Actividad 12.2

1)

Codificación convolucional

Existen dos estrategias posibles para recibir de forma fiable y libre de errores la información transmitida desde una fuente:

- ARQ (Automatic Repeat Request), basada en la detección de errores pero sin la posibilidad de corrección, solicitando al transmisor la repetición del mensaje en caso de error.
- FEC (Forward Error Correction), basada en la detección y corrección en el extremo receptor de los posibles errores.

En ambos casos es necesario añadir cierta redundancia al mensaje a transmitir para detectar o corregir estos errores, este proceso se denomina codificación del canal.

La Codificación Convolucional con la decodificación de Viterbi es de las técnicas FEC más adecuadas en canales en los que la señal transmitida se ve corrompida principalmente por ruido gausiano blanco y aditivo (AWGN).

Los Codificadores de Bloque, si bien pueden detectar errores, no pueden corregir el dato de llegada cuando hay más de un bit erróneo.

En la Codificación Convolucional es posible desarrollar un decodificador que corrija errores múltiples en los datos originales que fueron afectados con error.

Los códigos convolucionales se describen a partir de ciertos elementos

- **k** número de bits que entran al codificador
- **n** número de bits que salen del codificador
- **R** tasa de código en k/n
- **K** longitud del código
- **m** memoria del codificador

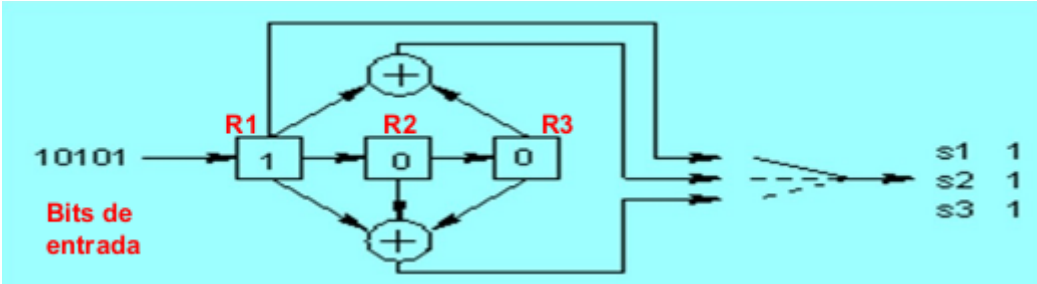
La longitud de código K denota en cuántos ciclos de codificación tiene influencia un bit que tengamos a la entrada del mismo a partir de un instante dado, ya que este bit en un instante dado irá recorriendo la cadena de FF del shift-register del Codificador.

Características de la Codificación Convolucional:

- Los Codificadores Convolucionales se basan en crear dependencia temporal y secuencial entre los bloques codificados.
- Estos condicionan la probabilidad de error de acuerdo al orden de llegada de cada bit del mensaje.
- Las operaciones de codificación no solo dependen de los calores de cada bit del bloque a procesar, sino también del orden de llegada de cada bloque.
- Gracias a la memoria, el codificador maneja estados que están determinados de acuerdo a la configuración de valores de la misma.
- Debido a la dependencia secuencia del proceso de cada bloque, se reduce la probabilidad de error porque genera eventos dependientes a través del tiempo.
- Esta característica se conoce como un Proceso Estocástico de Markov, el cual asegura que: “La probabilidad de eventos futuros no solo están condicionados al estado presente sino también al resultado del estado anterior.”

A continuación se presenta un ejemplo para la presentación de la arquitectura de los Códigos Convolucionales.

Se muestra un codificador básico con 3 registros, de los cuales R1 es un registro de tránsito, y R2 y R3 son registros de memoria. Después de procesado el bit, este se mueve hasta el registro siguiente.



Según la configuración de este diagrama, el circuito está configurado con k=1 entradas, restricción de palabras K=3, y salidas n=3, con m=2 memorias.

Las operaciones del circuito anterior para cada salida son:

S1 = R1
S1 = R1 **xor** R3
S1 = R1 **xor** R2 **xor** R3

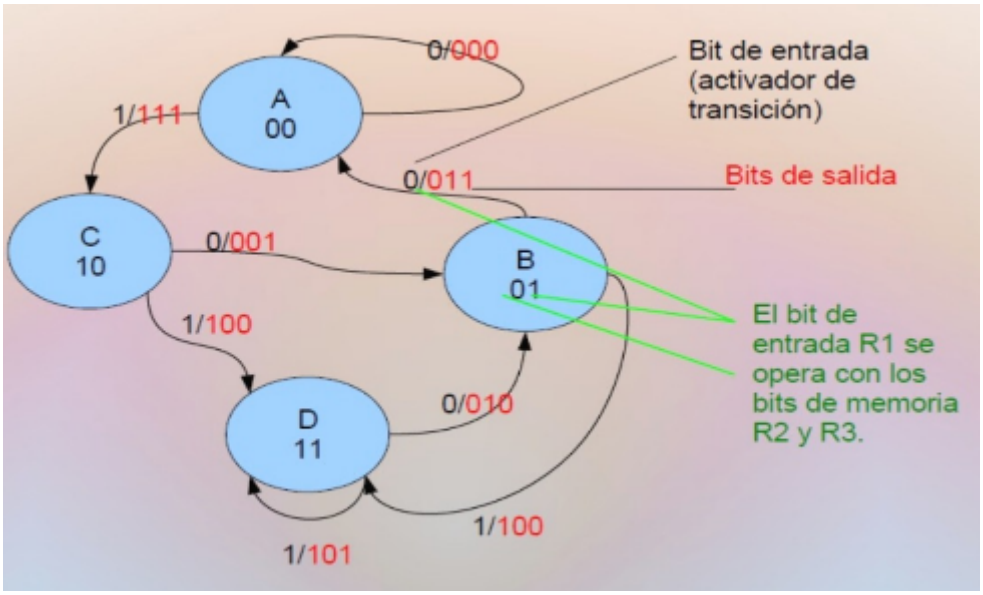
R1	R2	R3	S1	S2	S3
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	0	1
0	1	1	0	1	0
1	0	0	1	1	1
1	0	1	1	0	0
1	1	0	1	1	0
1	1	1	1	0	1

Debido a que tiene 2 registros de memoria, los posibles estados son 4

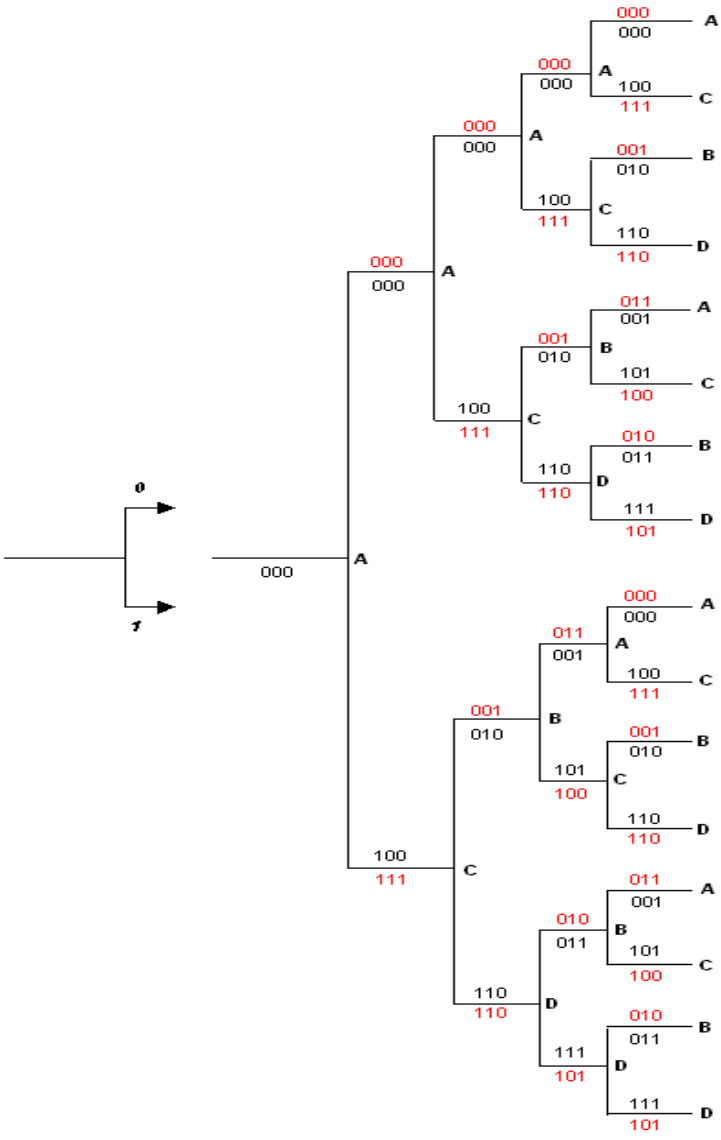
R1	R2	Estado
0	0	A
0	0	B
0	1	C
0	1	D
1	0	A
1	0	B
1	1	C
1	1	D

R1	R2	R3	S1	S2	S3	Estado
0	0	0	0	0	0	A
0	0	1	0	1	1	B
0	1	0	0	0	1	C
0	1	1	0	1	0	D
1	0	0	1	1	1	A
1	0	1	1	0	0	B
1	1	0	1	1	0	C
1	1	1	1	0	1	D

De donde se obtiene el siguiente diagrama de estados



Pudiendo obtenerse el diagrama de árbol del mismo



Características del diagrama de árbol:

La profundidad del árbol es 2*(m-1), y el número de estados es 2*(m-1)*k. La interpretación del árbol del código es la siguiente:

- Hay dos ramas en cada nodo
- La rama superior corresponde a una entrada de ‘0’
- La rama inferior corresponde a una entrada de ‘1’
- En la parte exterior de cada rama se muestra el valor de salida
- El número de ramas se va multiplicando por dos con cada nueva etapa

A partir del segundo nivel, el árbol se vuelve repetitivo. En realidad, solo hay cuatro tipos de nodos: A, B, C y D. Estos tipos de nodos en realidad son estados del codificador. A partir de estos nodos, se producen los mismo bits de salida y el mismo estado. Por ejemplo, de cualquier nodo etiquetado como C se producen el mismo par de ramas de salida: salida 001, estado B y salida 110 y estado D.

Diagrama de Trellis (o enrejado)

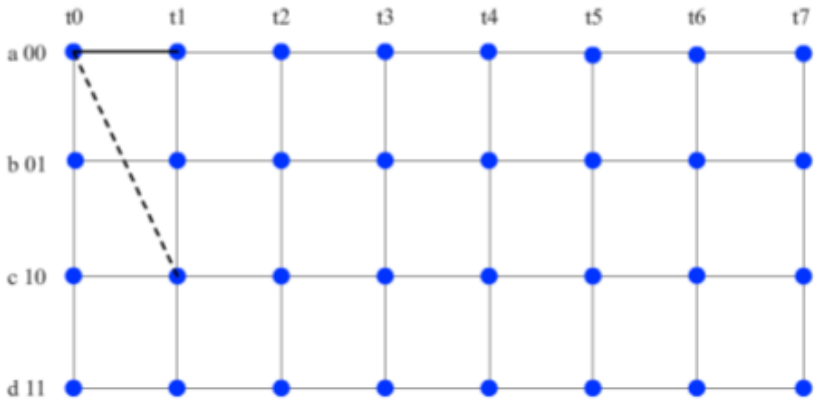
El árbol de Trellis se basa en los estados del codificador y en sus posibles caminos. De cada nodo parten ramas hacia los nodos siguientes, permitiendo representar de forma lineal la secuencia de los eventos, el único inconveniente es que luce algo desordenado.

Elaboración del diagrama de Trellis:

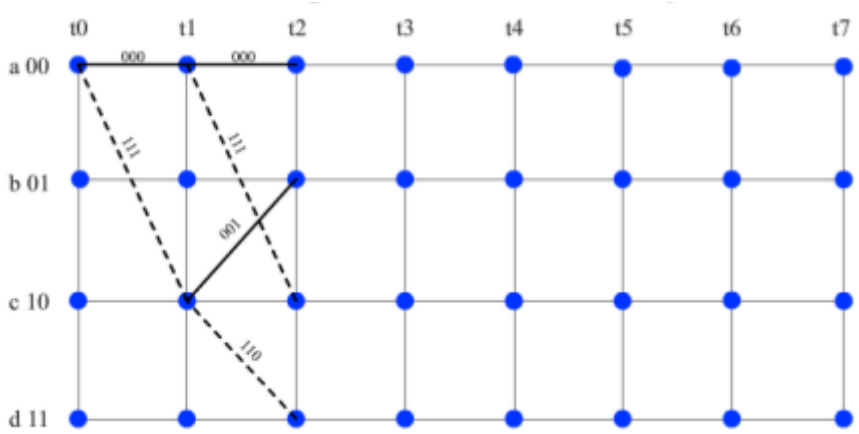
- Se tiene 4 estados, los mismos del diagrama de estados

R1	R2	Estado
0	0	A
0	0	B
0	1	C
0	1	D
1	0	A
1	0	B
1	1	C
1	1	D

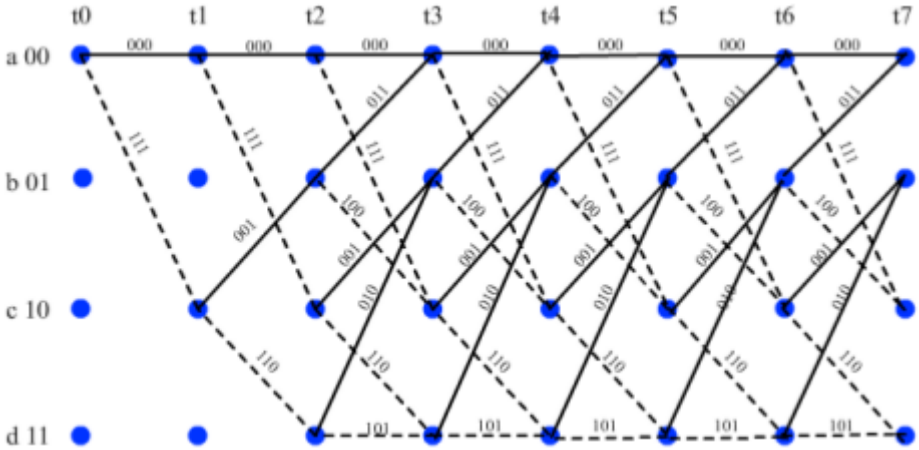
- Si al codificador de Trellis entra un ‘1’ se pinta la trayectoria hacia el otro estado con línea punteada, de lo contrario se pinta con línea continua
- El estado inicial es el estado A = 00 y las trayectorias se pintan de izquierda a derecha
- Cada transición de tiempo indicará que un nuevo bit ha ingresado
- El eje “x” es usado para representar el tiempo de todos los posibles estados y el eje “y” muestra todos los posibles estados del código(A, B, C ó D)
- Partimos del estado A(00) y t0, a partir de aquí se trazan dos líneas desde este estado, una línea para el caso en que la entrada sea ‘0’ y otra línea para el caso que la entrada sea ‘1’, esto corresponde a t1



De la misma manera se deberá construir el diagrama para cada nodo que se vaya creando. En t2 el diagrama quedaría así



Cada transición de tiempo indicará que un nuevo bit ha ingresado. Para este caso el diagrama se estabilizará en t3 dado que seguirá la misma secuencia y por consiguiente realizará los mismos recorridos. Finalmente el diagrama quedará así



Algoritmo de Viterbi:

El problema que se aborda con este algoritmo es que desde la teoría de la codificación, considerando como transparente el canal y los bloques modulador-demodulador. Para ello consideramos los tres bloques como uno solo, el canal digital.

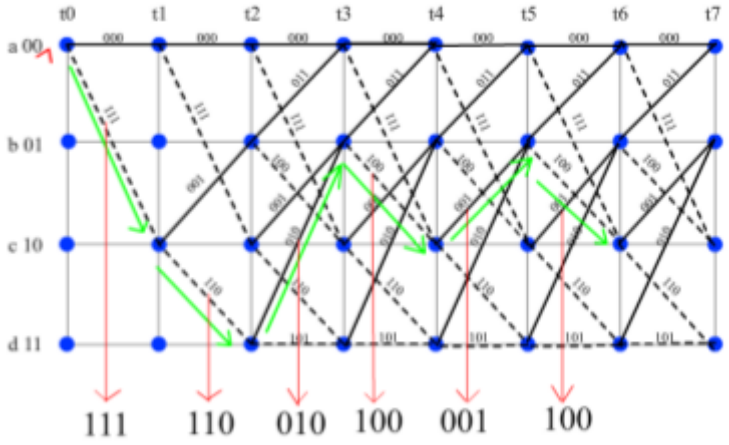
Decodificación de Viterbi:

- Fortaleza en capacidad de corregir errores y decodificar más de un error
- El decodificador es también una máquina de estados similar al codificador, con la misma correspondencia de estados y transiciones
- Se basa en el diagrama de Trellis, incluyendo todas las posibles rutas y combinaciones, para finalmente, con la distancia de Hamming de todas las posibles rutas, tomar la de menor peso (dado por la menor distancia de Hamming)

Supongamos que volvemos al ejemplo de hoy, cuyo dato codificado es

111 110 010 100 001 100

y le corresponde el siguiente diagrama de Trellis



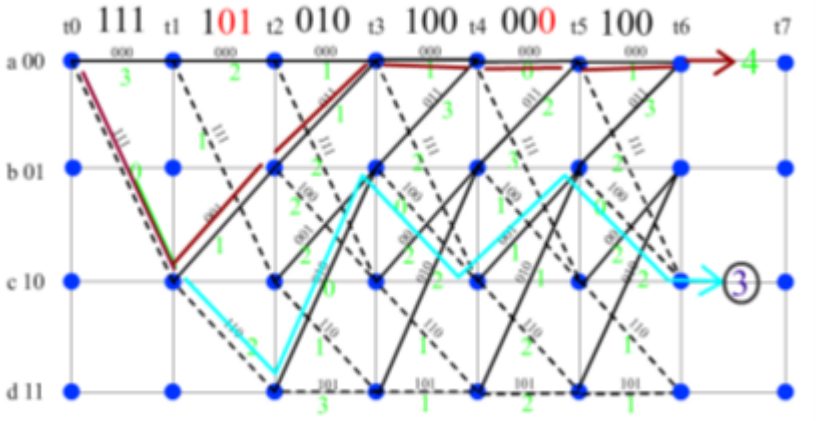
Ahora bien, supongamos que en el proceso de transmisión se dañaron 3 bits (en rojo)

111 1**1**0 010 100 00**1** 100

cambiando a

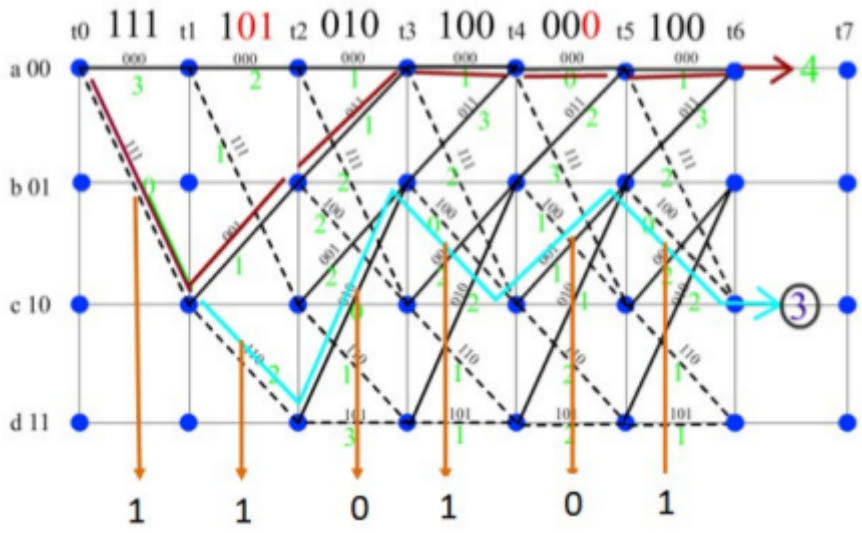
111 1**0**1 010 100 00**0** 100

La ruta con menor sumatoria de distancias de Hamming contendrá la información correcta que fue transmitida



Ruta Roja = 0 + 1 + 1 + 1 + 0 + 1 = 4
Ruta Celeste = 0 + 2 + 0 + 0 + 1 + 0 = 3

De esta manera de la ruta celeste se extrae la información real

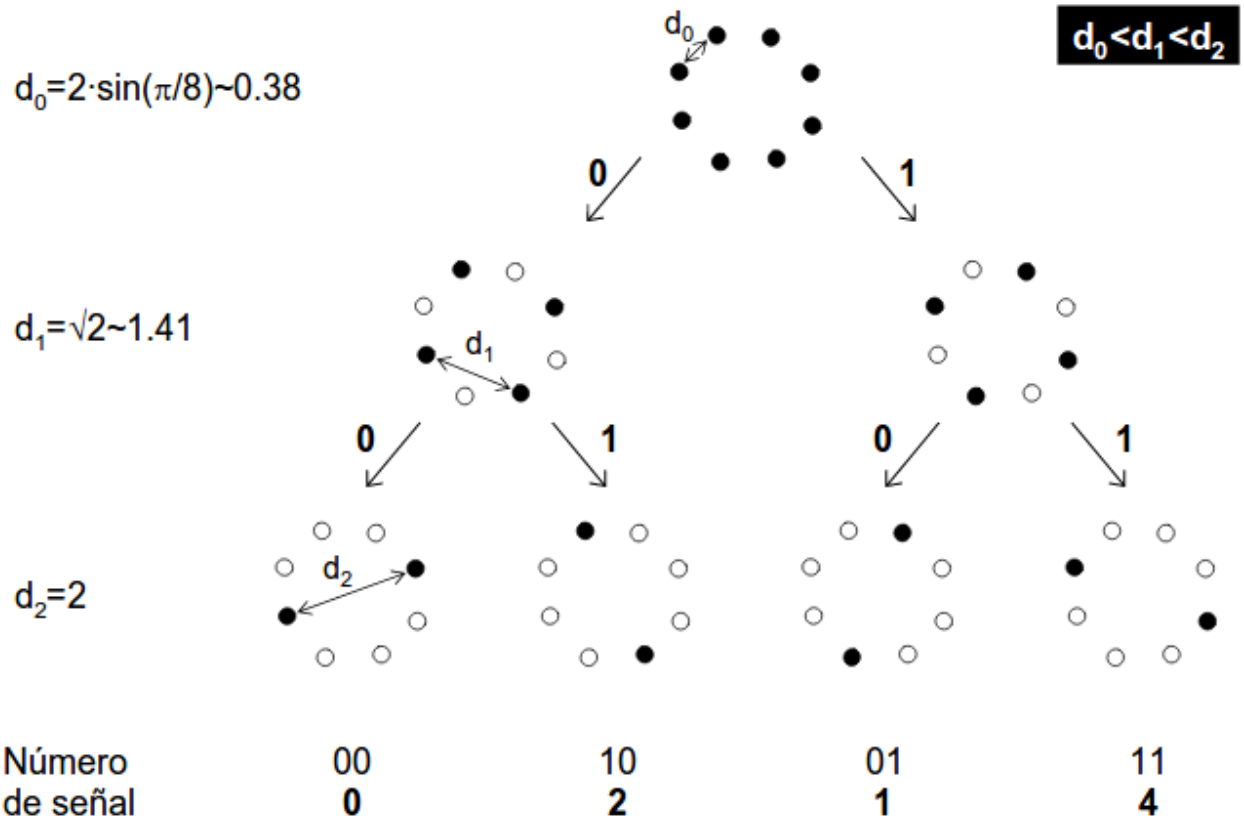


2) **Puncturing:**

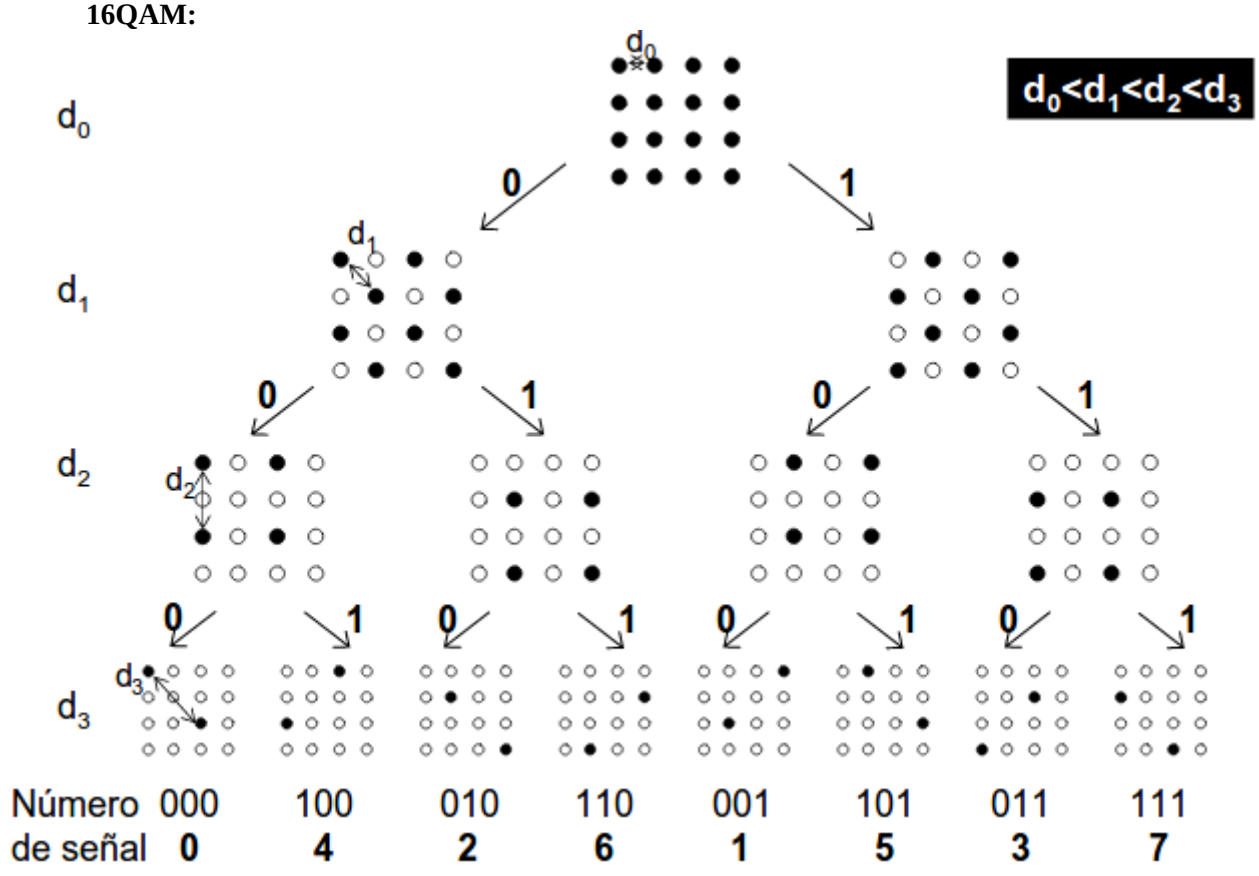
Es el proceso de remover algunos bits de paridad tras la codificación; esto tiene el mismo efecto que codificar a una mayor tasa o menor redundancia. Sin embargo, tras utilizar Puncturing, es posible utilizar el mismo decodificador sin importar cuántos bits hayan sido removidos, es decir, que se agrega flexibilidad al sistema sin incrementar demasiado su complejidad.

3) En el caso de sistemas multinivel, se debe replantear el particionamiento de asignación de estados para armar el Trellis. A continuación se presentan ejemplos del particionamiento

8PSK:



16QAM:


d0
d0<d1<d2<d3
d0
d1
d2
d3
0
1
Número 000 100 010 110 001 101 011 111
de señal 0 4 2 6 1 5 3 7