CAPÍTULO 3

EL ESQUEMA DE CODIFICACIÓN TAN BUSCADO: LOS TURBO-CÓDIGOS

3.1. Introducción.

Los Turbo-códigos fueron presentados a la comunidad científica en 1993 en la IEEE International Conference on Communications en Geneva, Suiza. Su invención fue el resultado de una construcción pragmática conducida por los franceses Claude Berrou y Alain Glavieux, quienes de acuerdo con [A.BERR] se basaron en las intuiciones de G. Battail, J. Hagenauer y P.Hoeher que a finales de los 80´s remarcaron el interés del procesamiento probabilístico en receptores de comunicación digitales.

Desde su invención, los Turbo-códigos han sido ampliamente estudiados y adoptados por varios sistemas de comunicación, permitiendo que ésta se lleve a cabo a tasas de transmisión cercanas a la capacidad del canal y con probabilidades de error muy bajas debido a que este esquema de codificación es capaz de operar a 0.1 dB de ese límite teórico descubierto por C. Shannon (ver Sección 3.4). Tasas de transmisión elevadas permiten la entrega de contenido multimedia y de otras formas densas de información a dispositivos portables. De acuerdo con [A.LEVI], la ganancia de codificación extra ofrecida por los Turbo-códigos también puede utilizarse en los dispositivos inalámbricos para reducir la fuerza de sus señales, lo que conlleva a que más dispositivos puedan compartir el mismo espectro de frecuencia al reducirse la interferencia *inter-dispositivo*. Así mismo, decrementar la fuerza de las señales también se ve reflejado en un decremento de los

requerimientos de potencia, lo cual es una consideración importante para los dispositivos inalámbricos alimentados por batería.

De acuerdo con [BAR04], las tres áreas clave en que los Turbo-códigos proveen un mejoramiento en el desempeño son: capacidad (los Turbo-códigos logran un desempeño cercano a los límites teóricos de la capacidad), eficiencia en el costo del sistema (un usuario es capaz de enviar la misma cantidad de información empleando únicamente la mitad del ancho de banda), y número de usuarios (un proveedor de servicio satelital es capaz de duplicar el número de usuarios sin incrementar la capacidad del satélite).

Los Turbo-códigos representan una clase de códigos convolucionales concatenados en paralelo [A.SALA]. Esto quiere decir que un Turbo-codificador codifica la misma información dos (o más) veces, pero de forma tal que la secuencia de información que ingresa a cada codificador convolucional independiente posea un orden distinto. Para lograr esto, la información a ser transmitida es almacenada en una memoria para ser permutada por medio de un (o más) dispositivo(s) conocido(s) como *interleaver*(*s*). De esta forma se producen secuencias no correlacionadas que son codificadas y transmitidas. De acuerdo con [BAR04] este concepto es la clave del desempeño tan bueno de los Turbo-códigos, y el tipo y tamaño del (de los) interleaver(s) utilizado(s) juegan un papel muy importante en el desempeño que puede alcanzarse.

La decodificación de los Turbo-códigos se basa en un procedimiento iterativo que permite que el proceso de decodificado no sea prohibitivamente complejo (y por consiguiente físicamente irrealizable) y que este esquema de codificación alcance niveles de desempeño tan elevados. Mediante este procedimiento, cada secuencia codificada recibida es decodificada de manera independiente por decodificadores distintos que intercambian (retroalimentan) sus decisiones *suaves* de decodificado de forma tal que el

proceso de Turbo-decodificación se refuerza de manera global, logrando tasas de error muy bajas. Dicho intercambio de decisiones suaves entre decodificadores requiere tanto que estos últimos sean del tipo *SISO* (*Soft-In-Soft-Out*) como de la utilización de algoritmos de decodificación *SISO* en el proceso de Turbo-decodificación.

Los Turbo-códigos deben visualizarse, más que como un nuevo esquema de codificación para control de errores, como una nueva forma de pensamiento en este campo de investigación, una forma de pensamiento que de acuerdo con [A.BERR] [BAR04] se basa en la aplicación de dos conceptos básicos y poderosos al control de errores: la *retroalimentación* y el concepto mundialmente válido de *divide y vencerás*.

3.2. El codificador Turbo.

Atendiendo a [A.LEVI], los Turbo-códigos codifican la información mediante una combinación de dos o más codificadores, junto con un cierto número de permutadores o *interleavers*. Los codificadores son típicamente codificadores convolucionales sistemáticos recursivos (RSC), que son fáciles de implementar dado que constan únicamente de un número relativamente pequeño de registros y sumadores módulo-2. La combinación de estos codificadores se conoce como *concatenación* y para codificadores convolucionales puede realizarse de las siguientes maneras [A.BARB]:

- *Concatenación Paralela de Códigos Convolucionales* (*PCCC*)
- *Concatenación Serial de Códigos Convolucionales* (*SCCC*)
- *Concatenación Híbrida de Códigos Convolucionales* (*HCCC*)

A continuación describiremos brevemente cada una de estas arquitecturas, pero en lo que reste de este Capítulo todo lo que se trate será con relación a la arquitectura PCCC,

ya que esta arquitectura representa la estructura típica de la codificación Turbo y la estructura empleada por la gran variedad de aplicaciones que adoptaron a los Turbo-códigos como parte de sus estándares, como los sistemas de comunicación inalámbrica UMTS y cdma2000.

3.2.1. Concatenación de códigos convolucionales.

3.2.1.1. Arquitectura PCCC.

Este tipo de arquitectura es definitivamente la más usada por los Turbo-códigos y recibe ese nombre debido a que la misma secuencia de información es codificada dos (o más) veces, en paralelo, utilizando las secuencias normal y permutada(s) de los bits de información [BAR04]. La siguiente Figura nos muestra el esquema general y básico de la arquitectura PCCC.

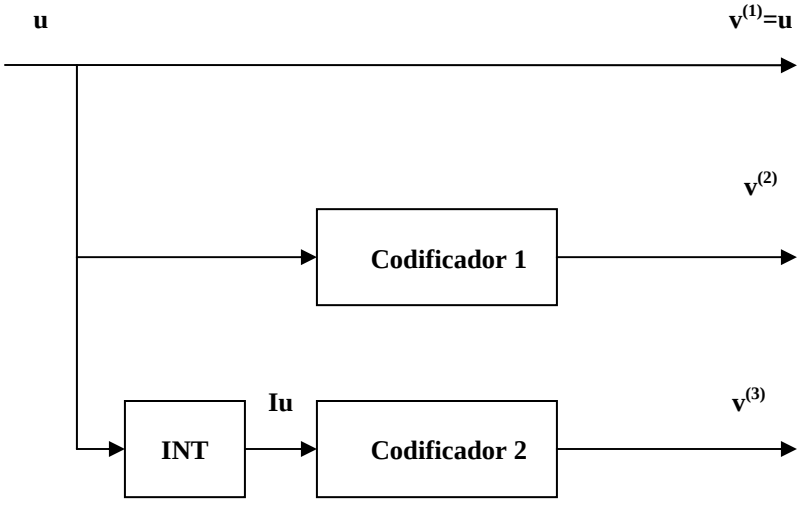

Figura 3.1. Arquitectura PCCC básica. [BAR04].

La Figura 3.1 nos muestra en realidad un Turbo-codificador que utiliza dos codificadores convolucionales (que típicamente son del tipo RSC), un Interleaver (**INT**) y

una conexión directa entre una de las salidas y la entrada del codificador (que lo hace un codificador sistemático). El Interleaver toma la secuencia de información **u** y la reordena en una manera pseudo-aleatoria, repetible y ajustada [A.LEVI] tal que si $\mathbf{u} = (u_1, \ldots, u_k)$ es un k-tuple binario, entonces **Iu** también será un k-tuple binario con la misma información de **u** pero ordenada en una forma completamente distinta. El *Codificador 1* toma la secuencia de información **u** y produce la secuencia de paridad $\mathbf{v}^{(2)}$, mientras que el *Codificador 2* toma la secuencia **Iu** y produce la secuencia de paridad $\mathbf{v}^{(3)}$. La secuencia de información $\mathbf{v}^{(1)} = \mathbf{u}$, junto con las secuencias de paridad $\mathbf{v}^{(2)}$ y $\mathbf{v}^{(3)}$ forman la salida del Turbo-codificador, que de acuerdo con [T.ING98] está dada por:

$$\mathbf{v} = (v_1^{(1)}, v_1^{(2)}, v_1^{(3)}, v_2^{(1)}, v_2^{(2)}, v_2^{(3)}, \ldots, v_k^{(1)}, v_k^{(2)}, v_k^{(3)}) \qquad (3.1)$$

Como se puede observar existen tres bits de salida por cada bit de entrada, por lo que decimos que el Turbo-código de la Figura 3.1 tiene una tasa de código $r = 1/3$. Es importante mencionar que los codificadores que forman parte de un Turbo-codificador no tienen que ser idénticos, aunque usualmente lo son.

Para lograr tasas de codificación más altas, se pude utilizar un elemento conocido como *Puncturer* en un Turbo-codificador. Este elemento generalmente consiste en un switch de multiplexado que reduce la cantidad de bits a ser transmitidos descartando algunos de los bits de paridad producidos por los codificadores del Turbo-codificador. De acuerdo con [A.LEVI], para estos casos el Turbo-decodificador asumirá que los bits descartados eran todos 1´s o todos 0´s. La Figura 3.2 nos muestra la implementación de un Turbo-código de tasa $r = 1/2$ que utiliza dos RSC´s y un puncturer, y cuya secuencia de salida está dada por: $\mathbf{v} = (v_1^{(1)}, v_1^{(2)}, v_2^{(1)}, v_2^{(3)}, v_3^{(1)}, v_3^{(2)}, v_4^{(1)}, v_4^{(3)}, \ldots)$. Nótese que los

RSC´s utilizados son de tasa r = 1/2 y que en cada uno la salida correspondiente a su secuencia de información se elimina para formar el Turbo-codificador (ver Sección 2.3.3).

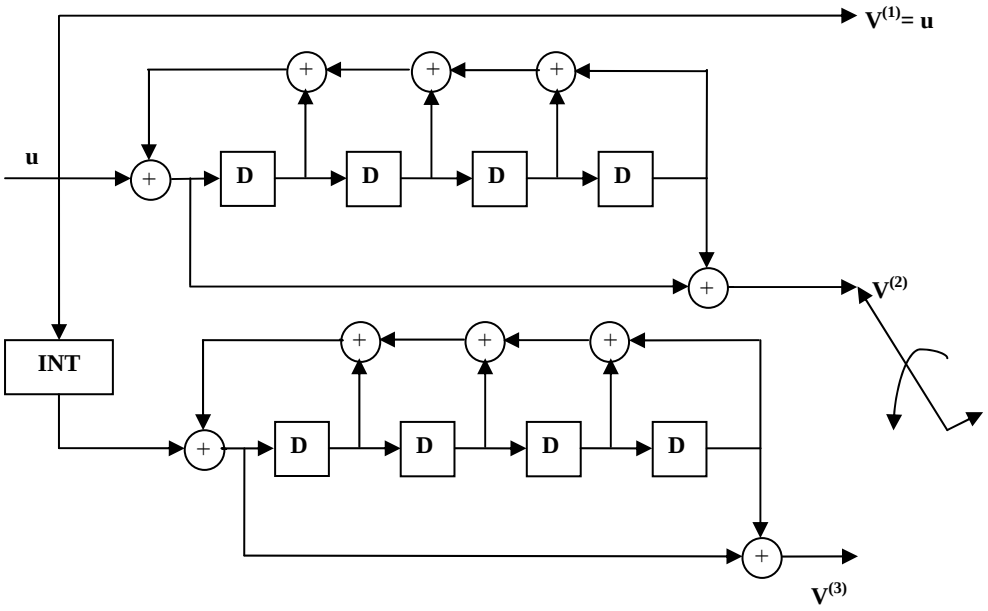

Figura 3.2. Turbo-codificador de tasa r = ½. [A.BARB].

De manera contraria, si lo que se desea es obtener tasas de codificación más bajas, entonces lo que puede hacerse es incrementar el número de interleavers y de codificadores en el Turbo-codificador para formar un *Turbo-código n-dimensional de baja tasa de codificación* como el que se muestra en la siguiente Figura:

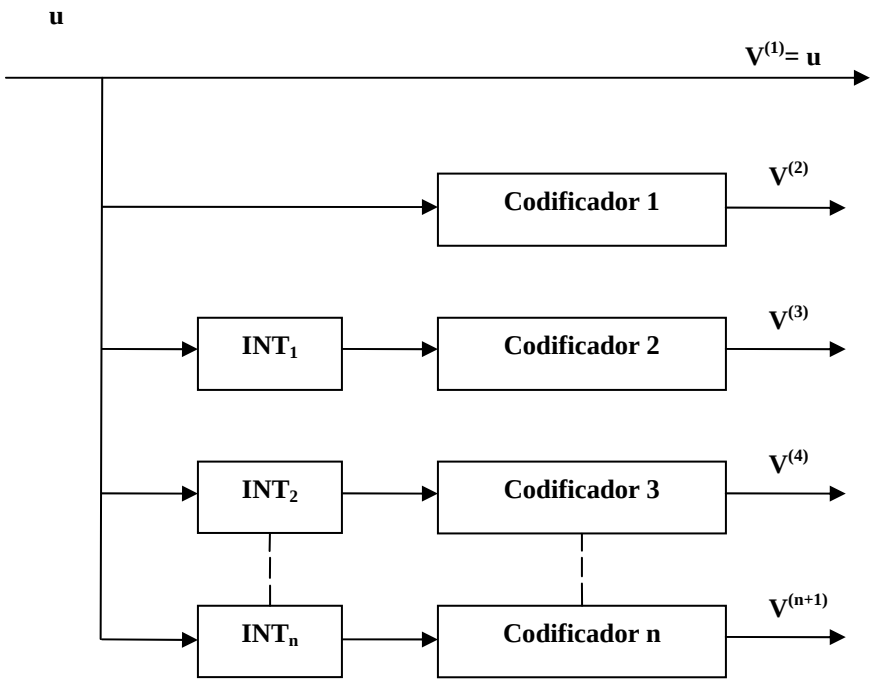

Figura 3.3. Turbo-codificador *n*-dimensional de tasa baja. [BAR04].

De acuerdo con [A.BARB] el primer sistema de comunicación comercial que utilizó Turbo-códigos fue diseñado en el Institute for Telecommunications Research (ITR) y se basó en una estructura PCCC como la mostrada en la Figura 3.2. Atendiendo a [BAR04] uno de los problemas que presenta la arquitectura PCCC consiste en que no resulta ser una solución óptima si se requiere lograr una BER muy baja (como p. ej. 10^{-10} para transmisiones de datos).

3.2.1.2. Arquitectura SCCC.

Una alternativa a la arquitectura PCCC la constituye la estructura SCCC. La siguiente Figura nos muestra un ejemplo de una arquitectura SCCC cuya tasa de codificación es $r = 1/3$.

Figura 3.4. Arquitectura SCCC de tasa *r = 1/3*. [A.BARB].

En la Figura 3.4 el *Codificador 1* se conoce como *codificador externo* y el *Codificador 2* como *codificador interno*. Es importante notar la presencia del interleaver *entre* ambos codificadores convolucionales, que provoca que este dispositivo produzca efectos distintos en el desempeño de las arquitecturas PCCC y SCCC. De acuerdo con [A.BARB], una característica de los esquemas PCCC es que presentan un mejor desempeño que las arquitecturas SCCC a bajas SNR´s (SNR = Signal-to-Noise Ratio); sin embargo, a SNR´s elevadas los esquemas SCCC superan en desempeño a las arquitecturas PCCC. El tamaño y diseño del interleaver definen el punto de la SNR en que el desempeño de una estructura PCCC deja de ser mejor que el de una estructura SCCC en particular.

3.2.1.3. Arquitectura HCCC.

Una alternativa más a la concatenación de códigos convolucionales la constituye la arquitectura HCCC. Esta estructura se conoce como *híbrida* debido a que es una combinación de las concatenaciones paralela y serial, como se puede apreciar en la Figura 3.5.

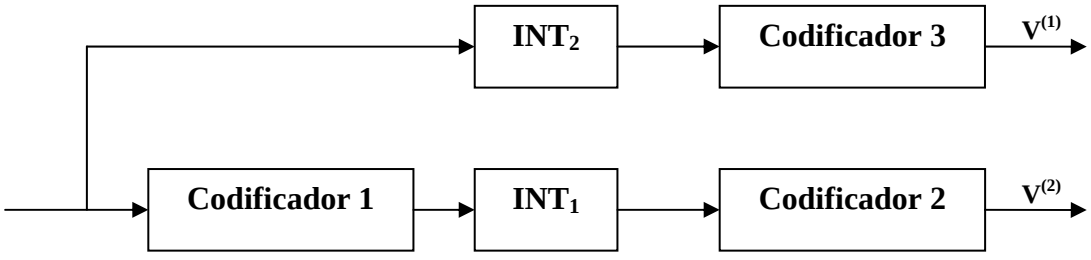

Figura 3.5. Arquitectura HCCC de tasa *r = ½*. [A.BARB].

Este tipo de arquitectura aún no ha sido estudiada con profundidad, sin embargo sí se sabe que constituye un mejoramiento definitivo de la estructura SCCC.

3.2.2. El Interleaver.

De acuerdo con [T.ING98] el interleaver es un dispositivo que permuta los bits de información en una manera predefinida, de forma tal que dos bits que están muy cercanos en la secuencia de información original queden lo más separados que se pueda en la secuencia de información permutada.

De acuerdo con [BAR04], el principal papel del interleaver en un proceso de Turbo-codificación consiste en tratar que los decodificadores hagan estimados no correlacionados de los valores "suaves" del mismo bit de información, ya que "entre menos correlacionados estén ambos estimados, mejor será la convergencia del algoritmo de decodificación iterativa" (Barbulescu, 2004: 52). Es por esto que el diseño y el tamaño del interleaver son factores clave que determinan el buen desempeño de un Turbo-código.

Atendiendo a [T.VAL99], el interleaver debe tener una estructura establecida y debido a esto trabaja con los bits de información por *bloques* y no de manera secuencial. Es por esto que los Turbo-códigos podrían considerarse códigos de bloque, y si el interleaver tiene un tamaño ajustado y los codificadores RSC de un Turbo-codificador están inicializados a cero, entonces estaríamos hablando de un código de bloque lineal. Una característica deseable de los códigos de bloque lineales es que su peso de Hamming (ver ecuación (2.4), Capítulo 2) o *distancia mínima* sea alta. La combinación del interleaving y la codificación RSC asegura que la mayoría de las palabras de código producidas por un

Turbo-codificador tengan un alto peso de Hamming, lo que permite que los Turbo-códigos ofrezcan un buen desempeño a bajas SNR´s.

De acuerdo con [T.ING98], los interleavers pueden dividirse en dos grupos: *interleavers de bloque* e *interlerleavers convolucionales*. Todo lo expresado en el párrafo anterior tiene que ver con interleavers de bloque. Los interleavers convolucionales son capaces de trabajar con los bits de información de manera secuencial, y son utilizados en una clase de Turbo-codificación conocida como *Stream-oriented Turbo-codes* que generalmente se emplea en aplicaciones que tienen que ver con conmutación de circuitos (como la Red de Telefonía Pública Conmutada - PSTN). En esta Tesis únicamente nos enfocaremos a la Turbo-codificación "clásica" y por lo tanto sólo trataremos con interleavers de bloque.

A continuación exponemos algunos de los interleavers de bloque más utilizados en la Turbo-codificación, siguiendo una presentación similar a la ofrecida por [A.BARB][BAR04].

3.2.2.1. Interleaver "Renglón-Columna".

Este es el interleaver más simple que se conoce y consiste en una memoria en la que la información es escrita a lo largo de los renglones y leída a lo largo de las columnas [A.BARB]. La siguiente tabla nos ilustra un ejemplo de la memoria de un interleaver de este tipo.

Tabla 3.1. Ejemplo de la memoria de un interleaver "Renglón-Columna" [A.BARB].

X_1	X_2	X_3
X_4	X_5	X_6
X_7	X_8	X_9
X_{10}	X_{11}	X_{12}
X_{13}	X_{14}	X_{15}
X_{16}	X_{17}	X_{18}
X_{19}	X_{20}	X_{21}

De acuerdo con su definición, la salida de este interleaver estará dada por: X_1, X_4, X_7, X_{10}, X_{13}, X_{16}, X_{19}, X_2, X_5, X_8, X_{11}, . . .

3.2.2.2. Interleaver "Espiral".

Un interleaver de este tipo consiste en una memoria en que la información es escrita como en la Tabla 3.1 pero leída en forma diagonal [A.BARB]. De acuerdo con su definición, la salida de este interleaver estará dada por: X_{19}, X_{17}, X_{15}, X_{10}, X_8, X_6, X_1, X_{20}, X_{18}, X_{13}, ...

3.2.2.3. Interleaver "Impar-Par".

De acuerdo con [BAR04] este tipo de interleaver ofrece un mejoramiento significativo cuando se utiliza en el diseño de un Turbo-codificador de tasa $r = 1/2$ como el mostrado en la Figura 3.2.

Por ejemplo, supongamos que tenemos una secuencia binaria de información que entra al RSC superior de la Figura 3.2 y que sólo almacenamos los bits codificados que estén en las posiciones impares. La siguiente tabla resulta:

Tabla 3.2. Bits codificados en las posiciones impares [A.BARB].

X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}
Y_1	-	Y_3	-	Y_5	-	Y_7	-	Y_9	-	Y_{11}	-	Y_{13}	-	Y_{15}

Un interleaver "Impar-par" es un interleaver "Renglón-Columna" cuya memoria tiene un número impar de renglones y un número impar de columnas [A.BARB]. Para nuestra secuencia de información la memoria del interleaver quedaría escrita de acuerdo a la siguiente tabla:

Tabla 3.3. Ejemplo de la memoria de un interleaver "Impar-Par" 3 x 5 [A.BARB].

X_1	X_2	X_3	X_4	X_5
X_6	X_7	X_8	X_9	X_{10}
X_{11}	X_{12}	X_{13}	X_{14}	X_{15}

Suponiendo que la secuencia de salida de este interleaver es codificada por el RSC inferior de la Figura 3.2 y que almacenamos los bits codificados que estén en las posiciones pares, la siguiente Tabla resulta:

Tabla 3.4. Bits codificados en las posiciones pares [A.BARB].

X_1	X_6	X_{11}	X_2	X_7	X_{12}	X_3	X_8	X_{13}	X_4	X_9	X_{14}	X_5	X_{10}	X_{15}
-	Z_6	-	Z_2	-	Z_{12}	-	Z_8	-	Z_4	-	Z_{14}	-	Z_{10}	-

Como podremos observar a partir de las Tablas 3.2 y 3.4, los 15 bits de información tienen un bit codificado asociado a cada uno de ellos, lo cual se traduce en que la potencia de codificación está uniformemente distribuida. Las dos salidas $\mathbf{v}^{(1)}$ y $\mathbf{v}^{(2)}$-$\mathbf{v}^{(3)}$ del Turbo-codificador considerado estarán dadas por la siguiente tabla:

Tabla 3.5. Salidas del Turbo-codificador de la Figura 3.2 considerando la utilización de un interleaver "Impar-Par" 3 x 5.

$\mathbf{v^{(1)}}$	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14	X15
$\mathbf{v^{(2)}}$-$\mathbf{v^{(3)}}$	Y1	Z6	Y3	Z2	Y5	Z12	Y7	Z8	Y9	Z4	Y11	Z14	Y13	Z10	Y15

3.2.2.4. Interleaver "Pseudo-aleatorio".

Un interleaver "Pseudo-aleatorio", de acuerdo con [T.ING98], es aquel en que la secuencia de información es escrita en su memoria en un orden pseudo-aleatorio y posteriormente es leída de la memoria columna por columna. Atendiendo a [A.BARB], los interleavers de este tipo son definidos por un generador pseudo-aleatorio de números donde todos los enteros entre 1 y N (el tamaño del interleaver) pueden ser generados.

3.2.2.5. Interleaver de 3G.

De acuerdo con [BAR04] el proceso de implementación del interleaver empleado en 3G (en particular en el sistema UMTS) consiste de dos pasos: generación del interleaver madre y descarte de bits. Existen 207 conjuntos de interleavers madre que pueden seleccionarse de acuerdo con el tamaño del bloque de información. Una vez que se selecciona un conjunto en particular, algunos bits son descartados con la finalidad de ajustar el tamaño al tamaño original del bloque de información.

La generación del interleaver madre se consigue en tres pasos [BAR04]: la secuencia de entrada es almacenada a lo largo de los renglones en una matriz rectangular; a continuación se efectúa una permutación inter-fila (entre renglones); y finalmente se realiza una permutación intra-fila (en cada renglón). El número de filas y de columnas de la matriz

es una función del tamaño del bloque de información que a su vez puede variar entre 40 y 5114 bits. Atendiendo a [BAR04], existen tablas con raíces primitivas que se utilizan para las permutaciones inter e intra-fila.

En el siguiente Capítulo de esta Tesis hablaremos con mayor detalle sobre este interleaver.

3.2.2.6. El mejor interleaver...

Debido a que los interleavers son diseñados con base en los requerimientos específicos del sistema en que se empleen, podemos decir que no existe una fórmula o regla universal que permita diseñar un "mejor interleaver". Por ejemplo, en [A.BARB] se comenta que para tamaños pequeños de bloque y una relación E_b/N_0 baja, un interleaver "Impar-par" ofrece un mucho mejor desempeño que un interleaver "Pseudo-aleatorio" y viceversa (para una relación E_b/N_0 alta).

3.3. La decodificación Turbo.

A partir de la Teoría de Codificación sabemos que entre mayor sea la longitud de una palabra de código, para el caso de códigos de bloque, o el tamaño de la memoria en un codificador convolucional, podemos conseguir mayores ganancias de codificación a bajas tasas de error de bit (BER´s) para un tipo de código en particular (teorema de codificación del canal de Shannon). Este teorema consideraba que la estrategia de decodificado para poder lograr esto era la decodificación de máxima probabilidad. Sin embargo como vimos en la sección 2.3.2. de esta Tesis, a pesar de ser en general una estrategia óptima, este tipo de decodificación presenta la gran problemática de que su complejidad se incrementa

exponencialmente con la longitud de las palabras de código (o el incremento de la memoria del codificador para el caso de códigos convolucionales) y por lo tanto se vuelve un esquema de decodificación impráctico si se busca alcanzar el desempeño óptimo de un código con ayuda de éste.

A través de los años y hasta antes del nacimiento de los Turbo-códigos, conforme los expertos en el campo de la Teoría de Codificación trataban de mejorar los esquemas de codificación existentes y de crear otros nuevos, el precio por conseguir buenas ganancias de codificación a tasas de error bajas permanecía inalterable: una gran complejidad en el proceso de decodificación.

Esto llevó a la necesidad de buscar nuevos esquemas de codificación con los que se pudieran reemplazar los algoritmos de decodificación de máxima probabilidad (*MLDA*, por sus siglas en inglés de *Maximum Likelihood Decoding Algorithms*) por estrategias de decodificado más simples. Esta búsqueda dominaría el campo de la codificación para control de errores por casi medio siglo...

Fue hasta 1993, con la aparición de los Turbo-códigos, que cesaría la búsqueda por encontrar un esquema de codificación que fuera capaz de alcanzar los límites de la capacidad formulados por Shannon al poder procesar en una manera no compleja palabras de código tan grandes como para poder vencer los efectos indeseables del ruido presente en el canal de comunicación.

Los Turbo-códigos son capaces de lograr ganancias de codificación nunca antes logradas por ningún otro esquema de codificación del canal a tasas de error muy bajas. Esto se debe principalmente a la introducción del interleaver a la arquitectura del Turbo-codificador (que permite una generación de palabras de código en cierta forma aleatorias y por ende más "resistentes" al ruido) y a la estrategia de decodificado empleada por este

esquema de codificación. Esta estrategia combina códigos simples de forma tal que cada uno de estos códigos puede decodificarse de manera independiente por medio de decodificadores SISO (Soft-In-Soft-Out) no complejos que pueden emplear un algoritmo de decodificación de máxima probabilidad. De acuerdo con [BAR04], "utilizando decodificadores SISO la información puede ser pasada de un decodificador al siguiente en un modo iterativo. Esta es una estrategia de "divide y vencerás" que en un proceso iterativo bien puede aproximarse al desempeño de los MLDA" (Barbulescu, 2004: 18).

Como ejemplo, en un Turbo-codificador con una memoria total de 1000 células, 992 representan la memoria empleada para el proceso de interleaving y las restantes 8 la memoria empleada por los codificadores RSC que componen al Turbo-codificador [BAR04]. De esta manera, el número de estados posibles en el trellis de cada codificador RSC es muy pequeño y por lo tanto cada decodificador individual correspondiente en el Turbo-decodificador tendrá una tarea simple de llevar a cabo al tener que encontrar la trayectoria de máxima probabilidad a través de un trellis no tan complejo. Sin embargo, nótese que el tamaño del interleaver debe ser grande para que un Turbo-código alcance un desempeño cercano a los límites teóricos de Shannon empleando codificadores y decodificadores simples en el Turbo-codificador y Turbo-decodificador, respectivamente.

El proceso iterativo, pues, "demostró ser la respuesta clave de ingeniería para lograr un desempeño muy cercano a la solución óptima con una complejidad en la implementación no prohibitiva" (Barbulescu, 2004: 24).

3.3.1. El decodificador Turbo y el proceso de decodificación iterativa.

Consideremos la estructura general del Turbo-decodificador iterativo que se muestra en la siguiente Figura.

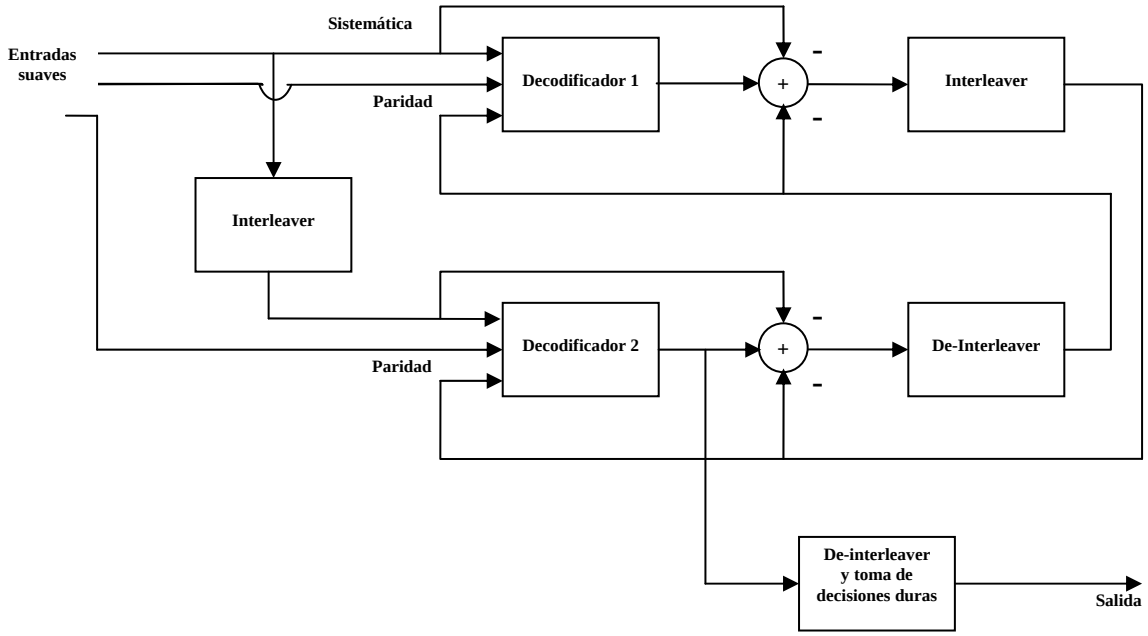

Figura 3.6. Esquema general de un Turbo-decodificador. [A. WOOD].

Este Turbo-decodificador corresponde a un Turbo-codificador de tasa $r = 1/3$ (ó $r = 1/2$ si se utiliza un puncturer) como el mostrado en la Figura 3.1 (ó en la Figura 3.2), y representa el tipo de decodificador iterativo más utilizado para la decodificación de Turbo-códigos.

Como se podrá observar, cada decodificador tiene tres entradas: los bits de salida del canal codificados sistemáticamente, los bits de paridad generados por el codificador correspondiente en el Turbo-codificador, y la información proveniente del otro decodificador acerca de los valores probables de los bits decodificados. Esta información proveniente del otro decodificador se conoce como *información a-priori* [A.WOOD].

Ambos decodificadores componentes del Turbo-decodificador deben explotar tanto las entradas provenientes del canal como la información a-priori proporcionada a cada uno de ellos. Estos decodificadores deben generar salidas *suaves* para los bits decodificados, lo que significa que no sólo deben proveer la secuencia de bits decodificados sino también la probabilidad asociada a cada bit de que éste haya sido decodificado correctamente. De acuerdo con [A.WOOD], estas salidas suaves se representan generalmente en términos de las llamadas *relaciones Log-likelihood* (*LLR´s*, por sus siglas en inglés de *Log-Likelihood Ratios*) cuyas magnitudes y amplitudes dan información sobre el signo de cada bit y la probabilidad de una decisión correcta, respectivamente. La LLR para el valor de un bit decodificado u_k está dada por:

$$L(u_k) = \ln\left(\frac{P(u_k = +1)}{P(u_k = -1)}\right) \tag{3.1}$$

donde $P(u_k = +1)$ es la probabilidad de que el bit $u_k = +1$ (1 binario) y $P(u_k = -1)$ es la probabilidad de que el bit $u_k = -1$ (0 binario). Una LLR positiva y alta indicará una probabilidad alta de que el bit decodificado sea un 1, mientras que una LLR negativa y alta indicará una probabilidad alta de que el bit decodificado sea un 0.

En la Figura 3.6 podemos ver que las entradas al Turbo-decodificador provenientes del canal también deben ser suaves. Esto implica que en el sistema de comunicación deberá utilizarse un demodulador de decisiones suaves, de salida no-cuantizada, y que previo al Turbo-decodificador deberá existir un *estimador de la LLR de cada bit,* cuya función será precisamente la de calcular las relaciones log-likelihood para cada bit recibido de la señal demodulada.

De este modo, los decodificadores del Turbo-decodificador deberán ser capaces tanto de generar como de recibir salidas y entradas suaves, respectivamente. Para que esto se cumpla, es necesaria la utilización de algoritmos de decodificación SISO. De acuerdo con [BAR04], existen dos categorías importantes de algoritmos de decodificación SISO: los algoritmos que minimizan la probabilidad de error de símbolo y los algoritmos de decodificación de máxima probabilidad, que minimizan la probabilidad de error de palabra o secuencia. A la primer categoría pertenece el algoritmo *MAP* (*Maximum A Posteriori*) y a la segunda el algoritmo *SOVA* (*Soft Output Viterbi Algorithm*), ambos algoritmos muy utilizados en Turbo-decodificación y de los cuales hablaremos más adelante.

Antes de describir el proceso de decodificación iterativa del Turbo-decodificador de la Figura 3.6 es necesario que expongamos algunos conceptos de gran importancia dentro de este contexto. Para poder describir de manera clara y precisa dicho proceso consideraremos que los decodificadores componentes del Turbo-decodificador de la Figura 3.6 corresponden a codificadores RSC en el Turbo-codificador y que dichos decodificadores emplean el algoritmo de decodificación SISO MAP, las cuales corresponden a las características propias de un típico esquema de codificación Turbo.

El algoritmo MAP entrega para cada bit decodificado u_k la probabilidad de que éste sea +1 o –1, dada la secuencia $\underline{y}$ de símbolos recibidos del canal. Esto es equivalente a encontrar la *LLR a-posteriori* $L(u_k \mid \underline{y})$, dada por [A.WOOD]:

$$L(u_k \mid \underline{y}) = \ln\left(\frac{P\left\langle u_k = +1 \middle| \underline{y} \right\rangle}{P\left\langle u_k = -1 \middle| \underline{y} \right\rangle}\right) \tag{3.2}$$

De acuerdo con [A.WOOD] para un código sistemático, como un código RSC, la salida de un decodificador MAP puede escribirse como:

$$L(u_k \mid \underline{y}) = L(u_k) + L_c y_{ks} + L_e(u_k) \tag{3.3}$$

donde $L(u_k)$ es la *LLR a-priori* del bit decodificado u_k, $L_c y_{ks}$ es la entrada suave al Turbo-decodificador (proveniente del canal) correspondiente al bit sistemático transmitido u_k (y_{ks} es la versión ruidosa recibida del bit sistemático transmitido u_k y el factor L_c es una medida conocida como *medida de confiabilidad del canal*), y $L_e(u_k)$ es la *LLR extrínseca* o *información extrínseca* para el bit u_k.

Hasta este punto hemos mencionado tres conceptos cuya definición es conveniente dar aquí de manera formal, basándonos en [A.WOOD]:

- ***LLR o información a-priori de un bit.-*** es la información que se conoce del bit antes de que comience el proceso de decodificado, proveniente, en forma ideal, de un "conducto de información" independiente de la secuencia $\underline{y}$ de símbolos recibidos del canal por un decodificador. También se conoce como *información intrínseca*.
- ***LLR o información extrínseca de un bit u_k.-*** es la información que provee un decodificador basándose en la secuencia $\underline{y}$ de símbolos recibidos del canal y en la secuencia de LLR´s a-priori proveniente de otro decodificador, *excluyendo* el bit sistemático recibido y_{ks} y la información a-priori del bit u_k, $L(u_k)$. Típicamente, un decodificador provee esta información utilizando las restricciones impuestas a la secuencia transmitida por el código usado. La información extrínseca para un decodificador MAP puede obtenerse restando los términos $L(u_k)$ y $L_c y_{ks}$ de la salida suave del decodificador $L(u_k \mid \underline{y})$, dada por la ecuación (3.3).

- ***LLR o información a-posteriori de un bit.-*** es la información que provee un decodificador basándose en *todas* las fuentes disponibles de información acerca de ese bit. En otras palabras, es la salida del algoritmo MAP $L(u_k \mid \underline{y})$.

Para describir cómo es que el proceso de decodificación iterativa de los Turbo-códigos se lleva a cabo nos basaremos en [A.BARB] [A.WOOD] [BAR04]. El proceso comienza en el Decodificador 1 de la Figura 3.6, con la primera iteración. Este decodificador recibe la secuencia del canal $L_c\underline{y}^{(1)}$ que contiene las versiones suaves recibidas de los bits sistemáticos transmitidos ($L_c y_{ks}$) y de los bits de paridad generados por el codificador correspondiente en el Turbo-codificador ($L_c y_{kl}$). Si en el Turbo-codificador se utilizó un puncturer para lograr una tasa de codificación más alta, el Turbo-decodificador deberá insertar ceros en las salidas suaves del canal $L_c y_{kl}$ que correspondan a los bits eliminados por el puncturer. En esta primera iteración el Decodificador 1 no cuenta con información a-priori acerca de los bits decodificados.

Una vez hecho esto, el Decodificador 1 procesa las entradas suaves provenientes del canal y produce su estimado $L_{11}(u_k \mid \underline{y})$ de los LLR´s condicionales de los bits de información u_k para $k = 1, 2, \ldots, N$. En la notación utilizada, el subíndice 11 en $L_{11}(u_k \mid \underline{y})$ indica que este es el LLR a-posteriori producido en la primera iteración por el Decodificador 1.

Seguido de esto, el Decodificador 2 de la Figura 3.6 entra en operación al recibir la secuencia del canal $L_c\underline{y}^{(2)}$ que contiene la versión suave de los bits de paridad generados por el codificador correspondiente en el Turbo-codificador y la versión permutada (pasada por el interleaver) de la versión suave de los bits sistemáticos recibidos. Nuevamente el Turbo-decodificador deberá insertar algunos ceros en las salidas suaves del canal correspondientes

a los bits de paridad recibidos en caso de que se haya utilizado un puncturer en el Turbo-codificador.

Sin embargo, la secuencia $L_c\underline{y}^{(2)}$ no es la única secuencia recibida por el Decodificador 2. Una secuencia de LLR´s a-priori $L(u_k)$ también es recibida por este decodificador en la primera iteración del proceso de decodificación iterativa. En un Turbo-decodificador, la versión permutada de los LLR´s extrínsecos $L_e(u_k)$ extraídos de los LLR´s condicionales $L_{11}(u_k|\underline{y})$ (los restadores mostrados en el Turbo-decodificador de la Figura 3.6 son debidos a este hecho) constituye dicha secuencia de LLR´s a-priori. Nótese que estamos hablando de una versión *permutada* de LLR´s extrínsecos debido a que el Decodificador 2 debe recibir los LLR´s a-priori en el mismo orden en que los bits de información fueron codificados por el codificador correspondiente a este decodificador.

Pero, ¿por qué tienen que ser los LLR´s extrínsecos $L_e(u_k)$ y no los LLR´s a-posteriori generados por un decodificador los que se conviertan en LLR´s a-priori para el otro decodificador?. De acuerdo con [A.WOOD][T.SEG95] esto se debe a que, como ya se mencionó anteriormente, en un proceso de decodificación *iterativo* lo ideal sería que la información a-priori utilizada por los decodificadores proviniera de una fuente de información independiente de la secuencia $\underline{y}$ de símbolos recibidos por los mismos. Sin embargo, como ya se describió anteriormente, los LLR´s extrínsecos $L_e(u_k)$ generados por un decodificador (y utilizados por el otro como información a-priori) utilizan la secuencia $\underline{y}$ de símbolos recibidos del canal excluyendo al bit sistemático recibido y_{ks}; dado que en esta secuencia están incluidos los bits sistemáticos transmitidos, podemos decir que la información a-priori utilizada por cada decodificador dentro del Turbo-decodificador no es realmente independiente de la secuencia $\underline{y}$ de entrada a cada uno de ellos, puesto que dichos

bits son empleados por ambos. Gracias a la combinación, sin embargo, de la utilización de un Interleaver en el proceso de Turbo-codificación junto con la utilización del concepto de información extrínseca, se consigue que la información a-priori empleada por los decodificadores componentes del Turbo-decodificador esté muy poco correlacionada con la secuencia $\underline{y}$ de entrada a cada uno de ellos, haciendo de la decodificación iterativa una estrategia de decodificación cuasi-óptima. Es por esto que el diseño del Interleaver es un factor clave que determina el buen desempeño de un Turbo-código, como se mencionó en la sección 3.2.2.

De este modo, el Decodificador 2 utiliza la secuencia del canal recibida $L_c\underline{y}^{(2)}$ y la secuencia de LLR´s a-priori $L(u_k)$ para producir sus LLR´s a-posteriori $L_{12}(u_k|\underline{y})$. Así termina la primera iteración del proceso de Turbo-decodificación.

Para la segunda iteración el Decodificador 1 nuevamente procesa su secuencia recibida del canal $L_c\underline{y}^{(1)}$, pero ahora también cuenta con una secuencia de LLR´s a-priori $L(u_k)$ suministrada por la parte extrínseca $L_e(u_k)$ de los LLR´s a-posteriori $L_{12}(u_k|\underline{y})$ calculados por el Decodificador 2, y por lo tanto genera un LLR a-posteriori $L_{21}(u_k|\underline{y})$ mejorado para cada bit de información decodificado. Nótese, a partir de la Figura 3.6, que la secuencia de LLR´s a-priori $L(u_k)$ recibida por el Decodificador 1 se encuentra ordenada en la misma forma en que los bits de información fueron codificados por el codificador correspondiente a este decodificador gracias al De-interleaver, cuya función es opuesta a la del interleaver.

La segunda iteración continúa, entonces, con el Decodificador 2 que utiliza como información a-priori la versión permutada de la parte extrínseca de los LLR´s a-posteriori $L_{21}(u_k|\underline{y})$ para calcular, junto con su secuencia recibida del canal $L_c\underline{y}^{(2)}$, sus LLR´s

a-posteriori $L_{22}(u_k | \underline{y})$ mejorados. De este modo termina la segunda iteración del proceso de decodificación iterativa.

De la misma manera descrita hasta este momento continúa el proceso de Turbo-decodificación, decrementándose con cada iteración nueva la BER de los bits decodificados. Sin embargo, la ganancia de codificación de una iteración a otra se decrementa con el número de iteraciones. Por esta razón y por motivos de complejidad en la implementación del Turbo-decodificador, generalmente sólo se efectúan de 6 a 8 iteraciones en un proceso de decodificación Turbo [A.WOOD]. La complejidad total de un Turbo-decodificador dependerá de qué tan eficientemente esté implementado el algoritmo de decodificación [BAR04].

El proceso de Turbo-decodificación finaliza con una secuencia de LLR´s a-posteriori $L_{i2}(u_k | \underline{y})$ a la salida del Decodificador 2 de la Figura 3.6, donde el subíndice *i* en $L_{i2}(u_k | \underline{y})$ representa el número total de iteraciones efectuadas en dicho proceso. Estos LLR´s son convertidos en 1´s y 0´s mediante una toma de decisiones duras (que se basa en las magnitudes y amplitudes de los LLR´s) y posteriormente son pasados a través de un De-interleaver para proporcionar los bits decodificados en el orden correcto a la salida del Turbo-decodificador.

3.3.2. Algoritmos de decodificación SISO.

En 1974 el algoritmo *MAP* (*Maximum A Posteriori*) fue propuesto por L. R. Bahl, J. Cocke, F. Jelinek y J. Raviv en [A.BAHL] para estimar las probabilidades a-posteriori de los estados y las transiciones de una fuente Markov observada en ruido sin memoria

[A.WOOD]. Debido a las iniciales de los apellidos de sus descubridores, este algoritmo también se conoce como *algoritmo BCJR*.

Bahl, Cocke, Jelinek y Raviv mostraron la forma en que este algoritmo podía ser utilizado para decodificar tanto códigos de bloque como códigos convolucionales. Mostraron que si se utiliza para decodificar códigos convolucionales, el algoritmo es óptimo en términos de minimizar la BER, lo cual es distinto a lo que hace el algoritmo de decodificación de Viterbi, que minimiza la probabilidad de que el decodificador seleccione una trayectoria incorrecta a través del trellis del código convolucional que esté en consideración. Sin embargo este hecho implica que el algoritmo BCJR tenga que examinar cada trayectoria posible a través del trellis del decodificador convolucional y por lo tanto resulta ser un algoritmo complejo. Debido a su complejidad y al hecho de que su desempeño resultó ser casi idéntico al del algoritmo de Viterbi para decodificación de salidas duras, el algoritmo MAP casi no se utilizó antes del descubrimiento de los Turbo-códigos.

Sin embargo el algoritmo MAP no sólo provee la secuencia de bits estimada, sino también las probabilidades para cada bit de que haya sido decodificado correctamente. Esto es esencial para el proceso de decodificación iterativa para Turbo-códigos y por lo tanto, a raíz de la invención de este poderoso esquema de codificación, el algoritmo MAP se convirtió en una gran opción para ser utilizado en la decodificación de salidas suaves de los códigos convolucionales componentes de los Turbo-códigos. Desde entonces se ha invertido un gran esfuerzo en la investigación para reducir la complejidad de este algoritmo a un nivel razonable, y hasta ahora sólo se han logrado desarrollar dos algoritmos que ofrecen un desempeño casi idéntico al ofrecido por MAP pero a una complejidad menor: el algoritmo *Max-Log-MAP* y el algoritmo *Log-MAP*.

De acuerdo con [T.VAL99], la gran complejidad del algoritmo MAP se debe principalmente a que debe efectuar una enorme cantidad de multiplicaciones y adiciones por cada estimado que produce de la probabilidad a-posteriori de cada bit de información decodificado. Este problema se puede aminorar si el algoritmo entero se lleva a cabo en el dominio logarítmico, en vez de esperar hasta el último paso para tomar el logaritmo de la LLR. El principal beneficio de ejecutar el algoritmo en el dominio logarítmico consiste en que la multiplicación se convierte en suma. Esta es la idea principal en la que se basan los algoritmos Max-Log-MAP y Log-MAP para reducir la complejidad del algoritmo MAP.

La diferencia entre los algoritmos Max-Log-MAP y Log-MAP radica principalmente en la forma en que efectúan la operación de suma en el dominio logarítmico. Para ilustrar la forma en que se efectúa una operación de suma en el dominio logarítmico consideremos el logaritmo Jacobiano, que de acuerdo con [T.VAL99] está dado por:

$$\begin{aligned} \ln (e^{x} + e^{y}) &= \max (x, y) + \ln (1 + \exp[- | y - x |]) \\ &= \max (x, y) + f_c(| y - x |) \end{aligned} \tag{3.4}$$

Como podemos observar, la suma en el dominio logarítmico se convierte en una operación de maximización seguida por una función de corrección f_c [T.VAL99]. Una aproximación a la ecuación (3.4) está dada por:

$$\ln (e^{x} + e^{y}) \approx \max (x, y) \tag{3.5}$$

En el dominio logarítmico, el algoritmo Max-Log-MAP efectúa la operación de suma como en la ecuación (3.5), mientras que el algoritmo Log-MAP efectúa dicha operación como en la ecuación (3.4). Esto nos permite decir que, de acuerdo con [A.WOOD], el desempeño del algoritmo Max-Log-MAP en comparación con el del algoritmo MAP es sub-óptimo debido a que utiliza una aproximación para efectuar las

operaciones de suma en el dominio logarítmico. El algoritmo Log-MAP fue propuesto para corregir dicha aproximación y su desempeño es idéntico al del algoritmo MAP. Sin embargo tal como se mencionó anteriormente, tanto el algoritmo Log-MAP como el Max-Log-MAP reducen en buena medida la complejidad del algoritmo MAP.

Atendiendo a [A.WOOD], el algoritmo Max-Log-MAP fue propuesto por W. Koch y A. Baier en [A.KOCH] en 1990. El algoritmo Log-MAP lo propusieron P. Robertson, E. Villebrun y P. Hoeher en [A.ROBE] en el año de 1995.

Otra gran opción para ser utilizado en la decodificación de salidas suaves de los códigos convolucionales componentes de los Turbo-códigos la constituye el algoritmo *SOVA* (*Soft Output Viterbi Algorithm*). A pesar de que el algoritmo de Viterbi (ver sección 2.3.2.1.) puede aceptar entradas suaves, únicamente es capaz de producir salidas duras y por lo tanto es inadecuado para Turbo-decodificación. De acuerdo con [T.VAL99], en 1989 J. Hagenauer y P. Hoeher propusieron una modificación al algoritmo de Viterbi en [A.HAGE] que produce las probabilidades a-posteriori de las transiciones de estado, o equivalentemente la confiabilidad de los estimados de bit. Este algoritmo se conoce como SOVA y de acuerdo con [A.WOOD] tiene dos modificaciones con respecto al algoritmo de Viterbi clásico que lo hacen adecuado para ser utilizado en decodificación Turbo: la primera consiste en que las métricas de trayectoria usadas son modificadas para tomar en cuenta información a-priori cuando se selecciona la trayectoria de máxima probabilidad a través del trellis, y la segunda radica en que el algoritmo provee una salida suave en forma del LLR a-posteriori $L(u_k \mid \underline{y})$ para cada bit decodificado.

Este algoritmo es el menos complejo de los algoritmos de decodificación SISO mencionados hasta este momento, sin embargo también es el menos preciso de estos

algoritmos. De acuerdo con [A.WOOD], cuando el algoritmo SOVA se utiliza en un Turbo-decodificador su complejidad es aproximadamente la mitad de la complejidad del algoritmo Max-Log-MAP, pero su desempeño es cerca de 0.6 dB peor que el desempeño que ofrece el algoritmo MAP.

En general la complejidad relativa de los algoritmos de decodificación SISO depende de la constraint length de los códigos convolucionales utilizados en el proceso de codificación Turbo. La complejidad y el desempeño de estos algoritmos ascienden en el mismo orden cuando se utilizan en un proceso de decodificación iterativa [A.WOOD]: la menor complejidad y el peor desempeño los ofrece el algoritmo SOVA, a continuación se encuentra el algoritmo Max-Log-MAP, luego el algoritmo Log-MAP, y finalmente el algoritmo MAP, siendo este último el algoritmo óptimo para el decodificado de Turbo-códigos pero a su vez un algoritmo extremadamente complejo. Atendiendo a [T.VAL99], en canales AWGN el algoritmo Max-Log-MAP provee un aproximado de 0.5 dB más de ganancia de codificación que la que provee el algoritmo SOVA, y el algoritmo Log-MAP provee un aproximado de 0.5 dB más de ganancia de codificación que la que provee el algoritmo Max-Log-MAP.

Además de los algoritmos MAP, Log-MAP, Max-Log-MAP y SOVA, existen otros algoritmos que se desarrollaron después que éstos y que también pueden ser utilizados en un Turbo-decodificador; por ejemplo, en 1998 V. Franz y J. B. Anderson propusieron en [A.FRAN] un algoritmo MAP de "búsqueda reducida" que de acuerdo con los autores es capaz de reducir la complejidad computacional de un Turbo-decodificador. Sin embargo, estos algoritmos no son tan ampliamente utilizados por los Turbo-códigos como los primeros debido a que no representan en realidad una mejora en el desempeño que puede alcanzarse con la utilización de los algoritmos MAP o Log-MAP, ni tampoco una reducción

en la complejidad algorítmica mayor que la que puede conseguirse con los algoritmos Max-Log-MAP o SOVA.

A continuación exponemos los algoritmos de decodificación SISO más utilizados en el proceso de decodificación iterativa de los Turbo-códigos. Debido a la gran complejidad matemática con la que tratan estos algoritmos sólo presentaremos una descripción precisa de cada uno de ellos, siguiendo una presentación similar a la ofrecida por [BAR04]. Una descripción matemática completa de estos y otros algoritmos de decodificación iterativa puede consultarse en [A.WOOD][T.VAL99].

3.3.2.1. Algoritmo SOVA.

Consideremos el diagrama de Trellis de un código convolucional en el que dos trayectorias divergen en la unidad de tiempo **i-d** y se vuelven a unir en la unidad de tiempo **i+1**. Las trayectorias sobrevivientes para los dos estados A y B en la unidad de tiempo **i**, desde las cuales las dos trayectorias volverán a unirse, están dadas por [BAR04]:

$$V_{A1} = \{ u_{1,(i-1)}, u_{1,(i-2)}, \ldots u_{1,(i-d)}, u_{(i-d-1)}, \ldots . \} \quad (3.6)$$

$$V_{B1} = \{ u_{2,(i-1)}, u_{2,(i-2)}, \ldots u_{2,(i-d)}, u_{(i-d-1)}, \ldots . \} \quad (3.7)$$

donde $u_{(i-d-1)}$ es la salida suave común previa a la divergencia.

Sea $\mathbf{M_{A,i}}$ la métrica de trayectoria en la unidad de tiempo **i** para la trayectoria proveniente del estado A, $\mathbf{z_i}$ la observación en la unidad de tiempo **i**, y $\mathbf{y_{1,i}}$ la salida deseada para la trayectoria 1. En la unidad de tiempo **i+1**, la métrica de trayectoria para la trayectoria proveniente del estado A está dada por [BAR04]:

$$m_1 = M_{A,i} + |z_i - y_{1,i}|^2 \quad (3.8)$$

Una definición similar aplica para la trayectoria proveniente del estado B [BAR04]:

$$m_2 = M_{B,i} + |z_i - y_{2,i}|^2 \quad (3.9)$$

Atendiendo a [BAR04] si $m_1 < m_2$, un decodificador Viterbi de salidas duras generaría la siguiente trayectoria sobreviviente:

$$V_{1,(i+1)} = \{ u_{1,i}, u_{1,(i-1)}, u_{1,(i-2)}, \ldots u_{1,(i-d)}, u_{(i-d-1)}, \ldots . \} \quad (3.10)$$

En el caso de un decodificador Viterbi de salidas suaves (SOVA), su salida está dada por [BAR04]:

$$V_{1,(i+1)} = \{[(1-D)u_{1,i} + Du_{2,i}], \ldots [(1-D)u_{1,(i-d)} + Du_{2,(i-d)}], u_{(i-d-1)}, \ldots . \} \quad (3.11)$$

donde $D = \exp(-m_1/2\sigma^2) / [\exp(-m_1/2\sigma^2) + \exp(-m_2/2\sigma^2)]$ para el caso de un canal AWGN en el que σ^2 es la varianza del ruido.

De acuerdo con [BAR04], la información extrínseca producida por un decodificador SOVA está altamente correlacionada con la información a-priori que ingresó al decodificador, lo cual se debe a que el algoritmo SOVA produce la salida suave considerando, en vez de todas las trayectorias, únicamente dos: la trayectoria de máxima probabilidad y la segunda mejor trayectoria. Esto ocasiona que el desempeño ofrecido por SOVA no sea óptimo.

3.3.2.2. Algoritmo MAP.

De acuerdo con [BAR04], el algoritmo MAP implica el cálculo de *métricas de estado hacia adelante* ("forward state metrics") α_i^m para cada estado *m* y tiempo *I*, como se muestra en la Figura 3.7. Estas métricas se basan en métricas de estado previas $\alpha_{i-1}^{m_j}$ y en métricas de rama previas $\delta_{i-1}^{k,m}$.

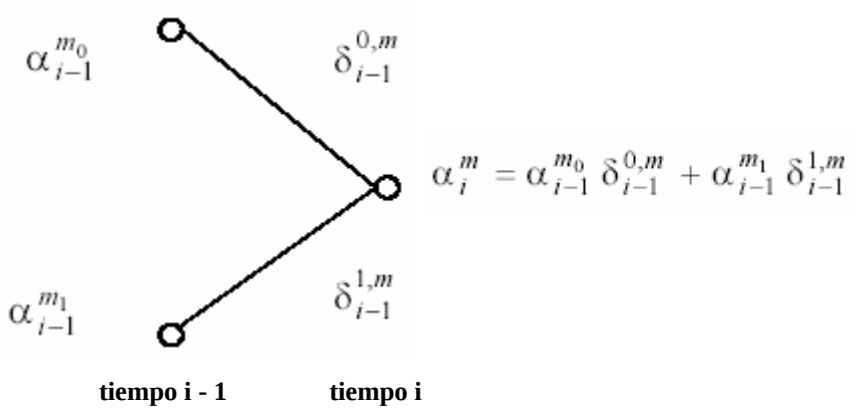

Figura 3.7. Cálculo de las métricas de estado hacia adelante. [BAR04].

El cálculo de un nuevo estado está típicamente dado por [BAR04]:

$$\alpha_i^m = \alpha_{i-1}^{m_0}\delta_{i-1}^{0,m} + \alpha_{i-1}^{m_1}\delta_{i-1}^{1,m} \tag{3.12}$$

donde m_0 (ó m_1) es el estado a partir del cual, dado un bit de entrada cero (ó uno), el estado del codificador cambia al estado *m*.

Dado que existen 2^v estados, donde *v* es el tamaño de la memoria del codificador, se requiere calcular 2^v métricas de estado nuevas. En la práctica el codificador parte del estado cero con cada bloque nuevo, por lo que las métricas de estado hacia adelante son inicializadas a cero exceptuando a la métrica de estado cero [BAR04].

Las métricas de rama se calculan por medio de [BAR04]:

$$\delta_i^{k,m} = P(d_i = k)\exp(L_c(x_i k + y_i c^{k,m})) \tag{3.13}$$

donde d_i es el bit de información, x_i es el símbolo de información recibido, c es el bit codificado, y_i es el símbolo codificado recibido, k puede ser 0 o 1, y $L_c = 2/(\sigma^2)$. Para cada x_i e y_i recibidas existen únicamente cuatro posibles combinaciones de d_i y c_i, por lo que se requiere calcular únicamente cuatro métricas de rama diferentes. La ecuación de las métricas de estado hacia adelante necesita dos de las cuatro métricas de rama para cada cálculo nuevo de métricas de estado [BAR04].

Las métricas de estado nuevo también deben ser calculadas yendo hacia atrás en el tiempo (comenzando al final del bloque), como se muestra en la Figura 3.8. Estas métricas se representan por β_i^m y un cálculo típico es [BAR04]:

$$\beta_i^m = \beta_{i+1}^{m_0} \delta_i^{0,m} + \beta_{i+1}^{m_1} \delta_i^{1,m} \tag{3.14}$$

donde m_0 (ó m_1) es el siguiente estado del codificador si el bit de entrada es cero (ó uno) y el estado del codificador es m.

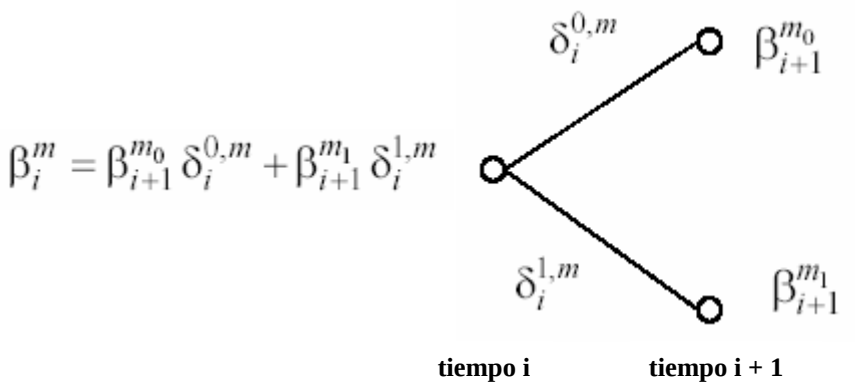

Figura 3.8. Cálculo de las métricas de estado hacia atrás. [BAR04].

Si el Trellis no es terminado en un estado conocido (generalmente el estado cero), las *métricas de estado hacia atrás* o "backward state metrics" β_i^m se inicializan al mismo valor.

La salida producida por el algoritmo MAP está dada por [BAR04]:

$$\lambda_i = \frac{\sum_m \alpha_i^m \delta_i^{1,m} \beta_{i+1}^{m_1}}{\sum_m \alpha_i^m \delta_i^{0,m} \beta_{i+1}^{m_0}} \tag{3.15}$$

La implementación del algoritmo MAP requiere calcular y almacenar las métricas de estado hacia adelante α_i^m para el bloque completo. Una vez que el bloque entero es recibido, las métricas de estado hacia atrás β_i^m y las salidas suaves λ_i pueden ser calculadas.

3.3.2.3. Algoritmo Log – MAP.

En este algoritmo las multiplicaciones involucradas en el algoritmo MAP son convertidas en sumas, lo cual reduce la complejidad de decodificado de este último algoritmo debido a que un sumador es más fácil de implementar que un multiplicador. De acuerdo con [BAR04], en el algoritmo Log-MAP las sumas involucradas en el algoritmo MAP son convertidas en el operador *E* dado por:

$$a\ E\ b \equiv -\log_c(c^{-a} + c^{-b}) = \min(a, b) - \log_c(1 + c^{-|a-b|}) \tag{3.16}$$

donde $c = \exp(\sigma^2 A/2)$, σ^2 es la varianza del ruido, y A es la amplitud de la señal.

Consideremos las siguientes expresiones, donde L es la relación Log-likelihood (LLR) producida por el decodificador Log – MAP, A es la métrica de estado hacia adelante, B es la métrica de estado hacia atrás, D es la métrica de rama, i es el indicador del tiempo, y m es el indicador del estado:

$$L_i = -\log_c \lambda_i \tag{3.17}$$

$$A_i^m = -\log_c \alpha_i^m \tag{3.18}$$

$$B_i^m = -\log_c \beta_i^m \tag{3.19}$$

$$D_i^{k,m} = -\log_c \delta_i^{k,m} \tag{3.20}$$

El algoritmo Log-MAP está dado, entonces, por [BAR04]:

$$L_i = \underset{m=0}{\overset{2^v-1}{E}}(A_i^m + D_i^{1,m} + B_{1+i}^{f(1,m)}) - \underset{m=0}{\overset{2^v-1}{E}}(A_i^m + D_i^{0,m} + B_{1+i}^{f(0,m)}) \tag{3.21}$$

$$A_i^m = \underset{i=0}{\overset{1}{E}}(A_{i-1}^{b(k,m)} + D_{i-1}^{k,b(k,m)}) \tag{3.22}$$

$$B_i^m = \underset{i=0}{\overset{1}{E}}(D_i^{k,m} + B_{i+1}^{f(k,m)}) \tag{3.23}$$

donde $\underset{j=0}{\overset{t-1}{E}} a^j = a^0 E a^1 E \ldots E a^{t-1}$; $b(k, m)$ es el estado del codificador a partir del cual, dado el bit de entrada k, el estado del codificador cambia al estado m; y $f(k, m)$ es el siguiente estado del codificador si el estado actual es m y el bit de entrada es k.

3.3.3. Complejidad de decodificación de un Turbo-código.

Anteriormente mencionamos que la gran complejidad del algoritmo MAP es debida al elevado número de operaciones que este algoritmo debe efectuar para producir el estimado de la probabilidad a-posteriori de cada bit de información decodificado. Del mismo modo, es común medir la complejidad de decodificación de Turbo-códigos en términos de la cantidad de operaciones que un Turbo-decodificador debe efectuar para producir cada bit decodificado. De acuerdo con [T.SEG95], la complejidad de decodificación de un código o de un algoritmo de decodificación cualesquiera "se mide como el número de operaciones por bit de información que se requieren para decodificar." (Seghers, 1995: 49).

La Tabla 3.6 nos muestra el número total de operaciones requeridas por el algoritmo MAP para decodificar un bit de información codificado por un codificador RSC con una memoria de tamaño M (complejidad de decodificado del algoritmo MAP). También nos muestra la complejidad de decodificación de un Turbo-código considerando un proceso de decodificación de 1 y 18 iteraciones y un Turbo-codificador cuyos codificadores componentes tienen una memoria de tamaño M y $M = 4$, respectivamente. Nótese que la complejidad del proceso de decodificación de 1 iteración es el doble de la complejidad de decodificado del algoritmo MAP, lo cual resulta ser algo obvio si tomamos en cuenta que cada iteración de un proceso de decodificación iterativa requiere que el algoritmo MAP sea aplicado dos veces.

Tabla 3.6. Complejidad de la decodificación de un Turbo-código [T.SEG95].

	Sumas	Multiplicaciones
Decodificador MAP	$4 \bullet 2^M$	$6 \bullet 2^M + 1$
Turbo-decodificador (1 iteración)	$8 \bullet 2^M$	$12 \bullet 2^M + 2$
Turbo-decodificador (18 iteraciones, $M = 4$)	2304	3492

Nótese que la complejidad de decodificación de un Turbo-código únicamente depende del tamaño de la memoria (M) de los codificadores componentes del Turbo-codificador.

Es importante mencionar que a pesar de que la complejidad de decodificación de un Turbo-código parece ser alta (ver Tabla 3.6), la complejidad de decodificado de un código convolucional que empleando el algoritmo de Viterbi ofreciera un desempeño similar al de un Turbo-código de alto desempeño sería, simplemente, inaceptable.

3.4. Desempeño de un Turbo-código en función de sus parámetros CODEC.

En el Capítulo 1 mencionamos que uno de los resultados más notables del trabajo de Claude Shannon, *A Mathematical Theory of Communication,* lo constituye el haber definido el límite fundamental de la velocidad a la que es posible lograr una transmisión de información confiable, virtualmente libre de errores, en un sistema de comunicación y la herramienta que debe utilizarse para poder alcanzar ese límite: nos referimos a la capacidad del canal y a la codificación para control de errores. Shannon estipuló que mediante la utilización de esquemas de codificación del canal adecuados es posible lograr una comunicación que ocurra a una velocidad igual a la capacidad del canal y con una probabilidad de error de símbolo (BER) muy baja.

Al principio de este Capítulo se mencionó que los Turbo-códigos representan un esquema de codificación muy poderoso e innovador debido a que le permiten a un sistema de comunicación operar a tan sólo 0.1 dB de distancia de la capacidad del canal, sin embargo no se profundizó más sobre este asunto. Nos encontramos en un punto en el que debemos considerar nuevamente y más a fondo esta importante afirmación sobre los Turbo-códigos y, sobre todo, debemos estudiar los más importantes de los distintos parámetros que afectan el desempeño de este esquema de codificación. De esta manera, el lector de esta Tesis podrá visualizar a través de gráficas de BER vs Eb/No las curvas de desempeño de distintas arquitecturas de Turbo-códigos (códigos Turbo caracterizados por parámetros diferentes) y así comprobar por sí mismo que existe más de una combinación de parámetros que hacen que este esquema de codificación constituya una respuesta directa a las predicciones de Shannon.

Algunos de estos parámetros son: el algoritmo de decodificación SISO utilizado, el número de iteraciones utilizado en el proceso de decodificado, el tamaño del interleaver utilizado, y la tasa de código utilizada en el proceso de codificado. Es importante recordar que en esta Tesis se estudia el esquema de codificación Turbo aplicado a canales AWGN, por lo que los resultados que se presenten en este sub-capítulo corresponderán a la forma en que se comportarían (desempeño) los distintos Turbo-códigos si operaran en sistemas de comunicación cuyo canal físico fuera afectado exclusivamente por ruido blanco Gaussiano.

Pero, para empezar, ¿qué significa que los Turbo-códigos le permitan a un sistema de comunicación operar a tan sólo 0.1 dB de distancia de la capacidad del canal?.

Retomemos la ecuación (1.11):

$$\frac{C}{W} = \log_2\left(1 + \frac{E_b R_s}{N_0 W}\right) \tag{3.24}$$

En esta ecuación C representa la máxima tasa a la que podemos transmitir información con un nivel de confiabilidad tan alto como se desee a través de un canal continuo AWGN limitado en banda a W Hertz; R_s representa la tasa de transferencia de información del sistema de comunicación que se considere y Eb/No representa la SNR requerida por dicho sistema para entregar cada bit de información con un nivel de confiabilidad deseado.

Consideremos un sistema digital codificado que emplea un codificador de tasa de código r. De acuerdo con la sección 1.3 de esta Tesis, la tasa de transmisión de información del sistema será de r/T bits por segundo (donde $1/T$ representa la tasa de transmisión de símbolos codificados dada en *símbolos por segundo*), y el ancho de banda del sistema deberá ser de $W = 1/(2T)$ Hertz para minimizar el efecto de la distorsión de canal en el proceso de detección de señales. El desempeño óptimo de un sistema de este tipo se alcanzará si su tasa de transmisión es igual a C bits/seg. y cada bit de información es

entregado al receptor con un alto nivel de confiabilidad. Si hacemos $C = r/T$, $W = 1/(2T)$, y $R_s = C$, y sustituimos estos valores en la ecuación (3.24), obtendremos **la relación *Eb/No* a la que el sistema lograría su desempeño óptimo**, denotada por **(*Eb/No*)*op***:

$$\frac{\frac{r}{T}}{\frac{1}{2T}} = \log_2\left(1 + \frac{E_b\left(\frac{r}{T}\right)}{N_0\left(\frac{1}{2T}\right)}\right) \Rightarrow \frac{E_b}{N_0}op = \frac{2^{2r}-1}{2r} \qquad (3.25)$$

A partir de la ecuación (3.25) podemos ver que dicha relación es una función de la tasa de código del codificador empleado por el sistema. Para una tasa de código $r = 1/2$, la relación $(Eb/No)op = 1$ (ó 0 dB). De acuerdo con [BAR04], simulaciones realizadas en [A.GLAV] indican que el Turbo-código considerado ahí, cuya tasa es $r = ½$ y su interleaver de longitud $L = 65536$ bits, es capaz de alcanzar una BER = 10^{-6} empleando una SNR por bit de información (*Eb/No*) igual a 0.7 dB. Así mismo, en [BAR04] se comenta que en [A.BRIN] se presenta un esquema SCCC de tasa $r = ½$ con un interleaver de longitud $L =$ 1,000,000 bits que alcanza una BER igual a 10^{-5} empleando una SNR por bit de información de 0.28 dB. Numerosos artículos científicos afirman que existen Turbo-códigos de tasa $r = ½$ capaces de alcanzar una BER = 10^{-5} empleando una relación *Eb/No* = 0.1 dB, lo que nos permite reafirmar lo dicho anteriormente: los Turbo-códigos representan un esquema de codificación muy poderoso debido a que le permiten a un sistema de comunicación operar a tan sólo 0.1 dB de distancia de la capacidad del canal.

En [BAR04] se realizaron simulaciones con 4 Turbo-códigos del tipo PCCC que emplean un interleaver de longitud $L = 10200$ bits y cuyas tasas de codificación son $r =$ 1/2, 1/3, 1/4 y 1/6. La relación *Eb/No* requerida por estos códigos para alcanzar una BER de

10^{-6} se muestra en la siguiente tabla, en la que también se muestra la relación (*Eb/No*)*op* para estas tasas de código.

Tabla 3.7. Relación (*Eb/No*)*op* en función de la tasa de código.

Tasa de código	**(*Eb/No*)*op* (dB)**	***Eb/No* de simulaciones realizadas en [BAR04]**
1/2	0.00	0.98
1/3	-0.55	0.37
1/4	-0.82	0.13
1/6	-1.08	-0.12
0	-1.59	-----

En esta tabla se puede apreciar el gran desempeño que se puede lograr con un Turbo-código: para una longitud de frame no excesivamente grande (10200 bits) los Turbo-códigos simulados ofrecen una probabilidad de error de símbolo considerablemente baja (10^{-6}) estando, en promedio, a tan sólo 0.95 dB de distancia de la relación (*Eb/No*)*op* (y por consiguiente de la capacidad del canal). Nótese que el desempeño ofrecido por los 4 Turbo-códigos es muy similar: los cuatro alcanzan una BER de 10^{-6} estando casi a la misma distancia de la capacidad del canal (≈ 0.95 dB); esto nos permite concluir que *sin importar su tasa de código, un Turbo-código se comporta de manera similar, ofreciendo un muy buen desempeño* [BAR04]. En la tabla 3.7 también podemos ver que la ganancia de codificación de un Turbo-código de tasa *r* con respecto al Turbo-código de tasa 1/2 es mayor conforme *r* se decrementa. Esto se puede apreciar mejor en la siguiente tabla, en la que se muestra la SNR por bit de información requerida para alcanzar una BER de 10^{-6}, en función de la tasa de código, para los Turbo-códigos simulados en [BAR04] y cuatro códigos convolucionales considerados en [A.LEE]:

Tabla 3.8. *Eb/No* requerida para una BER = 10^{-6} para CC´s y TC´s [BAR04].

Tasa de código	Eb/No (en dB) requerida para BER = 10^{-6}		Ganancia de codificación (en dB) con respecto al código de tasa r = 1/2	
	Código Convolucional	Turbo-código	Código Convolucional	Turbo-código
1/2	4.80	0.98	0.00	0.00
1/3	4.49	0.37	0.31	0.61
1/4	4.37	0.13	0.43	0.85
1/6	4.28	-0.12	0.52	1.10

A partir de la tabla podemos ver que la ganancia de codificación que presentan los Turbo-códigos con respecto al código Turbo de tasa r = 1/2 es casi del doble de aquella presentada por los códigos convolucionales con respecto al código convolucional de tasa r = 1/2. Esto nos permite concluir, de acuerdo con [BAR04], que *decrementar la tasa de un Turbo-código produce una ganancia de codificación considerablemente mayor que aquella que se produce al decrementar la tasa de un código convolucional.* Nótese el impresionante desempeño que ofrece un Turbo-código en comparación con el desempeño ofrecido por un código convolucional: en la tabla 3.8 se puede ver que para las tasas de código consideradas y para una BER = 10^{-6} los Turbo-códigos presentan, en promedio, una ganancia de codificación de 4 dB con respecto a los códigos convolucionales. Para que un código convolucional ofreciera un desempeño semejante al ofrecido por un Turbo-código su memoria tendría que ser de un orden muy alto, lo que implicaría una decodificación de Viterbi extremadamente compleja e inútil en la práctica. ¡He ahí la necesidad que se tenía de encontrar un nuevo y mejor esquema de codificación!.

En realidad, y estrictamente hablando, la gran necesidad que se tenía antes de la invención de los Turbo-códigos era la de encontrar un esquema de *de*codificación que fuera

capaz de ofrecer un desempeño muy cercano a la solución óptima con una complejidad en la implementación no prohibitiva. En otras palabras, lo que se buscaba era obtener un algoritmo de decodificación que pudiera decodificar una secuencia de información a partir de palabras de código complejas, en una manera útil en la práctica y ofreciendo un desempeño lo más cercanamente posible al que ofrecería un algoritmo de decodificación de máxima probabilidad (MLDA). De esta manera, sería posible alcanzar tasas de error (BER) muy bajas empleando un mínimo de energía para transmitir en los sistemas digitales de comunicación reales.

Dado que el Interleaver de un Turbo-codificador tiene una estructura fija y trabaja con los bits de información por bloques, los Turbo-códigos se consideran códigos de bloque (ver sección 3.2.2.). Si recordamos que el Teorema de Codificación del Canal de Shannon (ver sección 1.2.1.) implica que se puede alcanzar una BER muy baja por medio de un código de bloque si la longitud *n* de sus palabras de código se hace grande, entonces el esquema de decodificación iterativa de los Turbo-códigos constituye la respuesta a esa búsqueda del algoritmo de decodificación "ideal", al permitirle a este esquema de codificación trabajar con palabras de código muy largas para producir tasas de error muy bajas.

Pero, ¿qué tan lejos o cerca está el desempeño ofrecido por este esquema de decodificación de aquél que ofrecería una decodificación óptima, de máxima probabilidad?. De acuerdo con [A.WOOD][BAR04], esta técnica iterativa es una forma muy eficiente de decodificar a los Turbo-códigos pues ofrece un desempeño cuasi-óptimo. En [A.MON][A.DIV] se obtienen curvas de desempeño teóricas para distintos tipos de arquitecturas PCCC que marcan el máximo desempeño en términos de BER vs Eb/No que con ellas se alcanzaría *empleando un algoritmo de decodificación de máxima probabilidad*.

Ahí mismo se efectúan simulaciones que permiten ver que las curvas de desempeño obtenidas por medio de una decodificación iterativa convergen con las curvas teóricas de máximo desempeño conforme el número de iteraciones se incrementa, evidenciando así la gran eficiencia del procedimiento iterativo de decodificación.

3.4.1. El Turbo-código visto como un Código de Bloque Lineal.

Como ya se dijo, los Turbo-códigos se consideran códigos de bloque. Si el Interleaver tiene un tamaño fijo y ambos codificadores RSC del Turbo-codificador comienzan a codificar con sus registros inicializados a cero, entonces el Turbo-código se considera un código de bloque lineal [T.VAL99].

En la sección 2.2.1. vimos que la distancia de Hamming de un código de bloque (también llamada *distancia mínima,* d_{min}) es un buen estimado, de primer orden, del desempeño del código. Para códigos de bloque lineales la distancia mínima corresponde al peso de Hamming más pequeño, diferente de cero, de todas las palabras de código en el código. Un objetivo primordial en el diseño de códigos de bloque en general siempre ha sido tratar de maximizar d_{min}. Atendiendo a [T.VAL99], la combinación del Interleaver y de una codificación RSC asegura que la mayoría de las palabras de código producidas por un Turbo-código tengan un alto peso de Hamming. Dado que un codificador RSC tiene una respuesta infinita al impulso, su salida generalmente tiene un alto peso de Hamming, aunque existen algunas secuencias de entrada que provocan que un codificador RSC produzca salidas de bajo peso. Sin embargo, debido a la presencia del Interleaver los dos codificadores RSC del Turbo-codificador reciben secuencias de entrada distintas (en distinto orden), por lo que si uno de los codificadores recibe una secuencia que origina una salida de bajo peso, es poco probable que el otro codificador también reciba una secuencia

que origine una salida con bajo peso de Hamming. Lamentablemente siempre existirán algunos mensajes de entrada al Turbo-codificador que provocarán que ambos codificadores RSC produzcan salidas de bajo peso y por consiguiente la distancia mínima de un Turbo-código no es, en general, particularmente alta.

Pero, de acuerdo con [T.VAL99], la multiplicidad de palabras de código de bajo peso producidas por un Turbo-codificador bien diseñado es baja, lo que permite que un Turbo-código ofrezca un buen desempeño a bajas SNR´s. Lamentablemente el desempeño de un Turbo-código a SNR´s más altas se ve limitado por la relativamente pequeña distancia mínima del código. Es por ello que "mientras el objetivo del diseño tradicional de códigos consiste en incrementar la distancia mínima del código, el objetivo del diseño de Turbo-códigos consistirá en reducir la multiplicidad de palabras de código de bajo peso." (Valenti, 1999: 37).

Considérese, por ejemplo, el Turbo-codificador ilustrado en la Figura 3.2. La siguiente figura nos muestra el desempeño ofrecido por este Turbo-código en un canal AWGN utilizando un Interleaver de 65536 bits (adviértase que un codificador de este tipo producirá palabras de código muy largas) y 18 iteraciones en el proceso de decodificación empleando el algoritmo log-MAP [T.VAL99].

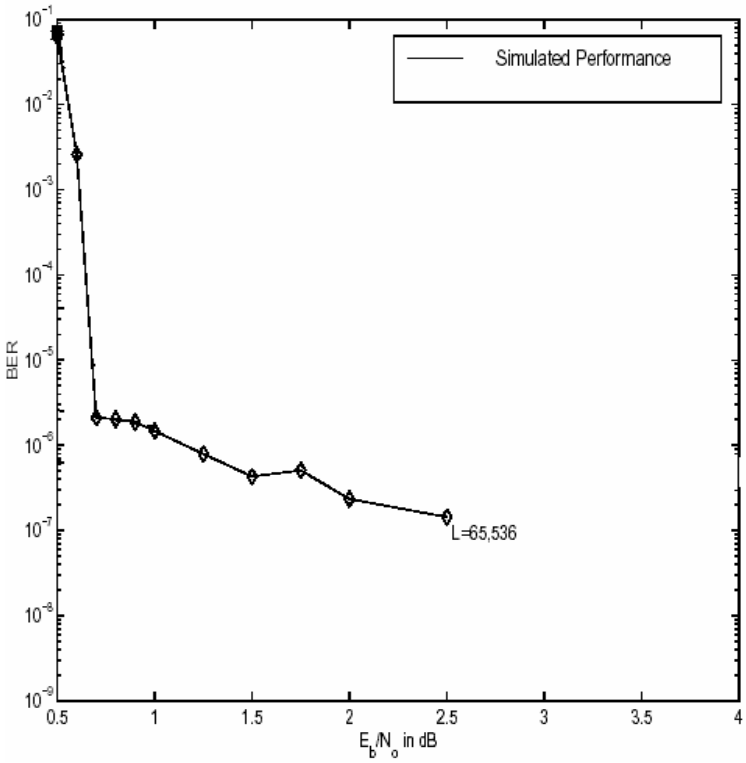

Figura 3.9. Desempeño de un Turbo-código de tasa r = ½ con I = 65536 bits. [T.VAL99].

En la Figura 3.9 se puede apreciar claramente el buen desempeño de este Turbo-código para una relación *Eb/No* < 0.7 dB. Sin embargo, nótese cómo la curva de desempeño del Turbo-código sufre un cambio radical en su pendiente a una *Eb/No* ≈ 0.7 dB; esto se traduce en un desempeño limitado de este código para una *Eb/No* > 0.7 dB.

Este cambio en la pendiente de la curva de desempeño de un Turbo-código constituye un rasgo distintivo clave de este esquema de codificación, y de acuerdo con [BAR04][T.VAL99] se conoce comúnmente como el ***piso de la tasa de error de bit*** (ó *BER-floor*, en Inglés). La BER a la que se produce el *BER-floor* en la curva de desempeño de un Turbo-código es una función, principalmente, del *tamaño del Interleaver*. Esto se

puede apreciar muy bien en la siguiente figura, en la que se añaden otras tres curvas de desempeño a la Figura 3.9 obtenidas, en [T.VAL99], de simular el mismo Turbo-código de la Figura 3.2 utilizando Interleavers de tamaño "L" distinto. Para la obtención de estas curvas también se utilizaron 18 iteraciones en el proceso de decodificación empleando el algoritmo log-MAP.

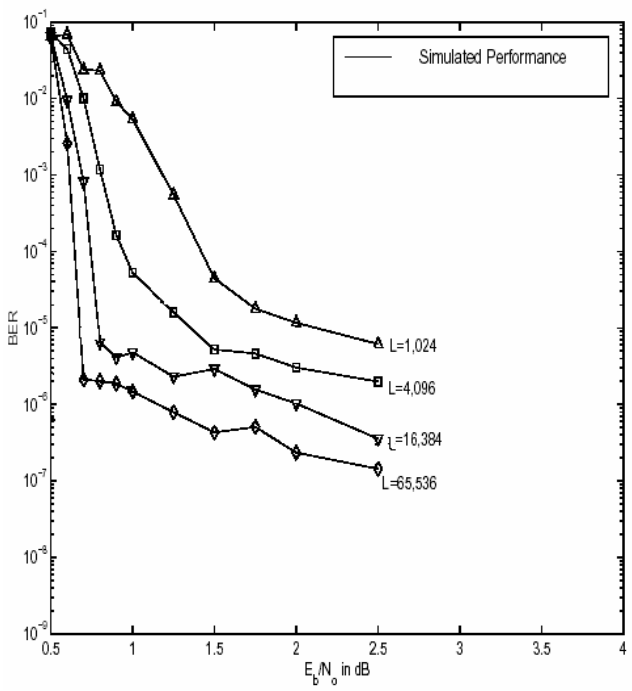

Figura 3.10. Desempeño de 4 T.C. de tasa r = ½ con I = 1024, 4096, 16384, 65536 b. [T.VAL99].

Nótese, a partir de la Figura 3.10, cómo el desempeño de un Turbo-código está fuertemente relacionado con el tamaño del Interleaver utilizado. Podemos ver que entre mayor es el Interleaver que se utiliza se obtiene un mejor desempeño del Turbo-código, por lo que podríamos suponer – y no de manera errónea – que con un Interleaver extremadamente grande un Turbo-código ofrecería un magnífico desempeño (esto es válido no sólo para un Turbo-código de tasa r = ½, sino para un Turbo-código de cualquier tasa r;

además, dado que el proceso de decodificación de un Turbo-código es iterativo, estamos considerando que el Interleaver produce una secuencia muy poco correlacionada con la secuencia original de información, ya que de lo contrario el algoritmo de decodificación iterativa no ofrecería un desempeño cuasi-óptimo y el desempeño del Turbo-código sería pobre, sin importar cuán grande fuese el Interleaver utilizado.). De esto podemos concluir que *el tamaño del Interleaver es uno de los factores que más influyen en el desempeño de un Turbo-código.*

Lamentablemente, entre más grande sea el Interleaver utilizado mayor será la memoria y el tiempo requeridos por el Turbo-decodificador para producir su secuencia de salida (mensaje decodificado); es por esto que "el tamaño del Interleaver debe elegirse de forma tal que se satisfagan los requerimientos del sistema de BER, tardanza de decodificación y memoria utilizada por el decodificador." (Valenti, 1999: 44).

Este último punto es muy importante ya que para muchas aplicaciones, como los sistemas de transmisión de voz, los largos retardos ocasionados por la transmisión de bloques de información, ó *frames*, muy largos (k > 500 bits) son inaceptables. Transmitir *frames* arriba de 2000 bits sería útil únicamente para sistemas de transmisión de datos ó sistemas que no requieran de una transmisión en tiempo real. Esto podría hacernos dudar por un instante de la importancia de utilizar Turbo-códigos en sistemas de telefonía celular y, por consiguiente, ¡de la importancia de esta Tesis!. Sin embargo, de acuerdo con [A.WOOD], para un sistema que utilizara 169 bits por *frame* los Turbo-códigos aún brindarían buenos resultados, comparables ó mejores que aquellos que ofrecería un código convolucional con una memoria de orden m = 8, y con una complejidad en el proceso de decodificado un tercio menor que la del código convolucional. Además los estándares de

telefonía celular no sólo utilizan canales de transmisión de voz (en los que se requiere una transmisión en tiempo real), sino también canales de datos, y es en estos canales en los que en los más recientes estándares de telefonía celular de 3G se decidió utilizar a los Turbo Códigos, por los grandes beneficios que, como veremos en el capítulo siguiente, traía consigo su utilización.

Por si esto fuera poco, un área importante de investigación sobre codificación-Turbo ya ha conseguido obtener en sistemas de *frames* cortos resultados tan sorprendentes como los obtenidos en sistemas de *frames* largos, gracias a la utilización de los Turbo-códigos. Por ejemplo, en [A.SALA] se presenta una forma de mejorar considerablemente el desempeño de un Turbo-código para *frames* cortos, mediante una distribución desigual y específica de energía entre los bits sistemáticos y de paridad de las palabras de código producidas por el Turbo-codificador.

3.4.2. Parámetros importantes que afectan el desempeño de un Turbo-código.

Como se mencionó al principio de este sub-capítulo, además del tamaño del Interleaver existen otros parámetros que pueden repercutir de manera importante en el desempeño de un Turbo-código. Hasta este punto sólo hemos visto que uno de esos parámetros lo constituye *la tasa de código r* de un Turbo-código: mientras más pequeña sea su tasa, mejor será el desempeño de un Turbo-código (ver Tabla 3.8).

El *diseño del Interleaver* también constituye un parámetro importante, pero sólo a SNR´s altas. De acuerdo con [T.VAL99], a una SNR baja los Turbo-códigos ofrecen un buen desempeño utilizando casi cualquier tipo de Interleaver (considerando que ambos RSC´s del Turbo-codificador reciben secuencias distintas y muy poco correlacionadas), pero a una SNR alta el desempeño de un código Turbo es afectado por las palabras de

código con un bajo peso de Hamming (que provocan que d_{min} sea relativamente pequeña), mismas que son significantemente "influenciadas" por el diseño del Interleaver. Atendiendo a [A.WOOD], a pesar de que se han propuesto varios algoritmos, como por ejemplo en [A.ROBE], para diseñar buenos Interleavers que maximicen la distancia mínima (d_{min}) de un Turbo-código, este proceso es muy complicado, y los Interleavers resultantes no son necesariamente óptimos. En [A.WOOD] se muestra que los Interleavers aleatorios ofrecen el mejor desempeño cuando se utilizan en Turbo-códigos para *frames* largos, mientras que para sistemas de *frames* cortos el mejor desempeño se obtiene utilizando un Interleaver de bloque con un número impar de renglones y de columnas.

Situados en este contexto podemos decir que *un buen Interleaver es aquél que: tiene un tamaño relativamente grande y está diseñado tanto para producir una secuencia que esté muy poco correlacionada con la secuencia original de información, como para tratar de maximizar la distancia mínima del Turbo-código*. Es importante mencionar que la distancia mínima de un Turbo-código también es afectada por los *polinomios generadores* $\mathbf{g^{(a)}(D)}$ y $\mathbf{g^{(b)}(D)}$ que se utilicen en los codificadores componentes del Turbo-codificador (ver sección 2.3.3.); sin embargo, de acuerdo con [A.WOOD], los Turbo-códigos casi siempre utilizan RSC´s cuyos polinomios generadores optimizan la mínima distancia libre (d_{free}) de estos últimos, haciendo que la distancia mínima de los primeros no se vea afectada negativamente por este parámetro.

Otros dos parámetros que repercuten de manera importante en el desempeño de un Turbo-código los constituyen *el tipo de algoritmo de decodificación SISO* utilizado en el proceso de Turbo-decodificación y *el número de iteraciones* utilizado en el mismo.

La siguiente figura nos muestra el efecto que produce en la curva de desempeño de un Turbo-código el variar el número de iteraciones utilizado en el proceso de

decodificación iterativa, empleando el algoritmo de decodificación MAP. Las curvas que se presentan fueron obtenidas en [A.WOOD] luego de simular, para un número variable de iteraciones, un Turbo-código de tasa $r = ½$ que utiliza dos codificadores RSC idénticos, de tasa $r = ½$ y polinomios generadores $\mathbf{g}^{(a)}\mathbf{(D)} = 1 + D^2$ y $\mathbf{g}^{(b)}\mathbf{(D)} = 1 + D + D^2$, y un Interleaver "Impar-Par" de 1000 bits. Para estas simulaciones se empleó una modulación BPSK sobre un canal AWGN.

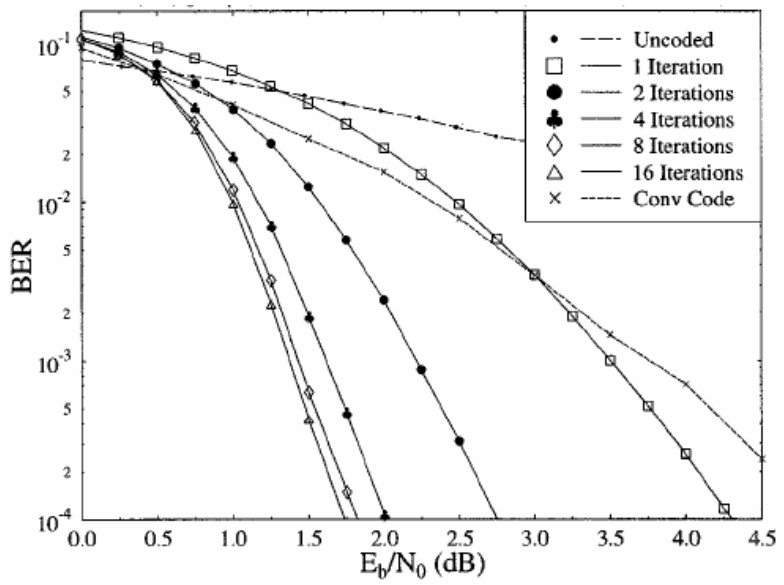

Figura 3.11. Desempeño de un Turbo-código en función del número de iteraciones. [A.WOOD].

La Figura 3.11 también nos muestra la curva BER que resulta de no utilizar una codificación para el control de errores y la curva BER que resulta de utilizar un código convolucional (2, 1, 2) NSNR, con polinomios generadores $\mathbf{g}^{(1)}\mathbf{(D)} = 1 + D + D^2$ y $\mathbf{g}^{(2)}\mathbf{(D)} = 1 + D^2$. Nótese que después de una iteración el desempeño del Turbo-código es similar al ofrecido por el código convolucional; sin embargo, podemos ver que conforme el número de iteraciones utilizadas por el Turbo-decodificador se incrementa, el desempeño del

código Turbo mejora significantemente: después de apenas 2 iteraciones el Turbo-código ya presenta una ganancia de codificación sobre el código convolucional de aproximadamente 2.25 dB para una BER = 10^{-4}. Es importante notar, sin embargo, que después de ocho iteraciones ya no se obtiene una mejora considerable en el desempeño del Turbo-código conforme este número se incrementa, lo cual coincide con lo que se dijo al final de la sección 3.3.1.

Este mismo efecto que se observa en la Figura 3.11 se producirá en el desempeño de cualquier Turbo-código, sin importar que utilice otro algoritmo de decodificación SISO (distinto a MAP): *entre más grande sea el número de iteraciones utilizado por el Turbo-decodificador mejor será el desempeño de un Turbo-código, aunque la diferencia entre usar 8 iteraciones ó más ya no será muy significativa.* Sin embargo, es importante hacer notar que entre más grande sea el número de iteraciones utilizado, la duración del proceso de Turbo-decodificación (retardo de decodificación) será cada vez mayor. Podría pensarse que éste es un punto débil de los Turbo-códigos, pues muchas aplicaciones requerirán que el retardo de decodificación sea lo más corto posible; mas de acuerdo con [BAR04] hoy en día contamos con la tecnología suficiente como para implementar Turbo-decodificadores capaces de producir su secuencia decodificada empleando un número considerable de iteraciones y un mínimo de tiempo en su proceso completo de decodificación, de hasta menos de 0.3μs por bit decodificado.

Para un número de iteraciones dado, el tipo de algoritmo de decodificación SISO que se utilice en el proceso de Turbo-decodificación también repercute de manera importante en el desempeño de un Turbo-código, como se mencionó líneas atrás. Como ejemplo de ello considérese la Figura 3.12, en la que se muestran las curvas de desempeño

obtenidas en [A.WOOD] luego de simular, para distintos tipos de algoritmos de decodificación SISO utilizados, un Turbo-código de tasa r = ½ que utiliza dos codificadores RSC idénticos, de tasa r = ½ y polinomios generadores $\mathbf{g}^{(a)}\mathbf{(D)} = 1 + D^2$ y $\mathbf{g}^{(b)}\mathbf{(D)} = 1 + D + D^2$, y un Interleaver "Impar-Par" de 1000 bits. Para estas simulaciones se empleó una modulación BPSK sobre un canal AWGN y 8 iteraciones en el proceso de Turbo-decodificación.

Nótese, a partir de la Figura 3.12, cómo el desempeño del Turbo-código depende del algoritmo de decodificación SISO que utilice, cumpliéndose lo previamente dicho en la sección 3.3.2: *los algoritmos Max-Log-MAP y SOVA degradan el desempeño de un Turbo-código con respecto a aquél que se obtiene de utilizar los algoritmos MAP y Log-MAP (cuyo desempeño es casi idéntico)*. A una BER = 10^{-4} esta degradación es de aproximadamente 0.1 dB para el algoritmo Max-Log-MAP y de alrededor de 0.6 dB para el algoritmo SOVA.

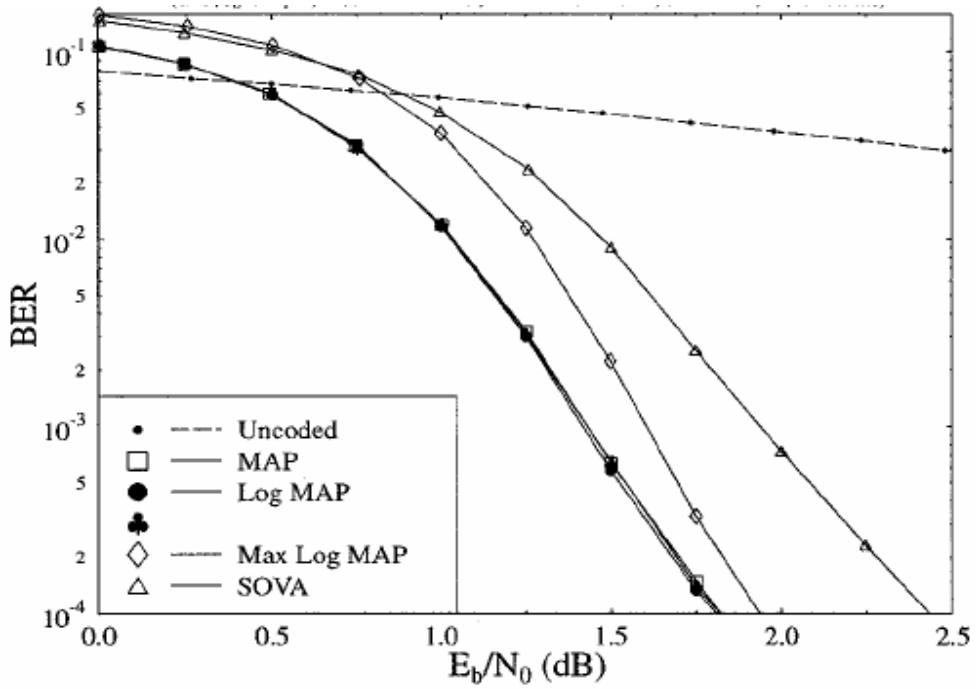

Figura 3.12. Desempeño de un T.C. en función del tipo de algoritmo de decodificación. [A.WOOD].

En [A.WOOD] se muestra que un Turbo-código de tasa $r = 1/3$ y un Turbo-código con un Interleaver de longitud $L = 169$ bits (ambos con las otras características iguales a las del Turbo-código simulado para obtener la Figura 3.12) presentan curvas de desempeño en las que se aprecia el mismo tipo de degradación. Sin embargo, es importante recordar que el desempeño de un algoritmo de decodificación SISO es proporcional a su complejidad (ver sección 3.3.2.).

Un último parámetro que podría considerarse importante por repercutir de alguna manera en el desempeño de un Turbo-código lo constituye la constraint length de los códigos RSC utilizados por el Turbo-codificador, aunque de acuerdo con [T.VAL99] "este parámetro no tiene una influencia significativa en el desempeño de un Turbo-código a una SNR baja. Por esta razón, los Turbo-códigos casi siempre utilizan códigos RSC simples, con una constraint length $3 \leq n_A \leq 5$. Sin embargo, para longitudes de *frame* suficientemente grandes, este parámetro puede ser un determinante secundario de la ubicación del *BER-floor* en la curva de desempeño de un Turbo-código." (Valenti, 1999: 46).

Como podemos ver existen varios parámetros que pueden repercutir de manera importante en el desempeño de un Turbo-código. La elección del tipo de cada uno de estos parámetros deberá hacerse de forma tal que se satisfagan los distintos requerimientos del sistema en consideración, pues como habremos podido notar dicha elección generalmente lleva consigo, no sólo ventajas, sino también desventajas que pueden resultar inaceptables para un sistema digital en particular.

3.5. Conclusiones del capítulo.

En este Capítulo presentamos una descripción completa del esquema de codificación para control de errores que revolucionó por completo este campo, al ofrecer la posibilidad tan buscada de ser capaces de transmitir a tasas muy cercanas a la capacidad del canal, con una probabilidad de error muy baja, empleando un mínimo de energía por bit de información para transmitir. Nos referimos a los Turbo-códigos.

Se mostraron las distintas arquitecturas que puede tener un codificador Turbo, siendo la más importante de éstas la concatenación paralela de códigos convolucionales (PCCC), pues es la arquitectura empleada por la gran mayoría de las aplicaciones que adoptaron a los Turbo-códigos como parte de sus estándares, entre ellas, las aplicaciones de telefonía celular de 3G.

En base a esto, se analizó el proceso de decodificación iterativa correspondiente a una arquitectura PCCC (las arquitecturas SCCC y HCCC también siguen un proceso de decodificación iterativo, muy similar en esencia al analizado en este capítulo, pero distinto en el aspecto de la estructura del decodificador, ya que los codificadores PCCC, SCCC y HCCC tienen una estructura distinta entre sí); este proceso constituye una parte fundamental en el gran desempeño que logra el esquema de codificación Turbo, pues su descubrimiento hizo posible el decodificado de palabras de código muy grandes, con una aproximación muy cercana al desempeño que ofrecería un algoritmo de decodificación de máxima probabilidad (MLDA), pero con una complejidad en su implementación no prohibitiva.

Así mismo, se describieron los distintos algoritmos de decodificación SISO que pueden ser empleados en el proceso de decodificación iterativa para decodificar las palabras de código enviadas por cada uno de los codificadores RSC que forman parte del Turbo-codificador.

Finalmente, se presentaron los distintos parámetros que pueden ser variados en un esquema de codificación Turbo en particular, así como los efectos que tales variaciones producen en el desempeño del mismo; algunos de estos parámetros son: el algoritmo de decodificación SISO utilizado, el número de iteraciones utilizado en el proceso de decodificado, la tasa de código utilizada en el proceso de codificado, y el tamaño y diseño de un elemento muy importante, constituyente de todo esquema de codificación Turbo: el Interleaver. Este componente se encuentra presente tanto en el Turbo-codificador como en el Turbo-decodificador, y su importancia radica en que brinda a las palabras de código generadas por un codificador Turbo, un "toque" de aleatoriedad que las hace más resistentes al ruido. En este capítulo, se presentaron los distintos tipos de Interleavers que más se utilizan en la Turbo-codificación.

Nos encontramos, pues, en un punto en el que nuestros conocimientos acerca de este revolucionario e innovador esquema de codificación, son bastante amplios como para comprender de manera clara la relevancia de su utilización en los estándares de telefonía celular de 3G, que estudiaremos en el capítulo siguiente.