidades de retardo en el esquema de codificación.

$$M_t = \sum_{i=0}^{k-1} M^{(i)}$$

con

$$M^{(i)} = \max_j \text{grado}\left(g_{ij}(D)\right) \equiv \text{memoria entrada } i\text{-ésima}$$

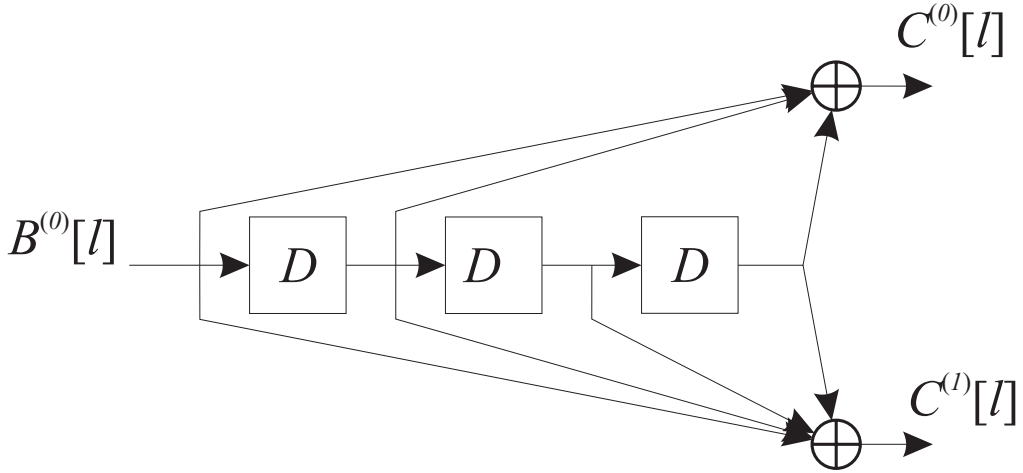
Definición: Longitud de restricción

...del código, K, es la longitud máxima de la respuesta al impulso,

$$K = 1 + \max_{i,j} \text{grado } \left(g_{ij}(D)\right)$$

Un código convolucional también puede ser **sistemático** (misma definición).

El codificador como máquinas de estados finitos



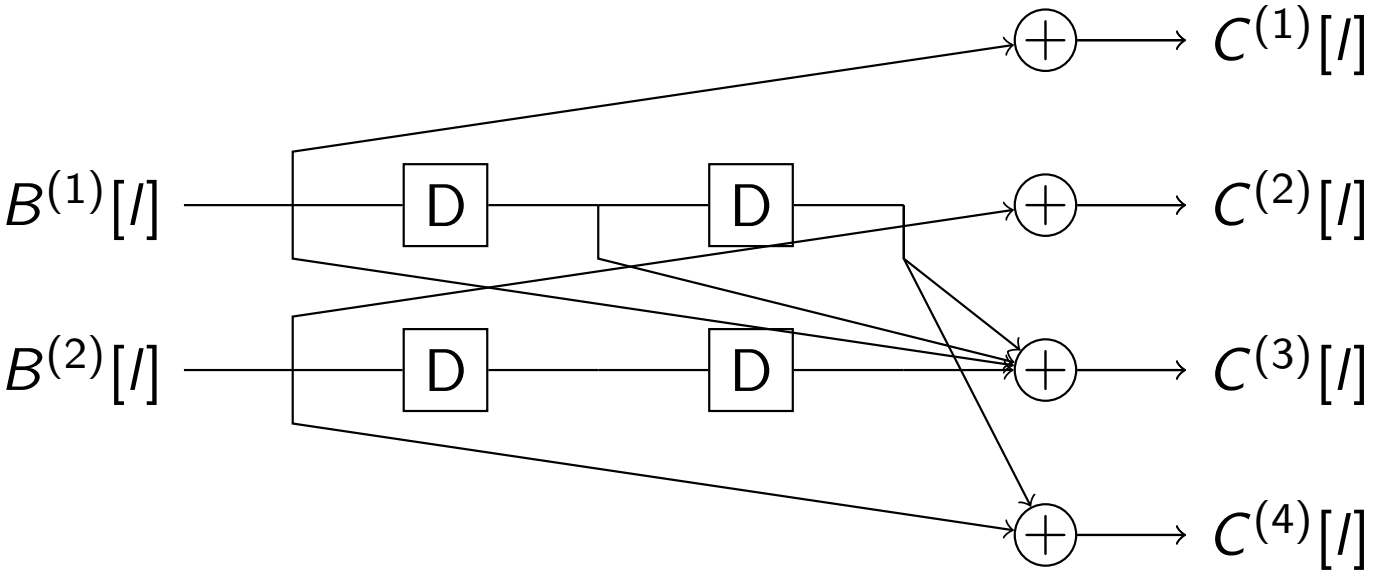
- El *estado* del codificador viene dado por los bits almacenados (producidos a la salida) de los elementos de retardo, aquí

$$\Psi \equiv (B^{(0)}[I-1], B^{(0)}[I-2], B^{(0)}[I-3]).$$

- **En general hay 2^{M_t} posibles estados** (se concatenan los bits almacenados por los elementos de retardo de todas las entradas). Una posible asignación en este caso:

$$\Psi_0 \to (0,0,0) \quad \Psi_1 \to (1,0,0) \quad \Psi_2 \to (0,1,0) \quad \Psi_3 \to (1,1,0)$$
$$\Psi_4 \to (0,0,1) \quad \Psi_5 \to (1,0,1) \quad \Psi_6 \to (0,1,1) \quad \Psi_7 \to (1,1,1)$$

Transición de estado I



Veamos la evolución del estado del sistema para un un ejemplo sencillo...

I	-2	-1	0	$+1$
$B^{(1)}[I]$	0	0	1	1
$B^{(2)}[I]$	0	0	0	1
	bits anteriores		**bits a codificar**	

Transición de estado II

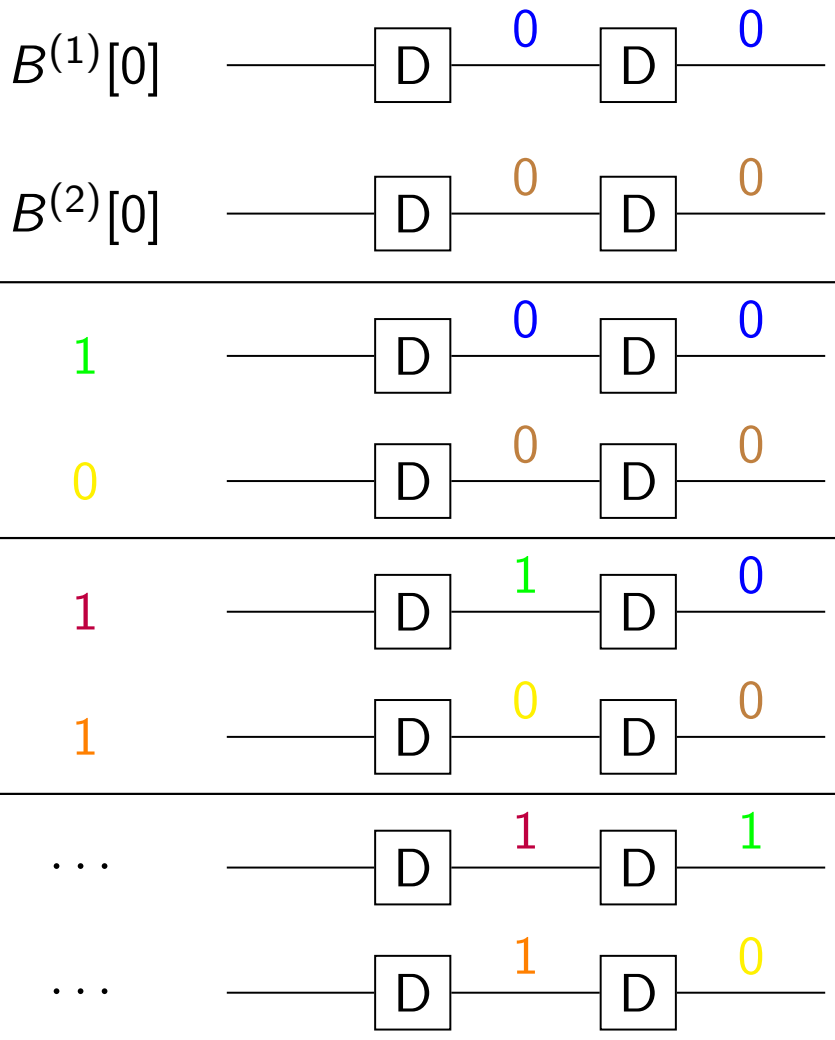
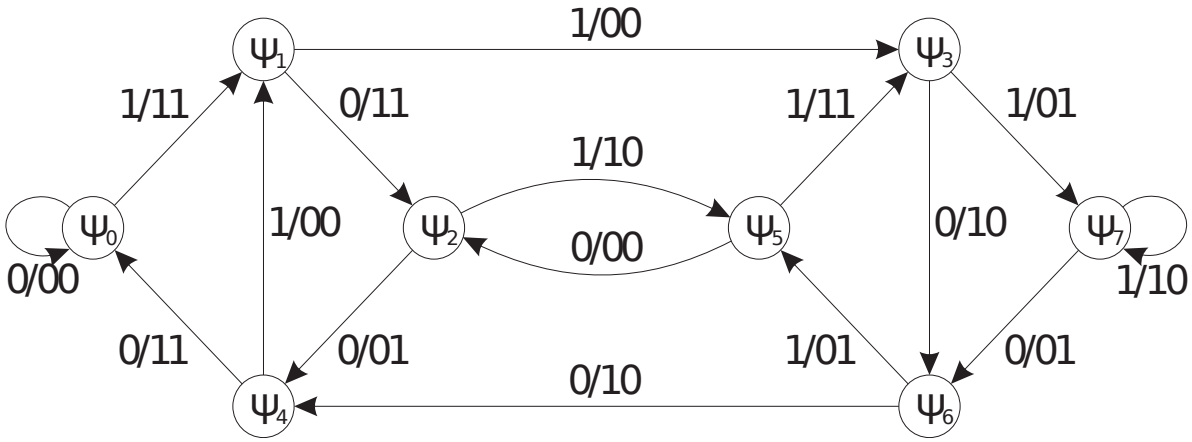


Diagrama de estados



Cada flecha esta etiquetada con

- los bits entrantes que dan lugar a esa transición, y
- los bits de salida que tienen lugar para ese estado y esos bits entrantes.

Índice

Codificación

Trivial si conocemos

- el estado inicial
- el diagrama de estados

Ejemplo

Supongamos que partimos del estado Ψ_0 y queremos codificar la secuencia [001]. El resultado es

$$[00\ 00\ 11]$$

Cabecera

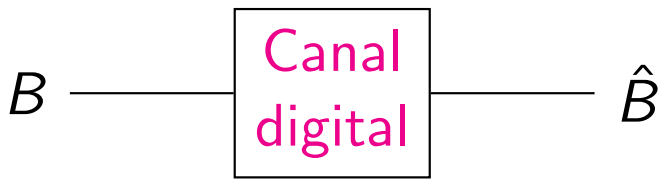
Después de la secuencia de information, se añade una *cabecera* para forzar al codificador a volver al estado inicial.

información	cabecera

Índice

Decodificación

Dura



- Métrica: distancia de **Hamming**

Blanda

- Métrica: distancia **euclídea** (en cada paso, diferencia al cuadrado)

En ambos casos:

- el objetivo es encontrar la secuencia *completa* más probablemente transmitida
- la solución viene dada por el algoritmo de Viterbi

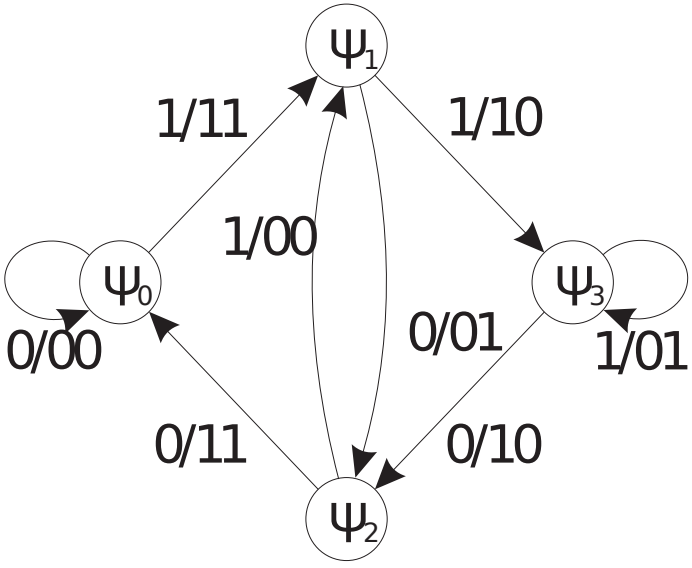
Algoritmo de Viterbi

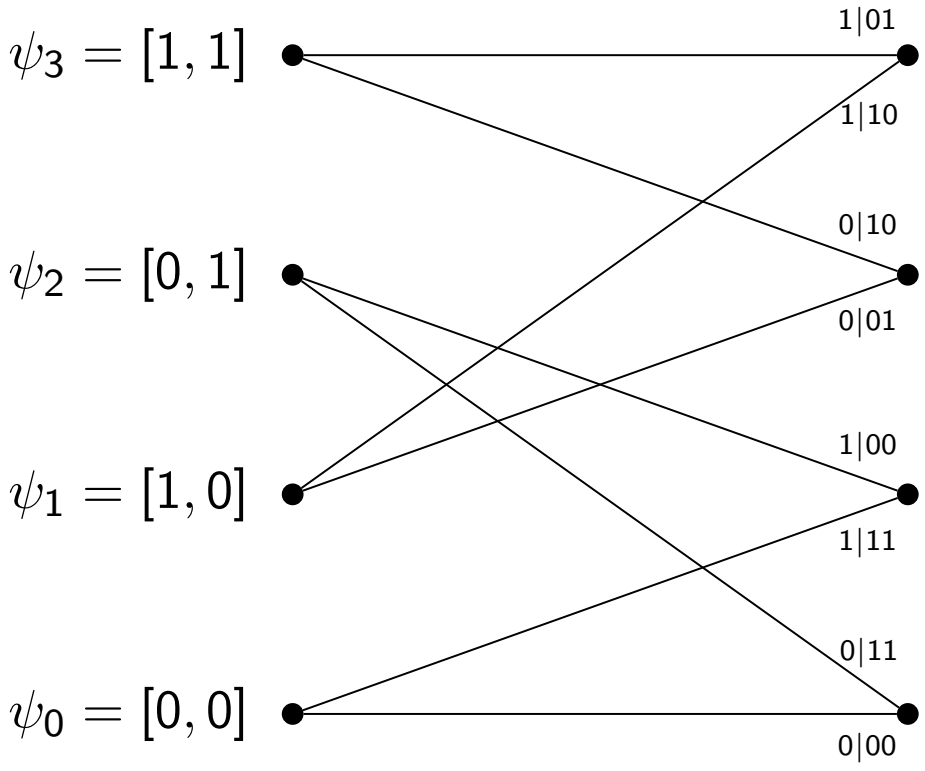
Algunas claves

- Tenemos que evaluar todas las posible trayectorias que comienzan en el estado inicial (típicamente Ψ_0).
- Para cada posible transición, comparamos su salida asociada con la observada.
- En cada instante de tiempo, para cada posible estado, tenemos que encontrar el camino que llega hasta él con el menor coste acumulado
- Cuando dos caminos llegan al mismo estado, nos quedamos con el de menor coste *acumulado*.
- Debido a la *cabecera* que se añade después de transmitir una secuencia de información, la decodificación debe terminar en el estado inicial.

Ejemplo sin errores

- Secuencia de entrada: 1 1 0 1 0 1 0 0
- Secuencia recibida: 11 10 10 00 01 00 01 11

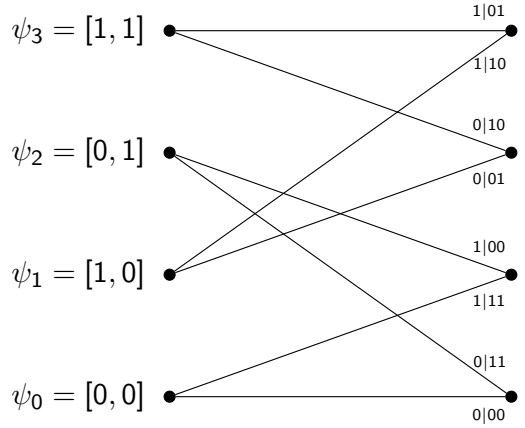


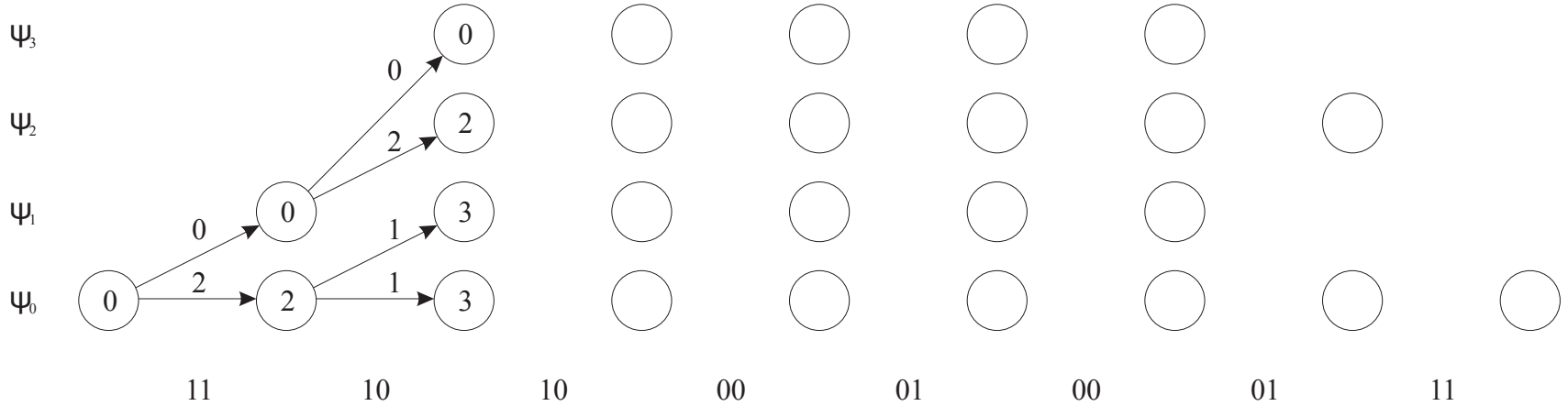


(Representación en **trellis**)

Ejemplo sin errores

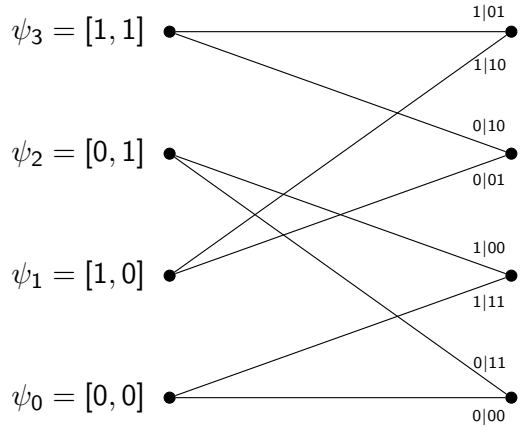
- Dos primeras iteraciones

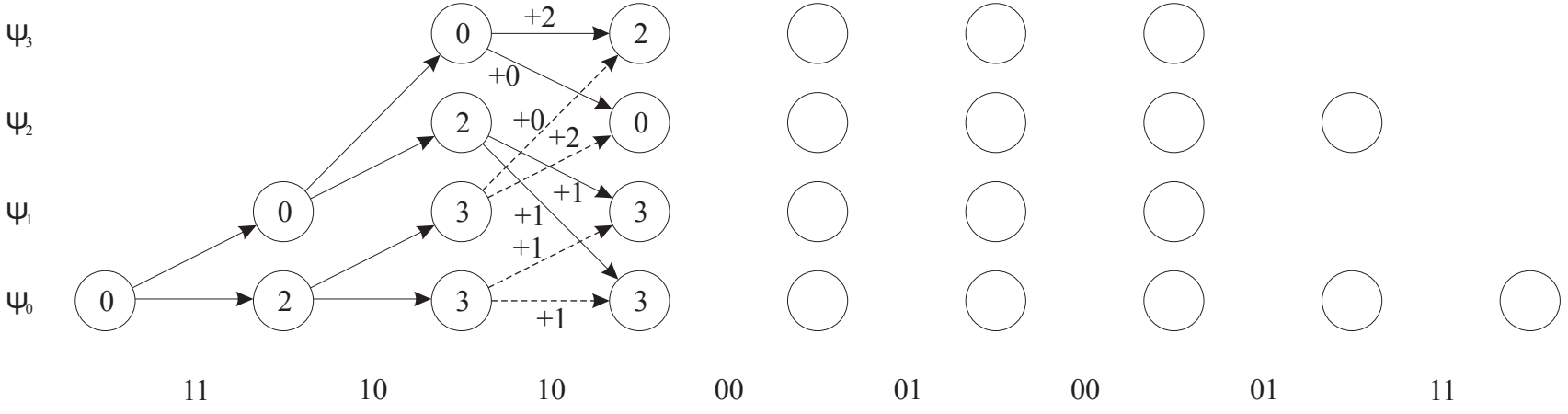




Ejemplo sin errores

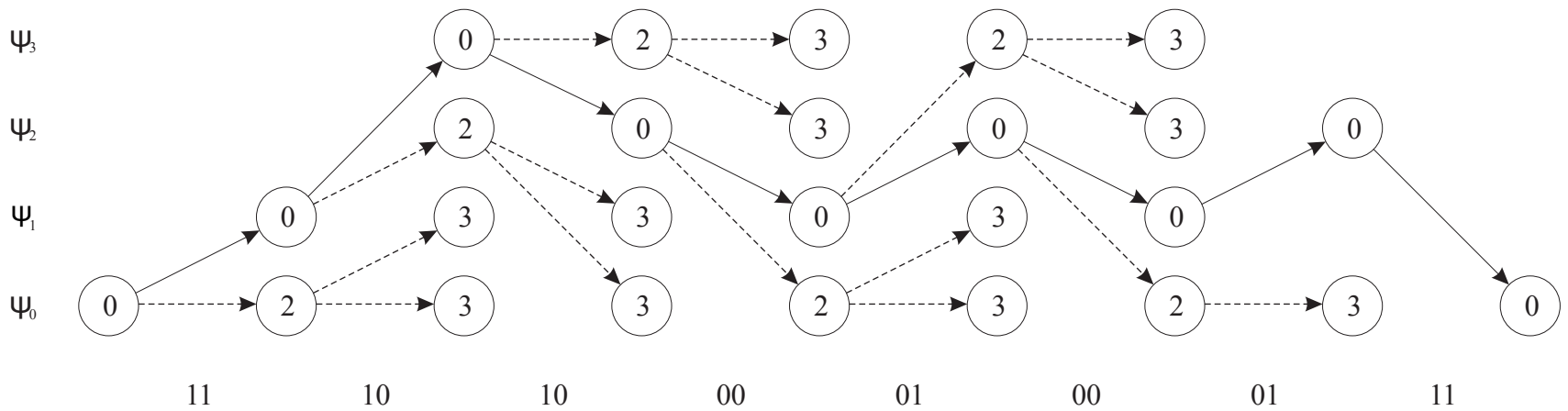
- Tercera iteración





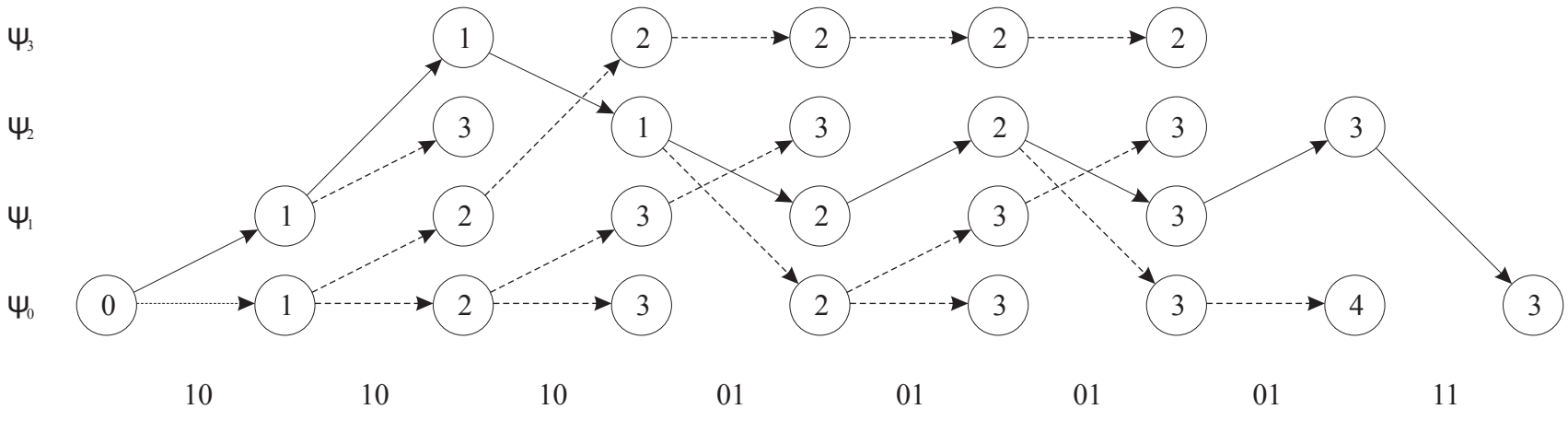
Ejemplo sin errores

- Resultado final



Ejemplo **con** errores

- Secuencia recibida: 10 10 10 01 01 01 01 11
- Resultado final



- La secuencia de estados $\Psi_0\Psi_1\Psi_3\Psi_2\Psi_1\Psi_2\Psi_1\Psi_2\Psi_0$ se corresponde con la secuencia de entrada 11010100

Prestaciones

- Decodificación blanda

$$P_e \approx \kappa_2 \mathrm{Q}\left(\sqrt{\frac{2D_{min}E_s}{N_0}}\right)$$

donde κ_2 es el número de errores de bit (en la secuencia *decodificada*) que provoca la trayectoria asociada a D_{min}.

- Decodificación dura

$$P_e \approx \kappa_2 \sum_{i=\lfloor (D_{min}-1)/2 \rfloor + 1}^{nz} \binom{nz}{i} \epsilon^i (1-\epsilon)^{nz-i}$$

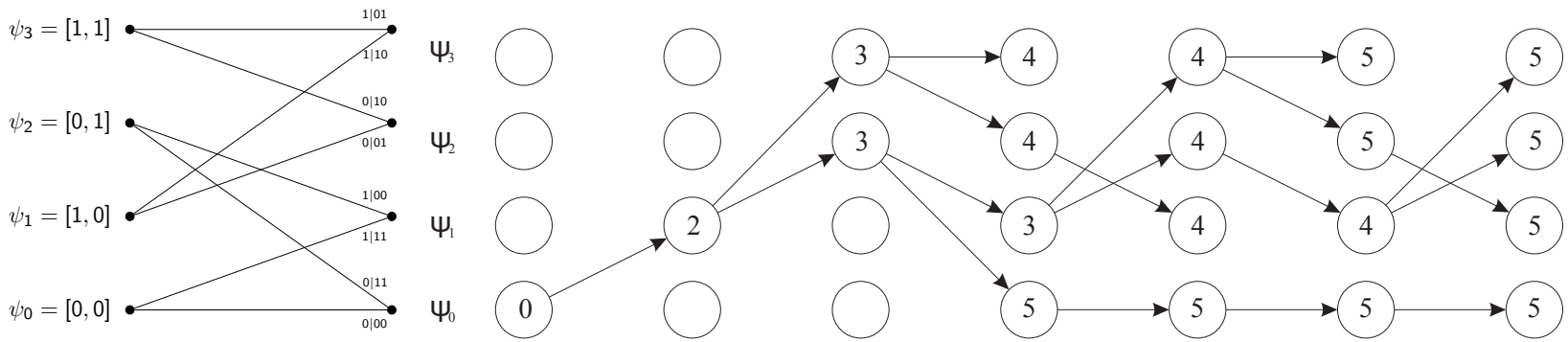
donde z es la longitud de la trayectoria asociada a D_{min} y ϵ es la probabilidad de error de bit.

Determinación de la distancia mínima, D_{min}

Un código convolucional es lineal, por lo que **la secuencia codificada más cercana a la todo-ceros determina** D_{min}.

Objetivo

Buscamos la secuencia de estados (camino) que empieza en la secuencia todo-zeros y vuelve a ella con el menor coste acumulado.



D_{min} permite calcular los otros parámetros que determinan las prestaciones

$$D_{min} = 5 \Rightarrow \begin{cases} \kappa_2 = 1 \\ z = 3 \end{cases}$$

Decodificación blanda

Por simplicidad, asumimos modulación antipodal i.e., $A[l] = \pm A$.

Notación

$$B_{0,:} = \{B[0], B[1], B[2], \cdots\} \equiv \text{bits de entrada}$$
$$q_{0,:} = \{q[0], q[1], q[2], \cdots\} \equiv \text{estimaciones blandas}$$

2 posibilidades

- Decodificación blanda de **secuencia**: minimiza la probabilidad de error de secuencia

$$\hat{B}_{0,:} = \arg \max_{B_{0,:}} p(q_{0,:}|B_{0,:})$$

- Decodificación blanda **bit a bit**: minimiza la probabilidad de error de bit

$$\hat{B}[i] = \arg \max_{B[i]} p(B[i]|q_{0,:}), i = 0, 1, \cdots$$

Decodificación blanda de secuencia

criterio ML $\equiv$ criterio MAP

$$\hat{B}_{0,:} = \arg\max_{B_{0,:}} p(q_{0,:}|B_{0,:}) = \arg\max_{B_{0,:}} \prod_{l} p(q[l]|B[l])$$

$$= \arg\min_{B_{0,:}} \sum_{l} -\log p(q[l]|B[l])$$

donde

$$p(q[l]|B[l]) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(q[l]-A[l])^2}{2\sigma^2}\right).$$

Se implementa con el algoritmo de Viterbi

Decodificación blanda de bit

criterio MAP

Aplicamos el criterio máximo a posteriori (MAP) a nivel de bit para calcular

$$P(B[l] = 1 | q_{0,:})$$

y decidimos

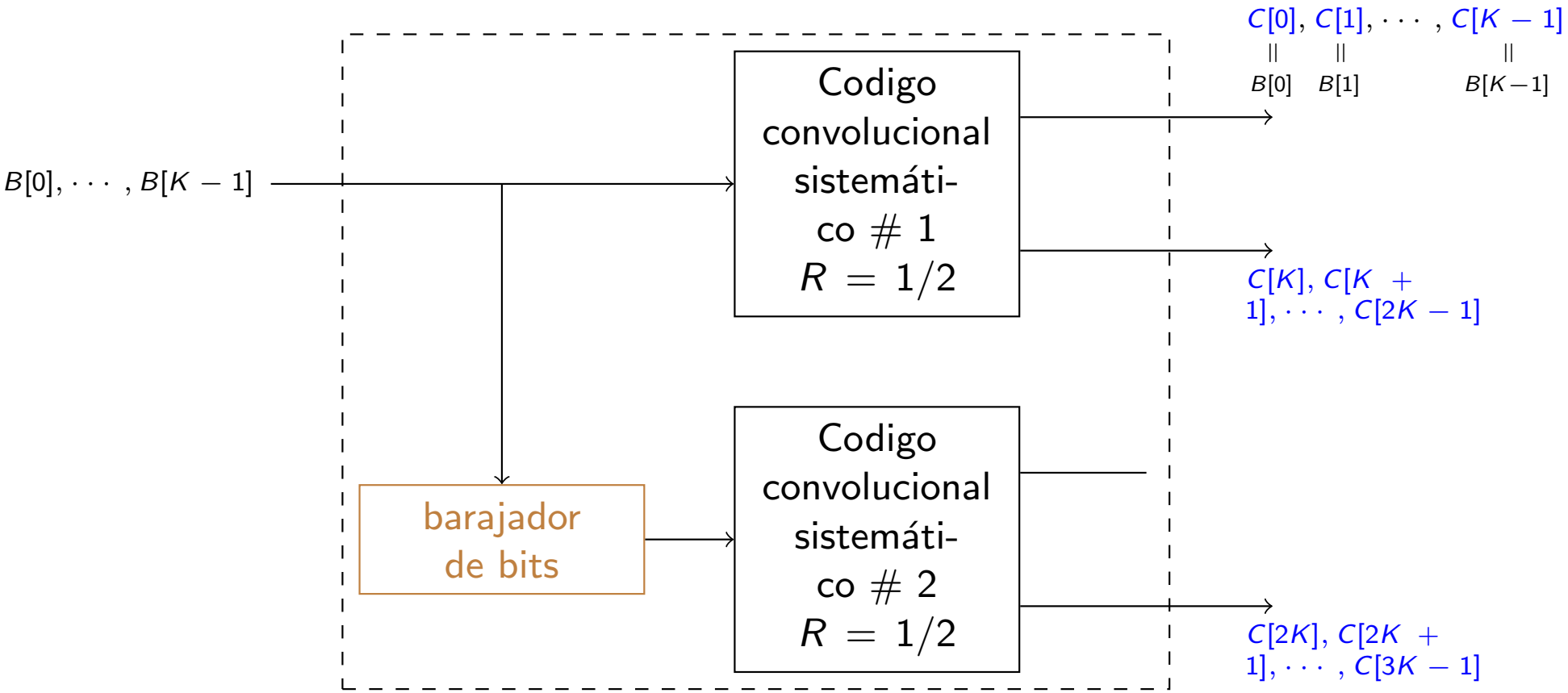
$$\hat{B}[l] = \begin{cases} 1 & \text{si } p(B[l] = 1 | q_{0,:}) > P(B[l] = 0 | q_{0,:}) \\ 0 & \text{en otro caso} \end{cases}$$

Se implementa con el algoritmo **BCJR** (Bahl, Cocke, Jelinek y Raviv)

Índice

Códigos turbo

- Se forman concatenando códigos convolucionales, que operan sobre los bits ordenados de distinta forma.
- Principales elementos:
  - 2 códigos convolucionales
  - barajador de bits



Observaciones

- El barajador se usa para aumentar la memoria del sistema sin aumentar la complejidad de decodificación.
- Buenas prestaciones a baja SNR (a 0.7 dB del límite de Shannon).
- En cuanto a la decodificación...
  - Es iterativa, ya que hay que "poner de acuerdo"la decodificación de la secuencia original y de la versión "barajada".
  - Se basa en el algoritmo BCJR