UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRIA Y DOCTORADO EN INGENIERIA

FACULTAD DE INGENIERÍA

Algoritmos iterativos de control de errores para sistemas de transmisión de información con razón señal-ruido en la región de piso de ruido

T E S I S

QUE PARA OPTAR POR EL GRADO DE:

DOCTOR EN INGENIERIA

DOCTORADO EN INGENIERIA ELECTRICA - TELECOMUNICACIONES

P R E S E N T A :

Ing. Saul Lazcano Salas

TUTOR:

Dr. Francisco García Ugalde

2011

JURADO ASIGNADO:

Presidente: Dr. Boris Escalante Ramírez

Secretario: Dr. Federico Kulhmann Rodriguez

Vocal: Dr. Francisco Javier García Ugalde

1^{er} Suplente: Dr. Horacio Tapia Recillas

2^{do} Suplente: Dr. Rogelio Alcántara Silva

México, Distrito Federal, Ciudad Universitaria

TUTOR DE TESIS:

Dr. Francisco Javier García Ugalde

_____

FIRMA

DEDICATORIA

Difícil el poder elegir a quien se le dedica un trabajo tan importante por todo lo que representa, con el riesgo de olvidar a alguien. Mencionaré algunos nombres, buscando no excluir a todas aquellas personas que he tenido la buena fortuna de conocer en mi vida y que con su paso, han dejado una huella.

En primer término a mis hijas, Aitana y Mariana, enormes motores que me impulsan día a día y que a pesar de su corta edad, no dejan de darme lecciones y enseñanzas.

A ti Natalia, nuestros caminos se entrelazaron en el momento adecuado.

A todas aquellas personas que me tendieron una mano cuando tuve la rodilla en el piso, su ejemplo y actitud marcaron la diferencia.

A mi comité tutoral, excelentes profesionales pero aun mejores seres humanos. Gracias por sus consejos, ejemplo, enseñanzas y tiempo.

AGRADECIMIENTOS

A CONACyT, por su valioso apoyo para la realización y conclusión de este trabajo, como becario del programa de becas para estudios de Posgrado, así como en el proyecto 83694. Al programa PAPIIT IN-102410, por su apoyo recibido.

Índice general

Índice de figuras

Índice de cuadros

1. INTRODUCCIÓN

Sería difícil de imaginar la gran cantidad y diversidad de servicios y aplicaciones en las tecnologías de la información y las comunicaciones (TIC) sin los avances que se han dado en la electrónica (circuitos de alta escala de integración, microprocesadores, etc.), en las telecomunicaciones, así como en el área de procesamiento digital de señales. Todo esto ha ocasionado que en la actualidad, sistemas de comunicaciones, sistemas de almacenamiento de datos, y en general el procesamiento de datos se desarrolle en un contexto digital. En un sistema digital, la información es convertida en su representación binaria para ser procesada y codificada.

La información digital se procesa de manera diferente que la analógica. La estimación de señal se convierte en detección de la misma, por ejemplo: debido a sus niveles preestablecidos una señal digital es más fácil de recuperar en ambientes con ruido respecto a una señal analógica; de hecho, una señal digital puede ser recuperada de manera exacta si el ruido no sobrepasa un determinado nivel (con suficiente razón señal a ruido). Lo anterior permite recuperar los datos digitales mediante el uso de herramientas de teoría de la información y la codificación.

Manejar señales digitales tiene otras ventajas, principalmente lo referente a codificación, trans-codificación, compresión, cifrado, etc.; es mucho más simple un algoritmo para una señal digital que para una señal analógica, de hecho se tienen algoritmos que solo se pueden desarrollar en el ámbito digital, por ejemplo el control de errores y el cifrado de la información.

Dentro del procesamiento digital de datos, una de las piedras angulares se da con el trabajo de Nyquist en 1928 [1]. En ese trabajo, conocido como el teorema de muestreo, se establece que una señal en banda base que tiene como frecuencia máxima f_m $[Hz]$, puede ser reconstruída de manera exacta a partir de un número infinito de muestras, siempre que dichas muestras sean tomadas en intervalos uniformes de $T_s \leq \frac{1}{2f_m}$ segundos. Al valor T_s se le conoce como frecuencia de muestreo o frecuencia de Nyquist [2].

El siguiente gran aporte teórico se da en 1948, con el trabajo de Claude E. Shannon: *A mathematical theory of communications* [3]. Este trabajo significó el inicio de la teoría de la información y de la teoría de la codificación. En su trabajo, Shannon demuestra matemáticamente que en un canal de transmisión de datos bajo presencia de ruido, es posible lograr un número arbitrariamente bajo de errores siempre que la la tasa de transmisión sea menor a la capacidad del canal, demostrando así que existe un código de canal que puede lograr la tarea anterior y estableciendo además una cota teórica en la cantidad máxima de información relativamente libre de errores que puede enviarse en un canal con presencia de ruido.

A partir de ese trabajo se ha realizado un gran esfuerzo en la investigación de diversos esquemas de codificación buscando alcanzar el límite teórico de Shannon, dado que la demostración de Shannon fue hecha desde una perspectiva matemática, pero nunca menciona como se podría alcanzar dicho límite de manera práctica. En este sentido se logró un gran avance con el trabajo de Berrou, Glavieux y Thitimajshida en 1993, ellos

proponen un esquema de codificación de canal llamado turbo codificación [4], mediante el cual el límite de Shannon es casi alcanzado.

El presente capítulo presenta una introducción a los sistemas digitales de comunicaciones, hace una breve reseña de algunos tipos de codificadores de canal así como una introducción a los turbo códigos, parte central de esta tesis. Finalmente se presenta un bosquejo general de este trabajo, propósitos y objetivos.

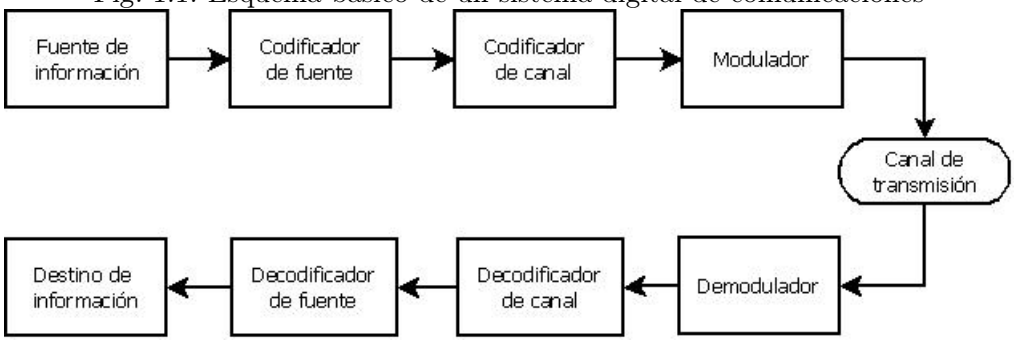
1.1. Sistemas digitales de comunicaciones

En la figura 1.1 [5] se muestra un esquema típico de un sistema digital de comunicaciones.

- El bloque conocido como codificador de fuente tiene esencialmente la tarea de representar adecuadamente la información binaria codificándola para lograr su compresión.
- La codificación de canal (esencia de esta tesis) tiene como objetivo reducir la probabilidad de error, P_e en el sistema, aunque como consecuencia aumenta la complejidad del decodificador, y aumenta las necesidades de ancho de banda requerido al agregar a la información a ser transmitida, redundancia de manera controlada y emplear dicha redundancia para reducir la P_e antes mencionada.
- La modulación se encarga de convertir el flujo de bits a la salida del codificador de canal, en formas de onda acopladas electromagnéticamente al medio de transmisión para que puedan ser transmitidas por el canal.

En la figura 1.1 existe una correspondencia entre los bloques del transmisor y del receptor, de manera que en este último se realizan las operaciones inversas.

Fig. 1.1: Esquema básico de un sistema digital de comunicaciones



Para los efectos de este trabajo de tesis, se asume que el canal se encuentra afectado por un proceso de ruido aditivo blanco gaussiano (additive white gaussian noise) AWGN.

1.2. Codificación de canal

La codificación de canal engloba la teoría y las técnicas empleadas para obtener una señal decodificada libre de errores en el receptor. Se puede tener un control de errores que se producen en el medio de transmisión por efectos de desvanecimiento de la señal, ruido

de diversas fuentes, multitrayectoria, etc. [6]. La codificación de canal juega un papel muy importante en sistemas tanto cableados como inalámbricos que requieren proteger la información contra los efectos no deseados antes mencionados.

Como se mencionó previamente, la teoría de la codificación de canal inicia formalmente a partir del trabajo de Shannon [3]. Uno de los resultados más importantes del trabajo de Shannon es el siguiente:

$$C = Wlog_2(1 + S/N)[bits/seg]. \tag{1.1}$$

En esta expresión, C representa la capacidad teórica de canal, es decir, la cantidad máxima de bits que pueden ser transmitidos en el medio de transmisión por unidad de tiempo. Depende del ancho de banda útil del sistema W y de la relación señal a ruido medida en el receptor S/N.

El segundo teorema de Shannon, conocido como el teorema de la codificación con ruido establece que se puede tener una comunicación confiable en un canal discreto sin memoria si la tasa de transmisión $R < C$; a tasas superiores a esa capacidad, la comunicación confiable no es posible. Desde el punto de vista de la codificación de canal este teorema implica que existe una manera de codificar tal que la probabilidad de error sea tan pequeña como se desee, siempre y cuando se respete la desigualdad antes expresada. La tarea principal de la investigación en teoría de códigos consiste en la búsqueda de esos esquemas de codificación, dado que el teorema de Shannon estableció su existencia, más no su definición. Esta teoría ha venido avanzando desde 1948, año en que se publicó el teorema hasta nuestros días, y un gran paso se dió en 1993 con los turbo códigos, estudio que motivó este trabajo de tesis.

De acuerdo con B. Sklar [6], las técnicas relacionadas con la codificación de canal se agrupan en dos tipos:

- Codificación por forma de onda: Engloba aquellas técnicas encaminadas a modificar la forma de onda de la señal original buscando que la señal resultante sea menos propensa a errores, mejorando la probabilidad de error (Pe) comparada con la señal original. Entre estas técnicas podemos encontrar la señalización M-aria, formas de onda ortogonales y modulación codificada en Trellis (trellis coded modulation) TCM.

- Codificación por secuencias estructuradas: Abarca las técnicas en las cuales se agrega redundancia de manera controlada a los datos originales, de modo de poder detectar y / o corregir los posibles errores en la señal recibida.

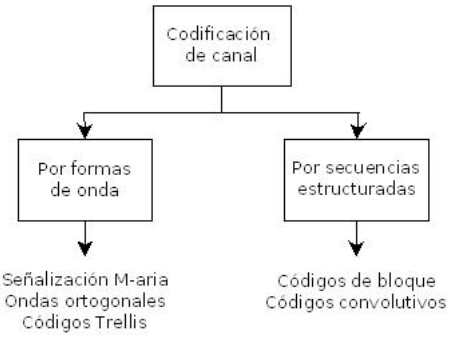
La codificación por secuencias estructuradas, de acuerdo a como es agregada la redundancia, puede ser subdividida en dos grandes grupos: códigos de bloque y códigos convolutivos [7].

En control de errores, tanto los códigos de bloque como los códigos convolutivos han tenido sus desarrollos teóricos y de aplicaciones, estas últimas dependen de los parámetros propios y de las condiciones de uso. Debido a que los primeros turbo códigos se diseñaron utilizando códigos convolutivos, en este trabajo nos dedicaremos más a este tipo de códigos.

1.2.1. Códigos concatenados

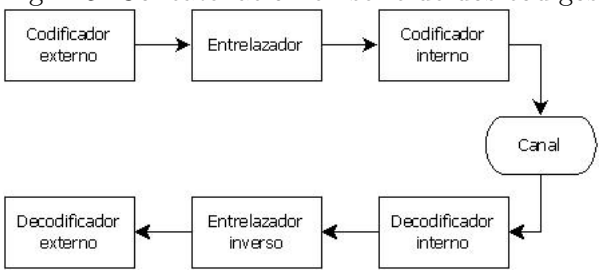
En 1966, G. D. Forney [8] analiza la idea de usar códigos concatenados, entendiendo el concepto de concatenación como la combinación de dos o más códigos, buscando combinar

Fig. 1.2: Clasificación básica de la codificación de canal



las mejores características de los códigos componentes, de tal forma que las deficiencias de un código puedan ser cubiertas con las ventajas del otro, y viceversa, logrando así un buen desempeño global con un grado de complejidad aceptable. En la figura 1.3 se muestra el ejemplo más común de concatenación de códigos en serie.

Fig. 1.3: Concatenación en serie de dos códigos



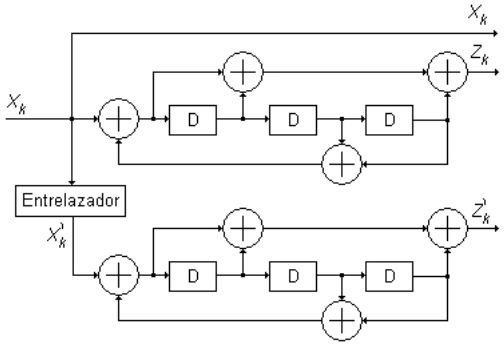
El código interno es el que tiene contacto directo con el modulador - demodulador y con el canal mismo. Este código se configura de modo tal de poder corregir la mayor cantidad de errores debidos al canal. El código externo generalmente es un código de baja redundancia (alta tasa de codificación), el cual se encarga de corregir posibles errores que el código interno pudiese no haber corregido, llevando de este modo la probabilidad de error (Pe) a un nivel más bajo. El papel del entrelazador conectando entre los dos codificadores es dispersar posibles errores en ráfaga que pudiesen presentarse en la salida del codificador interno.

Una de las principales razones para emplear códigos concatenados es que se puede lograr una tasa de errores baja, manteniendo la complejidad del decodificador considerablemente menor comparada con el caso de un solo código necesario para alcanzar el mismo desempeño. La configuración más utilizada y conocida actualmente consiste en emplear un código convolutivo como codificador interno, en conjunto con un código de bloque de Reed-Solomon (RS) como código externo. Esta combinación ha sido ampliamente utilizada por la NASA en sus sondas espaciales [7].

La concatenación en serie no es la única forma de concatenar códigos. En 1993 C. Berrou, A. Glavieux y P. Thitimajshida [4] proponen una concatenación en paralelo de dos códigos convolutivos recursivos sistemáticos (Códigos RSC) conectados a través de un entrelazador, como se muestra en la figura 1.4.

A esta concatenación en paralelo de dos o más códigos, conectados a través de uno o más entrelazadores, se le conoce con el nombre de turbo códigos. Gran parte de la

Fig. 1.4: Estructura de un turbo codificador



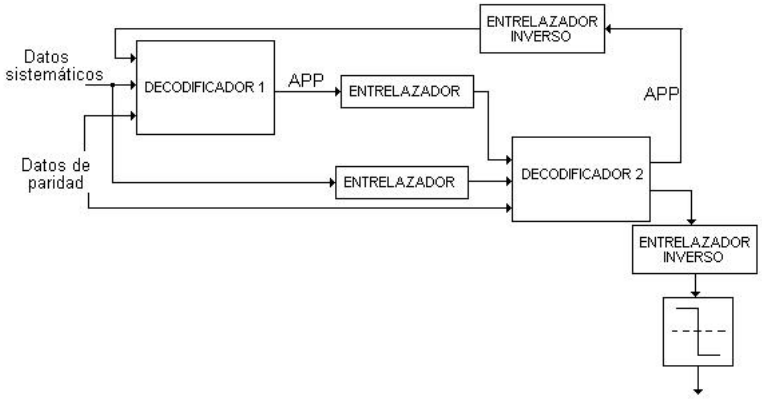
aceptación y éxito de los turbo códigos se debe a que su desempeño es muy cercano al esperado teórico predicho por Shannon [3]. Pueden alcanzar una tasa de bit en error (BER) de 10^{-5} para una $\frac{E_b}{N_0}$ de 0,7 dB con una eficiencia espectral de 0,5 bits/s/Hz [5].

Debido a la presencia del entrelazador, realizar una decodificación mediante un algoritmo de máxima verosilimitud (maximum likelihood) *ML* se vuelve una tarea complicada (equivalente a realizar la decodificación de un código de bloque del mismo tamaño que el entrelazador) y por consiguiente no es práctica.

Para resolver este problema, se opta por dividir la decodificación en bloques (cada bloque decodificado corresponde a cada uno de los codificadores que componen el codificador), en donde cada bloque decodificador tiene su solución óptima y realiza un intercambio de información con los otros bloques en un esquema iterativo, como se observa en la figura 1.5.

Este esquema de decodificación iterativa es justamente el que le da el nombre a los "turbo códigos" ya que se asemeja al comportamiento de una "máquina turbo", en donde los gases de escape de la máquina (energía residual) son retroalimentados a la entrada para hacer más eficiente el ciclo de combustión de la misma. Se debe considerar que en este esquema iterativo, la ganancia en desempeño obtenida en cada iteración disminuye gradualmente en relación con la iteración previa.

Fig. 1.5: Estructura de un turbo decodificador



1.3. Resumen de la tesis

En un contexto teórico-práctico, el objetivo primordial de este estudio es analizar el comportamiento de un turbo código bajo condiciones de razón SNR alta, empleando tamaños de entrelazador pequeños (por restricciones prácticas de tiempo de procesamiento, en telefonía celular por ejemplo). Analizando el impacto en el desempeño de la tasa de bits en error (BER) que tienen los diferentes elementos que constituyen un turbo codificador, bajo las condiciones antes citadas. Como resultado de este estudio, se propone una metodología hasta ahora no encontrada en la literatura científica especializada, para la construcción de un turbo codificador para tramas pequeñas, haciendo énfasis en la elección de los elementos del turbo codificador de manera ligada, de modo que se busque mejorar el desempeño global, en términos de BER del sistema en conjunto, para condiciones de perforado y sin ellas. El interés de considerar una SNR alta radica en que se ha encontrado que los turbo códigos disminuyen su buen rendimiento en esta región. Por otro lado el perforado de un código constituye una parte importante de los códigos convolutivos porque permite a partir de un código original de una cierta tasa de codificación, obtener otro de tasa diferente, manteniendo casi la misma complejidad en el decodificador. En estas condiciones el esquema global de nuestro esquema de codificación constará del entrelazador, del código turbo y de la matriz de perforado. La metodología de diseño propuesta en esta tesis considera estos tres elementos constitutivos y hace una optimización global en términos de probabilidad de bit en error BER. De ahí su carácter innovador.

En el capítulo 1 se presenta un bosquejo general de la codificación de canal, el capítulo 2 se enfoca al proceso de la turbo codificación y los elementos inmersos en la misma, el capítulo 3 analiza el proceso de la turbo decodificación, proceso clave para el excelente desempeño en términos de su capacidad de corrección de errores de los turbo códigos, el capítulo 4 aborda el tema del perforado en los turbo códigos, en el capítulo 5 se propone una metodología para la construcción ligada de un turbo codificador y finalmente, en el capítulo 6 se muestran las conclusiones y resultados del presente trabajo de tesis.

2. TURBO CODIFICACIÓN

En este capítulo se presentan antecedentes y algunas definiciones necesarios para comprender mejor la codificación de canal en general. Se analiza también de manera detallada los componentes de un turbo codificador y su efecto al evaluar el desempeño del turbo codificador en términos del BER.

2.1. Introducción

Dentro de la codificación de canal, una característica muy deseada en cualquier código es la linealidad, ya que gracias a su estructura algebráica, los procesos de codificación y decodificación son más simples de describir comparados con los no lineales [9]. Considerando que los turbo códigos son códigos lineales, en esta sección se analizan de manera particular características y propiedades válidas para códigos lineales.

2.1.1. Codificación

El proceso de codificación puede entenderse como un mapeo unívoco de un conjunto A, cuyos elementos o símbolos son información de longitud k: $i_0, i_1, \ldots, i_{k-1}$, hacia otro conjunto B, con elementos denominados palabras del código, de longitud n: $c_0, c_1, \ldots, c_{n-1}$, en donde $(n > k)$ y la diferencia $(n - k)$ es la redundancia que se le añade a la palabra codificada.

Dadas dos operaciones binarias, denominadas como suma $(+)$ y producto $(.)$, los elementos de un código lineal pertenecen a un alfabeto, mismo que constituye un grupo finito, en donde un grupo se puede definir como una estructura algebráica que cumple con los siguientes axiomas [10]:

- Cerradura: Cualquier operación que se aplique a dos elementos del grupo, dará como resultado un tercer elemento que también pertenece al grupo.
- Ley asociativa: Para tres elementos cualquiera a, b, c del grupo, se cumple que $(a + b) + c = a + (b + c)$ para el caso de la suma y $(a.b).c = a.(b.c)$ para el caso del producto.
- Existencia de un elemento neutro: Para el caso de la suma, existe un elemento neutro, 0, tal que $a + 0 = 0 + a = a$. Para el caso del producto existe su elemento neutro 1 tal que a,$1 = 1.a = a$.
- Existencia de un elemento inverso: Para el caso de la suma, existe un elemento inverso, $-a$, tal que se cumple que $a + (-a) = (-a) + a = 0$. Para el caso del producto, existe su elemento neutro, a^{-1}, tal que se cumple que $a.a^{-1} = a^{-1}.a = 1$.

En resumen, un código lineal se define como aquel en donde el resultado de cualquier combinación lineal de sus palabras código es a su vez una palabra del código [11].

Por su relación con los sistemas de transmisión de información en los sistemas de cómputo, en este trabajo nos enfocaremos en códigos lineales binarios, es decir, las palabras código se constituyen de n elementos binarios. El proceso de codificación se puede ver como una función biunívoca en donde a cada elemento del conjunto información le corresponde un elemento del conjunto palabras codificadas y viceversa.

La construcción de un buen código es una tarea difícil, ya que mientras un código lineal tiene 2^k palabras código y el conjunto de palabras codificadas posee 2^n elementos, donde $2^n >> 2^k$, lo cual implica que se deben seleccionar únicamente 2^k palabras de entre las 2^n existentes, manteniendo la característica de linealidad además de que las palabras código seleccionadas deben estar lo mas distantes posibles, tomando como métrica la distancia de Hamming.

Algunas métricas importantes [11] para analizar un código son:

Peso de Hamming: El peso de Hamming de un vector $\mathbf{c}$ se define como el número de coordenadas diferentes de cero, variando desde cero hasta la longitud del vector.

$$wt(\mathbf{c}) = \sum_{j=0}^{n-1} wt(c_j), \text{ en donde } wt(c_j) = \begin{cases} 0 & si \quad c_j = 0 \\ 1 & si \quad c_j \neq 0 \end{cases} \tag{2.1}$$

Distancia de Hamming: La distancia de Hamming entre dos vectores $\mathbf{c}$ y $\mathbf{d}$ es el número de coordenadas en donde $\mathbf{c}$ y $\mathbf{d}$ no coinciden.

$$dist(\mathbf{c},\mathbf{d}) = \sum_{j=0}^{n-1} wt(c_j + d_j), \text{ en donde } wt(c_j + d_j) = \begin{cases} 0 & si \quad c_j = d_j \\ 1 & si \quad c_j \neq d_j \end{cases} \tag{2.2}$$

Distancia mínima: La distancia mínima d_{min} de un código $\mathbf{C}$, es la menor distancia de Hamming entre dos palabras código cualesquiera.

$$d_{min} = min\left\{wt(\mathbf{c} + \mathbf{d})\right\} = min_{\mathbf{c}\neq 0}\left\{wt(\mathbf{c})\right\} \tag{2.3}$$

Ponderación de peso (*weight enumeration*): La ponderación de peso para un código $\mathbf{C}$ de longitud n, se puede definir como un vector $\mathbf{W} = (w_0, w_1, \ldots, w_{n-1})$, en donde la j-ésima coordenada representa el número de palabras de código de peso j.

Probabilidad de bit en error (P_e) : La probabilidad de bit en error es igual a la probabilidad de que un bit transmitido sea recibido con error.

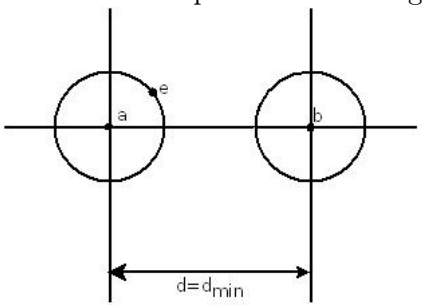
Probabilidad de bloque en error (P_b): La probabilidad de bloque en error es igual a la probabilidad de que una palabra del código transmitida no corresponda con la palabra del código decodificada en el receptor.

Para los códigos de bloque, la distancia mínima (d_{min}), es igual al peso de Hamming de la palabra de menor peso del código C [9], excepto la palabra nula, con lo que se vuelve una importante herramienta en el diseño de códigos dado que es más rápido medir pesos que distancias; por lo que, evaluar la distancia mínima mediante la palabra de menor peso de Hamming es mucho más simple que evaluarla mediante algún algoritmo computacional.

La distancia mínima es una medida relacionada con la capacidad de detección y/o corrección de un código C, la distancia mínima corresponde al evento con el menor número de errores alrededor de una palabra del código y por consiguiente bajo el criterio de

máxima verosimilitud *ML*, al evento con la mayor probabilidad de ocurrencia, ya que se asume que el evento con menor número de posibles errores es el más probable de ocurrir y de esta manera, poder realizar el mapeo de la palabra recibida hacia la palabra del código más cercana [11]. Lo anterior puede explicarse de una manera mas simple con base en la figura 2.1.

Fig. 2.1: Esfera de palabras del código C



Sean dos palabras $a, b \in C$ separadas por una distancia igual a la d_{min} y sea la palabra recibida con error r. La d_{min} existente entre las palabras a y b permite que la palabra e sea identificada como una palabra del código ya que la palabra a es la palabra del código válida más cercana.

El número de errores, e, que un código C puede corregir se puede obtener a partir de la distancia libre de la siguiente manera: $e \leq \left\lceil \frac{d_{min}}{2} \right\rceil$.

Si se conoce la distancia mínima de un código lineal C, la misma se incluye para caracterizar al código dado, quedando de la siguiente manera: $C(n, k, d_{min})$.

Las dos maneras más comunes de representar un código lineal C son mediante una matriz de chequeo de paridad y una matriz generadora.

La matriz generadora para un código $C(n, k)$ es una matriz G de dimensión k x n cuyos renglones forman una base para C. No hay una única matriz generadora para un código. Por lo general, se acostumbra expresar la matriz G de modo que las k primeras columnas (deben ser linealmente independientes) se rearreglen y agrupen de modo de tener una matriz identidad de dimensiones k x k. Si se tiene ese caso, se dice entonces que el código es sistemático, dichas primeras k columnas corresponden a los k bits de información y las restantes $r = n - k$ columnas representan la redundancia del código C, además de que entonces, la matriz G, ahora en la forma $[I_k|A]$ es única para el código C [11], en donde I_k es la matriz identidad de tamaño k x k formada anteriormente.

A partir de la matriz A antes mencionada, se puede construir una matriz H, llamada matriz de verificación de paridad, la cual se construye mediante la expresión: $H = [-A^T|I_{n-k}]$, en donde la matriz A^T es la transpuesta de la matriz A. Una de las características más importantes de la matriz H es que siendo ortogonal a G, cualquier palabra del código $\mathbf{c}$ multiplicada por la matriz de paridad H, da por resultado el vector nulo. De este modo, es posible utilizar la matriz de verificación de paridad para cerciorarse que una determinada palabra recibida pertenezca, o no, a una palabra del código $C = [\mathbf{c} \in \mathbb{Z}_2^n | H\mathbf{c}^T = \mathbf{0}]$ [11].

2.1.2. Códigos convolutivos

Los códigos convolutivos o de convolución, fueron definidos en 1955 por Elias [12], en la actualidad se encuentran implementados en una gran gama de aplicaciones que van

desde comunicaciones inalámbricas, comunicaciones satelitales y espaciales, dispositivos de almacenamiento, televisión de alta definición, por mencionar algunos ejemplos. También es posible encontrarlos como parte de esquemas de codificación concatenados, tanto en serie como en paralelo; esta última configuración corresponde a los turbo códigos.

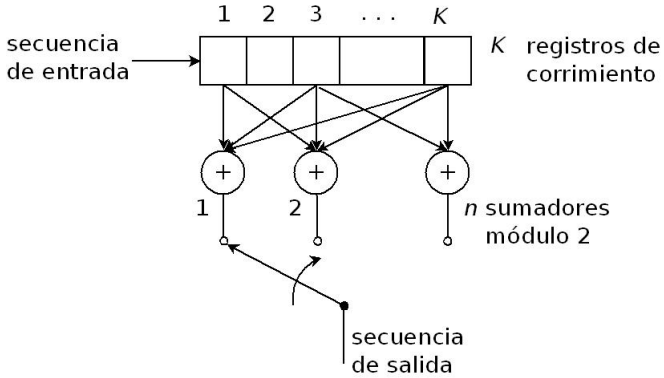
Mientras los códigos de bloque trabajan con palabras de longitud fija (k), los códigos convolutivos trabajan con flujo continuo de bits, con lo cual, los parámetros (n, k) tienen un significado diferente, no así la relación $R = k/n$ que sigue manteniendo el mismo significado de la tasa del código.

Otra importante diferencia es que los códigos convolutivos son códigos con memoria, es decir, la salida en un instante determinado no sólo depende de la entrada en ese instante, sino de las $(K - 1)$ entradas previas. Al parámetro K se le conoce como longitud de restricción *(constraint lenght)* y representa el número de registros de corrimiento, uno por cada bit de información [7]. El parámetro k define el número de bits a la entrada del codificador en un periodo de reloj, mientras que el parámetro n define el número de sumadores módulo 2 y correspondiente número de bits de salida.

Así como en los códigos de bloque, una importante métrica para medir el desempeño de los mismos es la distancia mínima, en los códigos convolutivos también se analiza la distancia de Hamming más pequeña entre cualquiera de las palabras del código. A esta distancia se le conoce como distancia libre (d_{free}). Una manera rápida para poderla calcular es seleccionando la secuencia nula como la secuencia de información a ser codificada, se introduce de manera arbitraria un error en dicha secuencia para que en el decodificador se tenga una secuencia diferente a la secuencia nula. La d_{free} es el peso de Hamming de la secuencia mas corta que, partiendo del estado nulo y en presencia del error, converge nuevamente al estado nulo dentro del enrejado trellis.

Sin perder generalidad, para este trabajo de tesis consideraremos únicamente códigos convolutivos con tasas del tipo $R = 1/n$ binarios, ya que son de los códigos convolutivos más empleados y son además de los considerados para construir un turbo codificador. Un codificador convolutivo típico se muestra en la figura 2.2 [6].

Fig. 2.2: Esquema general de un codificador convolutivo

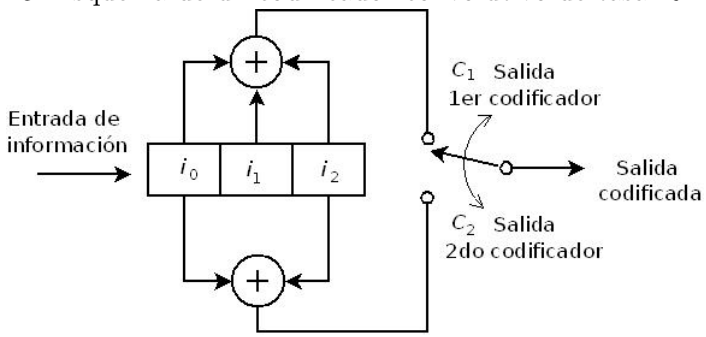


2.1.3. Representación de un código convolutivo

Una de las formas más comunes de representar un código es mediante un diagrama de conexiones. La figura 2.3 muestra un código convolutivo con $K = 3$, $n = 2$, $R = 1/2$.

En la figura 2.3, en cada ciclo de reloj se tiene un bit de entrada de la secuencia de información $\mathbf{i} = i_0, i_1, ...$ en el extremo izquierdo del circuito, mientras que el resto de los

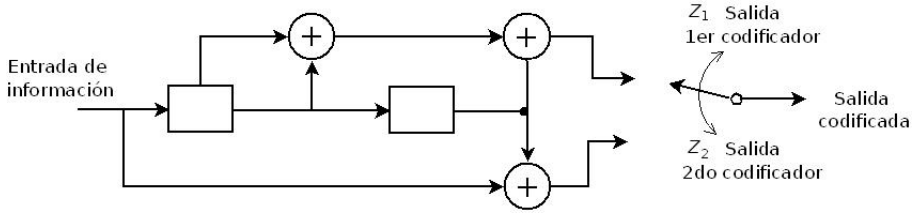
Fig. 2.3: Esquema de un codificador convolutivo de tasa $R = 1/2$.



bits se recorre una posición a la derecha. La salida c_1, c_2 se obtiene al operar cada casilla con los sumadores módulo 2 (cada sumador indica con que casillas se opera). El primer bit de salida c_1 corresponde a la salida del primer sumador módulo 2 el cual se obtiene de la siguiente manera: $c_1 = b_0 \oplus b_1 \oplus b_2$, de manera similar, el segundo bit de salida c_2 corresponde a la salida del segundo sumador módulo 2; $c_2 = b_0 \oplus b_2$. En el apéndice B de este trabajo de tesis, se puede consultar un breve resumen sobre la aritmética de enteros, más conocida como enteros modulares. En el mismo se dan tablas para la operación $\oplus$ entre enteros módulo 2.

La memoria de un código convolutivo está dada por $m{=}(K-1)$, en donde m es la longitud total de los registros de corrimiento. En la figura 2.4 se muestra la representación del codificador de la figura 2.3 mediante registros de corrimiento.

Fig. 2.4: Representación por registros de corrimiento de un codificador convolutivo.



El código convolutivo es determinado por las conexiones entre los estados y los sumadores, cualquier cambio en dichas conexiones conlleva a construir un código diferente [6]. Estas conexiones se especifican mediante un vector binario de tamaño m para cada sumador. En este vector, un valor "1" en la posición i representa la conexión del i-ésimo estado con el respectivo sumador. Para el ejemplo de las figuras 2.3 y 2.4, sus respectivos vectores son: $g_1 = 111$ y $g_2 = 101$. Con estos vectores binarios las conexiones de los codificadores pueden ser también descritas en forma polinomial, mediante los polinomios generadores donde cada bit del vector es un coeficiente del polinomio correspondiente. Al igual que la representación vectorial, se emplea un polinomio por cada sumador módulo 2, y los coeficientes de cada término ("1" o "0") representan si existe, o no, conexión entre el sumador y el registro respectivo. El grado de cada uno de estos polinomios es como máximo de $(K-1)$. El término de menor grado del polinomio corresponde con la entrada del codificador. Para nuestro ejemplo, los polinomios generadores, en términos de la variable D *(delay)* son:

$$g_1(D) = 1 + D + D^2$$
$$g_2(D) = 1 + D^2$$

Un codificador convolutivo es un sistema lineal, invariante en el dominio del tiempo. A partir de su respuesta a impulso, es posible calcular la salida a cualquier entrada como una superposición de respuestas a impulso, es decir, como una suma lineal de las respuestas a impulso defasadas. A partir de esto, se comprueba que un código convolutivo es lineal.

Otra forma de calcular la salida **z** para una determinada secuencia de entrada **u** es [7]:

$$z^{(j)}[i] = \sum_{l=0}^{m} u[i-l]g_j[l] \tag{2.4}$$

Lo anterior para $0 \leq j \leq n$.

La ecuación anterior equivale a decir que la salida **z** corresponde a la convolución discreta entre la secuencia de entrada $\mathbf{u}(D)$ y cada uno de los polinomios generadores $g(D)$, de ahí el nombre de códigos convolutivos o de convolución.

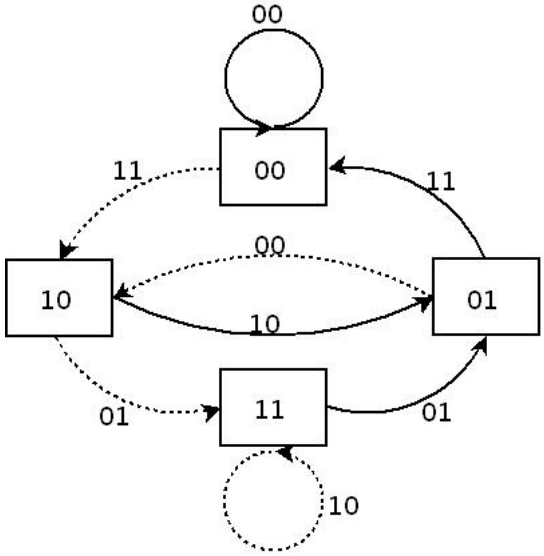
La ecuación antes citada puede ser escrita en forma matricial como: $\overline{z} = \overline{u}G$, en donde la matriz G es la matriz generadora del código convolutivo. Para un código convolutivo con $R = 1/2$ de memoria m se tiene que la matriz G está dada por [7]:

$$G = \begin{pmatrix} g_0[0]g_1[0] & \cdots & g_0[m]g_1[m] & & \\ & g_0[0]g_1[0] & \cdots & g_0[m]g_1[m] & \\ & & g_0[0]g_1[0] & \cdots & g_0[0]g_1[0] \\ & & \ddots & & \ddots \end{pmatrix} \tag{2.5}$$

Otra de las formas más útiles de representar un código convolutivo es mediante su diagrama de estados. Un código convolutivo puede ser visto como una máquina de estados finitos; en donde un estado se entiende como la cantidad mínima de información dentro de la máquina que, en conjunto con la entrada actual, son necesarios para describir la salida. Se tiene 2^m posibles estados, correspondientes al posible contenido de las $(K-1)$ casillas más significativas (a la izquierda del registro de corrimiento). Dado que se habla de un código convolutivo binario, se tiene para cada estado dos posibles entradas y dos salidas. Las ramas que conectan los diferentes estados representan las dos posibles entradas; por convención, una rama continua representa una entrada "0" mientras que una rama discontinua representa una entrada "1". La salida del codificador se representa mediante una etiqueta en la parte superior de cada posible entrada. El diagrama de estados para el codificador ejemplo es el mostrado en la figura 2.5 [6].

Aunque el diagrama de estados caracteriza adecuadamente un codificador convolutivo, presenta la desventaja de no poder realizar fácilmente un seguimiento de las transiciones del codificador a lo largo del tiempo; el diagrama de estados carece explícitamente de memoria para poder cotejar dichas transiciones. Para solucionar lo anterior, se puede emplear el diagrama de árbol, el cual añade la variable tiempo al diagrama de estados. Cada transición en el diagrama de estados se representa como una transición de izquierda a derecha en el árbol, en donde cada rama del diagrama de árbol describe la correspondiente salida. La entrada al codificador corresponde a cada rama, en donde por convención una rama hacia arriba representa una entrada al codificador "0″ mientras que una rama hacia abajo representa una entrada "1″. Se asume que en el codificador, el estado inicial es cero.

Fig. 2.5: Diagrama de estados para el codificador $g_1 = 111$, $g_2 = 101$

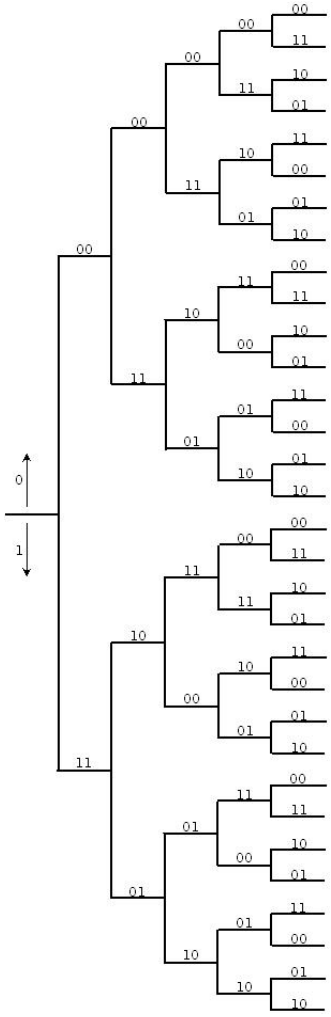


A su vez, una desventaja que el diagrama de árbol conlleva es que el tamaño y el número de ramas del mismo crece a una razón de 2^n, donde n representa el n-ésimo instante de tiempo, con lo cual se vuelve poco práctico. En la figura 2.6 se puede observar como crece el tamaño del árbol dados algunos instantes de tiempo. Para solucionar lo anterior, existe la representación por diagrama de enrejado "trellis", el cual explota la simetría del diagrama de árbol debida a la memoria limitada del circuito codificador, para tener entonces una representación lineal que es fácilmente manejable. Dentro del enrejado trellis, los nodos representan los posibles estados del codificador, por convención una línea continua representa una entrada "0" mientras que una línea discontinua representa una entrada "1", (similar a la convención seguida en el diagrama de estados); en cada ciclo de reloj, el enrejado de trellis requiere 2^{k-1} nodos para representar cada uno de los 2^{k-1} estados. Después de K unidades de tiempo, el enrejado trellis alcanza una regularidad y a partir de este punto la estructura se repite; cada estado se puede identificar a partir de las $K-1$ entradas precedentes.

2.1.4. Códigos catastróficos

Existe un tipo de códigos llamados catastróficos los cuales deben ser evitados ya que generan un número infinito de errores en la decodificación a partir de un número finito de errores en la secuencia recibida por el decodificador, o dicho de otra forma, una secuencia de información de peso infinito genera una secuencia codificada de peso finito. Una manera rápida de identificar un código catastrófico es a través de su diagrama de estados; un código catastrófico presenta en su diagrama de estados un ciclo, en el cual una secuencia no nula de información está asociada a una salida de peso nulo. Otra forma de identificar un código catastrófico es a través de sus polinomios generadores, un código de tasa $R = 1/n$ es catastrófico si entre sus polinomios generadores, existe un factor común diferente de la unidad o dicho de otro modo, si el máximo común divisor de sus polinomios generadores es diferente de la unidad [13]. Un código convolutivo sistemático nunca sera catastrófico [14], sea por ejemplo el código con polinomios $g_1(D) = 1 + D$ y $g_2(D) = 1 + D^2$, entre estos polinomios existe el factor común $1 + D$, ya que $1 + D^2 = (1 + D)(1 + D)$, con lo

Fig. 2.6: Diagrama de árbol para el codificador $g_1 = 111$, $g_2 = 101$



que el código asociado a dichos polinomios es un código catastrófico.

2.1.5. Códigos sistemáticos

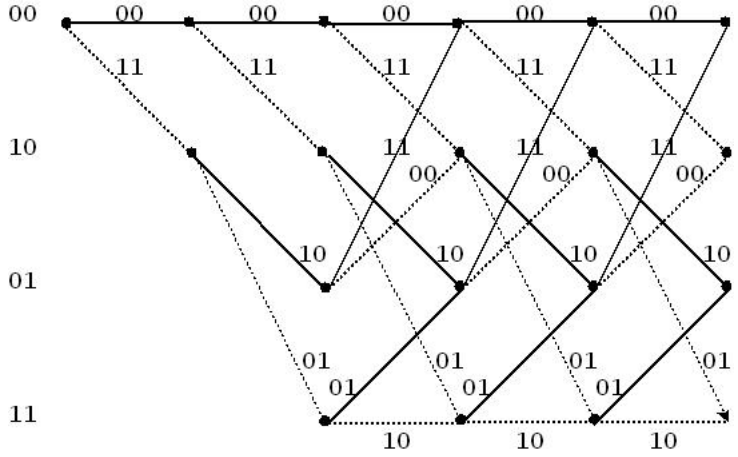
Otro tipo particular de códigos convolutivos son los llamados códigos sistemáticos, cuya principal característica es que la secuencia de entrada **u** forma parte de la palabra codificada **z**. Como se vió en la sección 2,1,1, se acostumbra escribir la matriz generadora de este tipo de códigos de la siguiente manera: $G = [A|I_k]$, en donde I_k es la matriz identidad de tamaño k x k.

El turbo codificador originalmente propuesto en [4] está compuesto por dos codificadores convolutivos recursivos y sistemáticos (códigos RSC), conectados en paralelo a través de un entrelazador. En la siguiente sección se analizan estos códigos y su desempeño.

2.1.6. Códigos convolutivos recursivos sistemáticos

Aunque un código sistemático presenta por lo general, una mejor distancia libre (la distancia libre de un código convolutivo, es el equivalente de la distancia mínima de un código de bloque) y por consiguiente, un mejor desempeño en términos de BER, comparado con los códigos no sistemáticos [14], en el trabajo antes citado se presenta un

Fig. 2.7: Enrejado trellis para el codificador $g_1 = 111$, $g_2 = 101$



método para la construcción de códigos sistemáticos a partir de códigos no sistemáticos con distancia libre muy similares. Sin embargo, este método permite tener códigos RSC que presentan un desempeño, en términos de BER, ligeramente inferior a los códigos no sistemáticos, a partir de los cuales fueron construidos (en [14] se menciona que esta diferencia se debe a que, aunque la distancia libre es muy similar, la ponderación de pesos de las palabras código resulta mayor), pero con la ventaja de que para razones señal a ruido SNR bajas, los códigos RSC presentan un mejor desempeño.

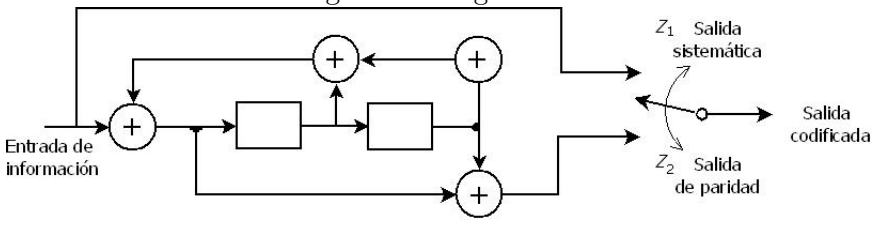
Un código RSC posee una matriz generadora G de la forma:

$$G = \left(\begin{array}{cccc} 1 & \frac{\bar{g}_1(D)}{\bar{g}_0(D)} & \cdots & \frac{\bar{g}_1(D)}{\bar{g}_0(D)}. \end{array} \right) \tag{2.6}$$

A partir de esta matriz y al igual que un código convolutivo no sistemático no recursivo, la palabra codificada se puede obtener de la siguiente manera [7]: $\overline{z} = \overline{u}'G$, en donde $\overline{u}' = g_0(D)\overline{u}$. De este modo, los codificadores convolutivos no sistemáticos y sistemáticos producen las mismas secuencias codificadas, pero en los códigos RSC la secuencia de información es afectada en su orden por el polinomio $g_0(D)$.

En la figura 2.8 se muestra el esquema de un código RSC de tasa $R = 1/2$, $K = 3$ y $G = [1, (1 + D^2)/(1 + D + D^2)]$.

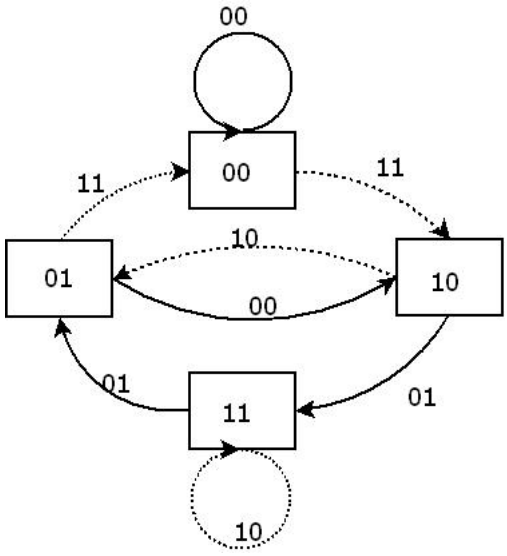
Fig. 2.8: Código RSC.



La figura 2.9 muestra el diagrama de estados para el codificador de la figura 2.8. Se sigue la misma convención antes usada; una rama continua representa una entrada al codificador "0" mientras que una rama discontinua representa una entrada "1".

La estructura del enrejado trellis es la misma que para un código no sistemático con polinomios $g_1 = 111$ y $g_2 = 101$, aunque en este caso, la relación entre los bits de entrada y de salida es diferente.

Fig. 2.9: Diagrama de estados para el codificador RSC.



2.2. Turbo codificación

El turbo codificador propuesto en [4] está compuesto por dos codificadores RSC idénticos, conectados por un entrelazador, como se muestra en la figura 1.4. La salida global del codificador se compone de las salidas de paridad de ambos codificadores y la salida sistemática del primer codificador; la salida sistemática del segundo codificador no se considera dado que es una versión entrelazada de la salida sistemática del primer codificador. De este modo, la tasa de codificación del turbo codificador es de $R = 1/3$. A esta concatenación de códigos se le conoce como concatenación en paralelo debido a que ambos codificadores trabajan con la misma secuencia de entrada (el segundo recibe una versión entrelazada de dicha secuencia). Por el contrario cuando un codificador trabaja con la salida del otro se tiene la concatenación en serie. Debido a la forma en que están conectados los codificadores, a los turbo códigos también se les conoce como concatenación en paralelo de códigos convolutivos *PCCC* (Parallel Concatenated Convolutional Codes) [5].

Cuando la aplicación requiere códigos con tasas superiores se tienen dos posibles alternativas:

- Selección de codificadores de tasa $R = k/n$, $k > 1$: esta solución conlleva la desventaja de que el enrejado *Trellis* asociado tiene un mayor número de ramificaciones 2^k, es decir, es más robusto y por consiguiente, el proceso de decodificación es más complejo y lento. Razón por la cual esta opción rara vez se adopta.

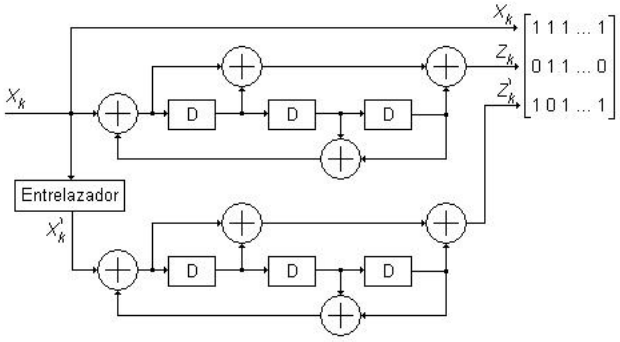
- Perforado: El perforado se puede definir como el proceso mediante el cual se elimina de manera sistemática ciertos bits del flujo de salida del codificador. El número de bits eliminados o borrados modifican la tasa R de codificación. Esta opción es comúnmente utilizada.

En el perforado la ubicación de los bits a ser eliminados puede especificarse mediante un arreglo matricial en el cual una posición marcada con un "1" indica un bit que será transmitido, mientras que una posición marcada con un "0" corresponde a una bit que será eliminado (perforado). Este arreglo matricial, mejor conocido como matriz de perforado

o patrón de perforado, corresponde a una matriz de n x p, donde p es conocido como el periodo de la matriz de perforado.

Para el caso de los turbo códigos, este proceso de perforado se agrega por medio de una matriz a la salida del turbo codificador. En la figura 2.10 se muestra un turbo codificador con una matriz de perforado agregada a la salida del mismo.

Fig. 2.10: Turbo codificador con matriz de perforado.



En la figura 2.10, cada columna de la matriz corresponde a un tiempo de entrada de un bit, y cada renglón está asociado a uno de los bits de salida (sistemático y de paridad) del turbo codificador. Así, el primer renglón corresponde al perforado de la salida sistemática; el segundo renglón corresponde a la salida de paridad del primer codificador y finalmente, el tercer renglón corresponde a la salida de paridad del segundo codificador. La construcción de la matriz de perforado se analiza en el capítulo 4 de este trabajo de tesis.

2.2.1. Análisis del desempeño de un turbo código

La elección de los elementos de un turbo codificador no se recomienda se realice de manera aleatoria, y de hecho es un proceso bastante complicado porque hace intervenir varios parámetros (de los códigos, del entrelazador y de la matriz de perforado), es por tanto recomendable seguir una metodología para de este modo poder garantizar que el turbo código construido posea un buen desempeño. Es esta metodología la parte fundamental de este trabajo de tesis y para poder diseñarla y justificarla, es necesario conocer como evaluar el desempeño de un turbo codificador.

Una de las tareas mas importantes del entrelazador es construir un código de bloque a partir de códigos RSC, con lo cual para efectos de analizar su desempeño, un turbo código es un código de bloque [15].

Considerando un canal con ruido *AWGN*, para un código de bloque la P_e queda acotada de acuerdo a la siguiente ecuación:

$$P_e \leq \sum_{d=d_{free}} B_d Q\left(\sqrt{2dr\frac{E_b}{N_0}}\right). \tag{2.7}$$

Generalmente, se toma la ecuación 2.7 como una cota superior para el cálculo de la P_e, con lo cual la ecuación 2.7 queda como:

$$P_e = \sum_{d=d_{free}} B_d Q\left(\sqrt{2dr\frac{E_b}{N_0}}\right). \tag{2.8}$$

En donde la función Q está definida por:

$$Q(z) = \frac{1}{\sqrt{2\pi}} \int_z^\infty e^{-x^2/2} dx. \tag{2.9}$$

E_b es la energía promedio por bit, N_0 es la densidad espectral de potencia unilateral del ruido, y las parejas (d, B_d) corresponden al espectro de distancias, también conocido como polinomio de ponderación de pesos del turbo código.

El cálculo del polinomio de ponderación de pesos del turbo código, se hace con base en la siguiente ecuación:

$$B_d = \sum_{d=w+z} \frac{w}{N} A_{w,z} = \sum_{d=w+z_1+z_2} \frac{w}{N} \frac{A^c_{w,z_1} A^c_{w,z_2}}{\begin{pmatrix} N \\ w \end{pmatrix}}. \tag{2.10}$$

El evaluar la función anterior requiere conocer las funciones de ponderación para los codificadores 1 y 2, respectivamente: A^c_{w,z_1} y A^c_{w,z_2}. La función para el codificador 1 no es dificil de calcular dado que se conoce el codificador RSC empleado, pero la función para el código 2 está muy ligada tanto a la estructura como al tamaño del entrelazador empleado. Para solucionar lo anterior, en [5] se propone un método para calcular dicha función en términos del tamaño del entrelazador partiendo de una generalización en la estructura del mismo, conocida con el nombre genérico de entrelazador uniforme.

Cada término en la sumatoria de la ecuación 2.10 depende de los i-ésimos coeficientes del espectro de distancias. En el trabajo de B. Vucetic y J. Yuan[5], se demuestra que para razones SNR bajas, la P_e es principalmente determinada por los coeficientes intermedios (términos 5 a 10 aproximadamente) del polinomio de ponderación de pesos, mientras que para razones SNR altas, la P_e está determinada por la d_{free}, de aquí que debido a la relativamente baja d_{free} de un turbo código [15], estos presentan un tope en su desempeño cuando se trabaja con razones SNR altas, efecto que se conoce como "zona de piso de ruido" *(error floor region)*. La figura 2.11 muestra una gráfica del desempeño P_b vs E_b/N_0.

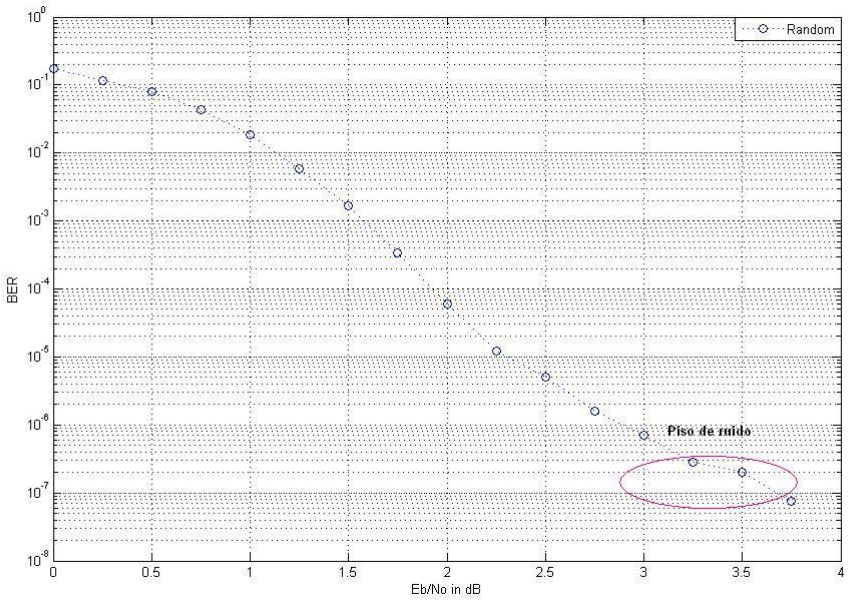
2.2.2. Proceso de aterrizado del turbo codificador

Aterrizar un codificador es sinónimo de llevar al enrejado trellis asociado al codificador a un estado conocido al término de la secuencia (una vez que la trama de información ha sido codificada), generalmente el estado nulo. Para el caso de los turbo códigos este proceso es necesario realizar al final de cada bloque, para que en el decodificador, como parte del algoritmo de decodificación, se pueda hacer un recorrido del enrejado trellis en ambos sentidos (hacia adelante y hacia atrás). El desempeño de un turbo código puede ser afectado en buena medida si no se realiza este proceso de aterrizado (tema que se analiza en detalle en el capítulo 3 de la presente tesis).

Llevar a cabo este proceso de aterrizado de un turbo código no es una tarea sencilla debido a los siguientes factores:

- Recursividad de los codificadores: No es posible aterrizar el codificador con solo agregar $(K-1)$ bits nulos, o de relleno; el valor de dichos bits de relleno dependen del estado actual del enrejado trellis una vez codificados los bits de la trama de información.

Fig. 2.11: Desempeño de un turbo codificador con tasa R=1/3, entrelazador pseudoaleatorio



- Presencia del entrelazador: Al tener los codificadores RSC conectados a través de un entrelazador, la tarea de aterrizar de manera simultánea los enrejados trellis asociados a cada codificador RSC se vuelve una tarea bastante compleja.

Este problema ha sido atacado desde muy diversas perspectivas, que van desde aterrizar únicamente el primero de los codificadores RSC [4] hasta el diseño de entrelazadores que permiten aterrizar de manera simultánea ambos codificadores [16]. La degradación en el desempeño que conlleva dejar sin aterrizar el segundo codificador se puede considerar despreciable si el tamaño N del entrelazador es lo suficientemente grande [17]. Sin embargo, esta última estrategia añade un nivel más alto de complejidad al diseño del entrelazador y reduce la gama de posibles entrelazadores que pueden ser utilizados [15].

2.2.3. Entrelazado

El entrelazador es en un sentido muy estricto, un permutador de tamaño N, que se puede describir como un proceso que cambia el orden de una determinada secuencia, en un proceso uno a uno; matemáticamente se puede describir como:

$$\Pi : i \rightarrow \pi(i)$$

Existen diversas estrategias para construir el entrelazador, cada cual con sus propias ventajas y desventajas, y por supuesto, cada una atacando el problema del entrelazado desde diferentes condiciones y perspectivas. Antes de comenzar a describir algunas de ellas, es conveniente mencionar las tareas que el entrelazado debe de cumplir [5]:

- Construcción de un código de bloque: La principal tarea del entrelazador es, a partir de códigos RSC de memoria pequeña, construir un código de bloque de tamaño N que genere por consecuencia, un código de mejor desempeño. En otras palabras, incrementa el peso de las secuencias codificadas.

- Romper la correlación: Otra tarea de gran importancia para el entrelazador es romper la correlación que existe entre las entradas de ambos codificadores. Lo anterior para que en el decodificador pueda darse un intercambio de información adecuada entre ambos decodificadores (principio turbo), para que en el caso de errores que no puedan ser corregidos por uno de los decodificadores, la probabilidad de que sean corregidos por el otro sea alta. Desde otro punto de vista, romper la correlación se puede interpretar como romper entradas de peso bajo; hay algunas entradas de peso bajo, particularmente las de peso de Hamming igual a 2, que ocasionan que el turbo codificador otorgue una salida de peso bajo afectando por consiguiente, toda su distancia libre. Si se logra que solo uno de los codificadores RSC genere una secuencia de peso bajo, la distancia libre del turbo codificador es globalmente mejorada.

No existe un entrelazador único que se pueda considerar "el mejor" entre todos los entrelazadores, es decir, que su uso proporcione el mejor desempeño en el turbo codificador construido. La elección del entrelazador depende de diversas variables como son por ejemplo: a) el tamaño de trama a ser empleada, b) la región de la razón SNR donde se desee trabajar el turbo codificador, c) minimizar o maximizar algún parámetro en particular.

Los diversos tipos de entrelazadores pueden ser agrupados en dos grandes categorías:

1. De naturaleza pseudoaleatoria: Su construcción se basa, por lo general, en generar un patrón pseudoaleatorio de N elementos con o sin la ayuda de diversas restricciones en la elección de dicho patrón. Cada vez que se ejecute el algoritmo de generación, es altamente probable que se genere un entrelazador diferente, aun cuando se seleccionen los mismos parámetros en el algoritmo.

2. De naturaleza determinística: A diferencia de los entrelazadores de naturaleza pseudoaleatoria, su construcción se basa en un algoritmo bien definido cuyo objetivo puede ser la maximización o la minimización de algún parámetro en particular. Cada vez que se ejecute el algoritmo de generación, invariablemente se generará el mismo entrelazador si se eligen los mismos parámetros.

En las siguientes secciones se mencionan algunos ejemplos de estos tipos de entrelazadores.

Entrelazadores de naturaleza pseudoaleatoria

Este tipo de entrelazadores son recomendados para tramas grandes ($N > 1000$). No se debe de perder de vista que una de las tareas del entrelazador es romper la correlación entre las entradas de los codificadores, con los tamaños de tramas antes citadas, se puede asegurar que esta tarea se cumple. A continuación se mencionan algunos entrelazadores pertenecientes a esta categoría.

Entrelazador pseudoaleatorio: Se trata de un arreglo de N elementos, donde cada elemento es elegido de manera aleatoria. Existen dos estrategias para su construcción:

- Generar de manera aleatoria y sin repetición números enteros entre 1 y N, el entrelazador queda determinado por el patrón generado.

- Generar números reales de manera aleatoria almacenando adicionalmente el orden en el que fueron generados, posteriormente, dicho patrón se ordena de mayor a menor o a la inversa. El entrelazador queda determinado por el arreglo correspondiente al orden de generación de los números aleatorios.

Entrelazador uniforme: Como se había mencionado, para poder calcular el desempeño de un turbo codificador se requiere de una generalización en el entrelazador conocida con el nombre de "entrelazador uniforme", el cual, dada una secuencia binaria de longitud k y peso de Hamming w, el entrelazador uniforme de longitud k corresponde con un concepto probabilístico que mapea la secuencia de longitud N y peso w, hacia cualquiera de las $\begin{pmatrix} N \\ w \end{pmatrix}$ permutaciones posibles de manera equiprobable $1/\begin{pmatrix} N \\ w \end{pmatrix}$, de aquí el nombre de uniforme.

Entrelazador S-random (Spread Random): Propuesto en 1995 [18], su construcción se basa en generar de manera aleatoria N números enteros sin repetición; cada entero generado es comparado con los S enteros previamente generados, si el entero en análisis se encuentra a una distancia d de alguno de los S enteros previos, $d < S$, donde $d = |\pi(i) - \pi(j)|$, con $|i - j| < S$, entonces se descarta y se genera un nuevo entero. Este proceso se repite hasta que se genere el arreglo de N enteros. El autor propone emplear $S = \sqrt{\frac{N}{2}}$ para que el algoritmo converja en un tiempo aceptable. Valores superiores de S raramente convergen a una solución.

Entrelazador High Spread Random: Este entrelazador, propuesto en 2000 por S. Crozier [19], basa su construcción en el algoritmo para el entrelazador S-random. Con las diferencias que redefine la métrica para la distancia S, y en vez de generar números enteros, trabaja con números reales. Su distancia queda entonces definida de la siguiente manera:

$$S(i, j) = |\pi(i) - \pi(j)| + |i - j| \tag{2.11}$$

Con estas modificaciones, se asegura un factor de dispersión de hasta $\left\lfloor \sqrt{2N} \right\rfloor$. Aunque es un entrelazador de tipo pseudoaleatorio, el autor reporta un buen desempeño para tramas pequeñas.

Entrelazadores de naturaleza determinística

Este tipo de entrelazadores son recomendados para tamaños de tramas pequeñas ($N < 1000$). Para estos tamaños, ya no es tan simple garantizar que la correlación entre las entradas de los codificadores RSC sea "rota" adecuadamente con una estructura pseudoaleatoria, por lo que en este rubro existen muy variadas perspectivas para la construcción de entrelazadores determinísticos. El concepto de romper la correlación se interpreta como mantener lo mas bajo posible el valor de correlación entre las secuencias en análisis.

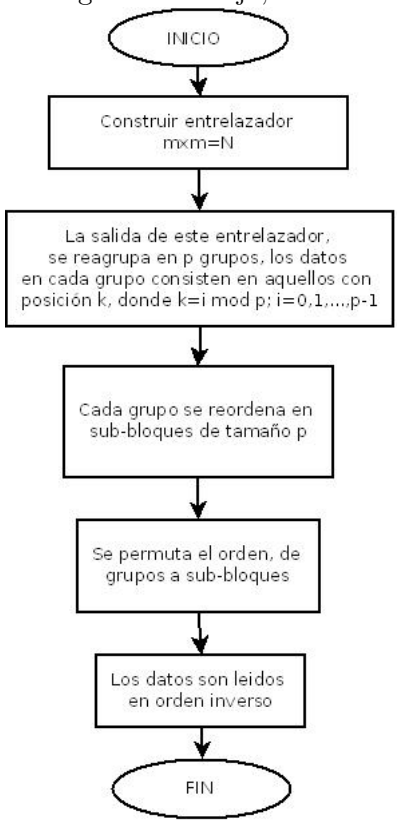
Entrelazador de bloque: Es quiza, el tipo de entrelazador más empleado en sistemas de comunicaciones rápida implantación. Se trata de un arreglo matricial en donde los elementos son acomodados por renglones y son leídos por columnas; razón por la cual es conocido también como entrelazador renglón-columna.

Entrelazador helicoidal: Se trata de una variante del entrelazador renglón-columna, en donde los datos son escritos por renglón pero son leídos en diagonal dentro del arreglo matricial construido.

Entrelazador Practical Size Interleaver: Llamado PSI en lo sucesivo, propuesto en 2000 por Wang & Kobayashi [20]. Los objetivos considerados en su diseño son: buscar romper la correlación entre las secuencias de entrada a los codificadores RSC, evitar en la medida de lo posible, las entradas de bajo peso que pudiesen generar salidas de bajo peso en el codificador, particularmente las entradas de peso 2. Lo sobresaliente de este entrelazador,

es que considera la memoria de los codificadores RSC para fijar un parámetro del algoritmo de generación, llamado periodo intrínseco p, $p \leq 2^{K-1} - 1$. El algoritmo de construcción se resume en la figura 2.12.

Fig. 2.12: Diagrama de flujo, entrelazador PSI.



Entrelazador Dithered Relative Prime: Llamado DRP en lo sucesivo, propuesto por Crozier en 2001 [21], basa su construcción en la elección de cuatro parámetros: **r**, **w**, s y p. El diagrama de flujo de su construcción se muestra en la figura 2.13 [21].

Entrelazador UMTS: este entrelazador, definido como parte del estándar UMTS para telefonía de tercera generación [22].

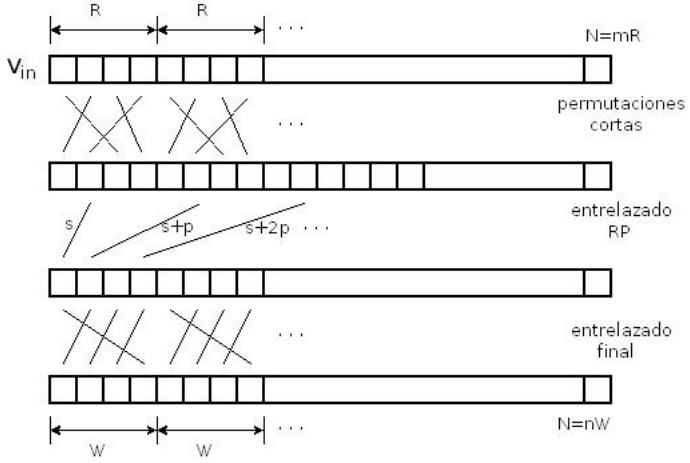
La elección de los parámetros para su construcción se puede hacer con base en los codificadores *RSC* empleados, tratando de minimizar las entradas de bajo peso que generen salidas de bajo peso. De este modo, este entrelazador presenta un buen desempeño para tramas pequeñas y relación SNR alta. Adicionalmente a su buen desempeño, requiere de algunos parámetros para su implementación, pocos parámetros comparados con el tamaño N del entrelazador, con lo cual se facilita bastante su implementación dentro de un sistema al requerir poca memoria. Bajo estas características, resulta muy recomendable su uso cuando se trabaja con tramas pequeñas.

2.3. Resumen

Los entrelazadores presentados en la sección anterior no son los únicos, las estrategias empleadas para la construcción de entrelazadores varían de acuerdo al autor y a la importancia que le asigne al buscar maximizar, o minimizar, algún parámetro en particular.

Para la construcción de entrelazadores se pueden seguir algunas estrategias, tales como:

Fig. 2.13: Esquema general de construcción entrelazador DRP.



- Maximizar el factor de dispersión S.
- Minimizar la correlación entre las secuencias de entrada a los codificadores RSC: Si esta tarea se logra, entonces el decodificador, que basa su operación en un algoritmo iterativo de intercambio de información (principio turbo), podrá realizar este intercambio de información de una manera más adecuada.
- Romper secuencias de entrada de bajo peso: Esto se hace buscando que solo uno de los dos codificadores RSC reciban esta secuencia. Como consecuencia, se maximiza la distancia libre del turbo código resultante y así se logra mejorar el desempeño del turbo código para razones SNR altas. Recuérdese que la distancia libre de un turbo código no es particularmente alta debido a dichas entradas de bajo peso [15].

El desempeño de un turbo codificador, al estar determinado por los coeficientes del polinomio de ponderación de pesos, es muy sensible entonces a la efectividad con la que el entrelazador empleado alcance sus objetivos. En particular, si se aumenta el tamaño N del entrelazador es menos probable que existan entradas de peso bajo que afecten simultáneamente a ambos codificadores RSC. La probabilidad de bit en error se ve afectada por un factor de $1/N$ y a este factor se le conoce como ganancia de entrelazado [5]. Para que esta ganancia de entrelazador pueda existir es necesario que los codificadores convolutivos empleados sean de naturaleza recursiva.

No existe un turbo codificador que presente un mejor desempeño (en términos de la BER) que cualquier otro, la elección de los elementos depende de diversos factores, como la elección adecuada de los codificadores, del entrelazador a ser empleado, del tamaño de trama, o de palabra, a ser codificada (tamaño del entrelazador) y de la región de la SNR en donde se desee trabajar [5].

3. TURBO DECODIFICACIÓN

Buena parte del alto desempeño en la ganancia de codificación alcanzado por los turbo códigos se debe sin duda a los algoritmos empleados para la decodificación. Al tener códigos concatenados en paralelo, la decodificación se realiza procesando en paralelo cada uno de los códigos e intercambiando información entre ellos en un proceso iterativo.

Existen diferentes algoritmos para la turbo decodificación, basados en modificaciones del algoritmo de Viterbi [23] o basados en algoritmos de decodificación de máxima probabilidad a posteriori (MAP, maximum a posteriori probability) [24]. En el presente capítulo se presentan estos algoritmos así como un comparativo entre ellos.

3.1. Introducción

3.1.1. Decodificación de máxima verosimilitud

Un receptor óptimo puede ser diseñado buscando minimizar alguno de los siguientes aspectos [5]:

- Tasa de palabras en error, WER (Word Error Rate): definida como la probabilidad de que $\hat{\mathbf{v}} \neq \mathbf{v}$ es decir, la probabilidad de que la palabra estimada y la palabra recibida sean diferentes.

- Tasa de símbolos en error, SER (Symbol Error Rate): definida como la probabilidad de que $\hat{\mathbf{x}_t} \neq \mathbf{x}_t$, $t = 1, 2, \ldots, N$ es decir, la probabilidad de que el símbolo estimado y el símbolo recibido en el instante t sean diferentes. Esta métrica va relacionada con el tipo de modulación empleada en el canal de comunicaciones.

- Tasa de bit en error, BER (Bit Error Rate): Definida como la probabilidad de que $\hat{\mathbf{c}_t} \neq \mathbf{c}_t$, es decir, la probabilidad de que el bit estimado y el bit recibido en el instante t sean diferentes.

Se tiene entonces tres posibles tipos de receptores, cada uno buscando minimizar alguno de las métricas anteriores: WER, SER o BER, respectivamente. Estos tres receptores tienen desempeños diferentes cuando se trabaja con razones señal a ruido (SNR) bajas, por el contrario su desempeño es muy similar para SNR altas. En el caso particular de un sistema binario, el SER y el BER son la misma medida.

La idea de la decodificación de secuencias con máxima verosimilitud (Maximum Likelihood Decoding, MLD) es un concepto en el cual el decodificador minimiza la probabilidad de error comparando todas las posibles secuencias recibidas y seleccionando aquella que menor probabilidad de error presente.

De una manera más formal, el decodificador compara todas las probabilidades condicionales $P(\mathbf{Z}|\mathbf{U}^{(m)})$, en donde $\mathbf{Z}$ es la secuencia recibida, $\mathbf{U}^{(m)}$ es una de las posibles secuencias transmitidas, y selecciona aquella secuencia transmitida que presente la máxima probabilidad de entre todas las posibles secuencias [6]. Al comparar todas las posibles

secuencias recibidas, la decodificación de máxima verosimilitud puede ser un proceso muy complicado si el número de posibles secuencias recibidas es muy alto.

En el caso concreto de un canal sin memoria con ruido AWGN, para un código convolutivo binario de tasa $R = 1/n$, dado que los eventos son independientes, la probabilidad $P(\mathbf{Z}|\mathbf{U}^{(m)})$ se expresa de la siguiente manera:

$$P(\mathbf{Z}|\mathbf{U}^{(m)}) = \prod_{i=1}^{\infty} P(Z_i|U_i^{(m)}) = \prod_{i=1}^{\infty}\prod_{j=1}^{n} P(z_{ij}|u_{ij}^{(m)}), \tag{3.1}$$

en donde: Z_i= i-ésimo símbolo de la secuencia recibida $\mathbf{Z}$,
$\mathbf{Z} = (z_1, z_2, \ldots, z_n, \ldots)$
$U_i^{(m)}$: i-ésimo símbolo de una palabra código $\mathbf{U}^{(m)}$ en análisis,
$\mathbf{U}^{(m)} = (U_1^{(m)}, U_2^{(m)}, \ldots, U_n^{(m)}, \ldots$
z_{ij}: j-ésimo bit de Z_i
$u_{ij}^{(m)}$: j-ésimo bit de U_i

Relacionando el problema de la decodificación con el enrejado trellis, el objetivo es seleccionar la trayectoria dentro del enrejado que menor métrica acumulada presente, de modo que la probabilidad de la ecuación 3.1 sea maximizada. Cada posible trayectoria dentro del enrejado representa una posible secuencia codificada. Por lo general y por facilidad de cómputo, la ecuación 3.1 se expresa en términos logarítmicos, quedando de la siguiente manera:

$$\gamma_u(m) = \log P(\mathbf{Z}|\mathbf{U}^{(m)}) = \sum_{i=1}^{\infty} \log P(Z_i|U_i^{(m)}) = \sum_{i=1}^{\infty}\sum_{j=1}^{n} \log P(z_{ij}|u_{ij}^{(m)}. \tag{3.2}$$

Para secuencias de tamaño L muy grandes, la complejidad de la tarea de decodificación es muy alta, el análisis y comparación de 2^L posibles trayectorias dentro del enrejado se vuelve una tarea muy ardua. Para simplificar lo anterior, se pueden descartar trayectorias dentro del enrejado que no se consideren candidatas a ser de máxima verosimilitud, la secuencia decodificada se elije de entre un número reducido de trayectorias candidatas, o sobrevivientes. En la siguiente sección se muestra como el algoritmo de Viterbi explota esta idea para seleccionar la trayectoria de máxima verosimilitud de entre un número acotado de trayectorias candidatas.

3.2. Algoritmo de Viterbi

El algoritmo de Viterbi, desarrollado en 1963 por A. J. Viterbi [23], fue presentado como una alternativa para la decodificación de códigos convolutivos y rápidamente fue aceptado por la comunidad científica y tecnológica debido a la facilidad que representa su implementación convirtiéndose en el algoritmo más empleado para la decodificación de códigos convolutivos. Actualmente, su uso no se limita a la decodificación sino que se aplica en diversas áreas tales como la detección y ecualización. Este algoritmo emplea la distancia de Hamming como métrica de comparación.

La idea principal del algoritmo de Viterbi es realizar una decodificación bajo el principio de máxima verosimilitud analizando todas las posibles secuencias, y eligiendo aquella que mayor probabilidad posea, o la que menor métrica acumulada contenga. Bajo esta

idea, el algoritmo de Viterbi reduce la complejidad en la búsqueda de la secuencia recibida que tenga la mayor verosimilitud, explotando la estructura asociada al enrejado. Este algoritmo requiere que se calcule una métrica, la cual mide qué tan cerca está la secuencia recibida respecto a todas las trayectorias que llegan a los nodos en el enrejado en el instante t_i, eliminando aquellas trayectorias que no son consideradas como de máxima verosimilitud. Si dos trayectorias llegan a un mismo nodo en el instante t_i, aquella con la menor métrica permanece y la otra se descarta, esta selección se realiza para cada nodo en el instante t_i. Cuando las métricas correspondientes a las trayectorias del enrejado que llegan a un mismo nodo son iguales, se decide arbitrariamente cuál de ellas permanece, dado que desde el punto de vista probabilístico ya no hay manera de decidir cuál de las trayectorias es mejor.

El algoritmo de Viterbi se resume en los siguientes pasos:

1. Se construye un enrejado similar al del codificador con la diferencia de que en cada instante de tiempo, cada rama del enrejado se etiqueta con la métrica entre el símbolo recibido y el símbolo (palabra del código) correspondiente a la misma rama del enrejado del codificador. Esta distancia se calcula al momento en que los símbolos son recibidos. Esta métrica representa la diferencia entre lo que debería de llegar, y lo que se recibe en realidad por efecto del ruido.

2. En el instante $(t + 1)$, para un código binario de tasa $1/n$ en cada nodo (estado) del enrejado se tienen dos posibles trayectorias de entrada, aquella que presente la menor métrica se elige como la sobreviviente y la otra es descartada. Este cálculo se realiza para cada estado en un determinado instante de tiempo, de modo que cada nodo cuenta con una trayectoria sobreviviente. Tanto la trayectoria sobreviviente como su métrica son almacenadas. Se tendrá entonces un máximo de 2^{K-1} trayectorias con sus respectivas métricas almacenadas, que corresponde con el número de estados del enrejado; la trayectoria de máxima verosimilitud está contenida en dichas trayectorias almacenadas. La métrica de las trayectorias de entrada a cada nodo se calcula como la métrica de cada trayectoria en el instante previo t, mas la métrica de la rama de entrada respectiva, es decir, es la métrica de la trayectoria en el instante $(t - 1)$ mas la métrica de la transición del estado $(t - 1)$ al estado t. De este modo, la métrica de cada trayectoria sobreviviente es el acumulado de las métricas de cada rama que componen dicha trayectoria.

3. El algoritmo se continua aplicando hasta llegar al final de la trama a decodificar y por consiguiente, hasta llegar al nodo final (generalmente se asume que el estado final del codificador es el estado nulo). En este instante se toma una decisión sobre la trayectoria más verosimil en base a la métrica de la misma.

Para su implementación, la cantidad de memoria necesaria para almacenar todas las trayectorias y sus métricas está en función de la longitud de la secuencia recibida, razón por la cual el algoritmo toma ventaja del hecho de que, después de ciertas unidades de tiempo, todas las trayectorias sobrevivientes en cada nodo tendrán una raíz en común, con lo que dicha raíz común se convierte en la salida del decodificador y solo se almacena una cantidad fija de información de las trayectorias y sus métricas. En la práctica se define una ventana de decodificación, la cual se elige en el orden de 5 veces la longitud de restricción del código K.

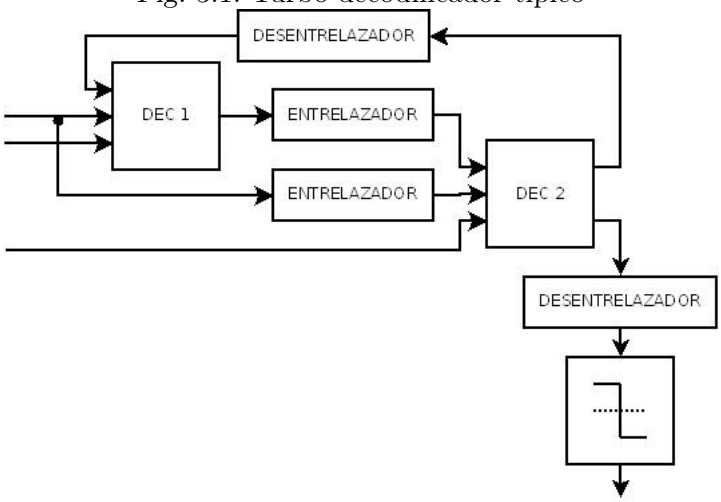
Otra propiedad del algoritmo de Viterbi es que está enfocado a minimizar la probabilidad de error de la trama en conjunto y no la probabilidad de bit (aislado) en error. Así

como el hecho que, este algoritmo recibe una entrada "dura" por parte del demodulador y entrega una decisión también "dura", entendiendo el término "duro" como un resultado binario, $0, 1$.

3.3. Algoritmos para la turbo decodificación

Un turbo decodificador típico, como se muestra en la figura 3.1, consiste en dos módulos decodificadores con entrada y salida "suave" (SISO, soft input soft output) los cuales trabajan en un proceso iterativo, intercambiando información entre ellos. El entrelazador empleado para interconectar al primer decodificador con el segundo es el mismo que el empleado por el codificador.

Fig. 3.1: Turbo decodificador típico



Asumiendo que la secuencia binaria de información $\mathbf{c}$ contiene elementos que son variables aleatorias independientes con una probabilidad a priori de

$$P\{c_i = 1\} = P\{c_i = 0\} = 0{,}5,$$

donde c_i representa al i-ésimo elemento de la secuencia $\mathbf{c}$.

La salida $\mathbf{v}$ del codificador se compone de la multiplexión o concatenación en serie de la salida sistemática v_0 y de la salida de paridad v_1 del primer codificador RSC, así como la salida de paridad v_2 del segundo codificador RSC, quedando globalmente de la siguiente forma:

$$\mathbf{v} = \{(v_{0,1}, v_{1,1}, v_{2,1}), (v_{0,2}, v_{1,2}, v_{2,2}), (v_{0,3}, v_{1,3}, v_{2,3}), \ldots\}.$$

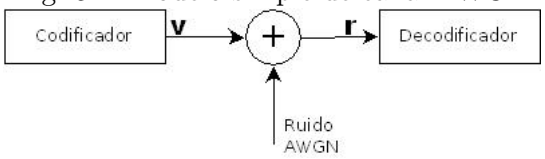
El decodificador recibe la secuencia $\mathbf{r}$, correspondiente a la salida del codificador mas una componente de ruido AWGN, debida al canal de transmisión, como se muestra en la figura 3.2.

La entrada $\mathbf{r}'$ al primer decodificador está compuesta por:

$$\mathbf{r}' = \{(r_{0,1}, r_{1,1}), (r_{0,2}, r_{1,2}), (r_{0,3}, r_{1,3}), \ldots\}.$$

El entrelazador recibe la secuencia $\mathbf{r}_0$, correspondiente a la salida sistemática $\mathbf{v}_0$ del codificador, mas una componente de ruido, y genera la secuencia $\tilde{\mathbf{r}}_0$. La entrada al segundo decodificador, $\mathbf{r}''$, está dada por:

Fig. 3.2: Modelo simple de canal AWGN



$$\mathbf{r}'' = \{(\tilde{r}_{0,1}, r_{2,1}), (\tilde{r}_{0,2}, r_{2,2}), \ldots\}.$$

Dados los vectores $\mathbf{r}'$ y $\mathbf{r}''$, la tarea del decodificador es tomar una decisión basándose en los vectores mencionados y ciertas condiciones de ruido del canal. De este modo, un algoritmo óptimo orientado a bits toma la decisión sobre el bit c_t basándose en la siguiente relación logarítmica [5]:

$$\Lambda(c_t) = log\frac{P_r(c_t = 1|\mathbf{r}', \mathbf{r}'')}{P_r(c_t = 0|\mathbf{r}', \mathbf{r}'')} = log\frac{\sum_{\mathbf{c}:c_t=1} P_r(c_t = 1|\mathbf{r}', \mathbf{r}'')}{\sum_{\mathbf{c}:c_t=0} P_r(c_t = 0|\mathbf{r}', \mathbf{r}'')}. \tag{3.3}$$

La regla de decisión es:

$$c_t = \begin{cases} 1, & si \qquad \Lambda(c_t) \geq 0 \\ 0, & \text{en caso contrario.} \end{cases}$$

La ecuación 3.3 se basa en estimar con la máxima verosimilitud el valor del bit c_t basándose en los valores recibidos $\mathbf{r}'$ y $\mathbf{r}''$. El valor $\Lambda(c_t)$ se conoce como relación logarítmica de máxima verosimilitud.

Las variables aleatorias $\mathbf{r}'$ y $\mathbf{r}''$ deben estar con la menor correlación posible, es en este punto donde el entrelazador asume esta tarea; el entrelazador debe de ser elegido de modo tal de asegurar que las secuencias a ser codificadas X_k y X'_k estén lo menor correlacionadas posibles (lo ideal es que la correlación sea nula) para de este modo poder garantizar que $\mathbf{r}'$ y $\mathbf{r}''$ sean independientes y entonces poder asumir que:

$$P_r(\mathbf{r}', \mathbf{r}''|\mathbf{c}) = P_r(\mathbf{r}'|\mathbf{c})P_r(\mathbf{r}''|\mathbf{c}).$$

Así, la ecuación 3.3 queda:

$$\Lambda(c_t) = log\frac{\sum_{\mathbf{c}:c_t=1} P_r(\mathbf{r}'|\mathbf{c})P_r(\mathbf{r}''|\mathbf{c})P_r(\mathbf{c})}{\sum_{\mathbf{c}:c_t=0} P_r(\mathbf{r}'|\mathbf{c})P_r(\mathbf{r}''|\mathbf{c})P_r(\mathbf{c})}. \tag{3.4}$$

La manera de calcular la expresión 3.3 es lo que hace diferentes los algoritmos definidos para la decodificación turbo.

3.3.1. Algoritmo SOVA

El algoritmo SOVA (soft output Viterbi algorithm), propuesto en 1989 por J. Hagenauer y P. Hoeher [25] se basa en el algoritmo de Viterbi, el cual se modifica para convertirlo en un algoritmo de entrada y salida "suaves", es decir, a la entrada recibe información "suave" en forma de probabilidades a priori, y en la salida produce una salida también "suave" en forma de probabilidades a posteriori. De esta manera se puede utilizar en el algoritmo iterativo de turbo decodificación.

Este algoritmo hace un doble recorrido del enrejado, el primero en dirección hacia adelante (en el mismo sentido del flujo de datos), conocido como recursión hacia adelante (forward recursion), lo cual es equivalente a ejecutar el algoritmo de manera tradicional. El segundo recorrido se hace en sentido inverso (en sentido contrario al flujo de datos), o hacia atrás (backward recursion). En este segundo recorrido se calculan las probabilidades de transición de estados. La utilización conjunta de las métricas obtenidas en ambos recorridos permite obtener la salida "suave". Esta salida "suave" es retroalimentada al otro decodificador en forma de información a priori en un esquema iterativo, es decir, la salida a posteriori de un decodificador es la entrada a priori del otro decodificador. A continuación se describe de manera similar a como se realiza en [5].

Este algoritmo hace un estimado de la decisión suave de cada símbolo binario c_t transmitido con base en la secuencia recibida $\mathbf{r}$, tal como se describe en la ecuación 3.4. El valor absoluto de $\Lambda(c_t)$ se toma como la salida "suave" y su signo como la decisión "dura" realizada, es decir, si $\Lambda(c_t)$ es positivo se decide como un "uno" binario, o un "cero" en caso contrario. Para calcular el valor de $\Lambda(c_t)$ se considera el esquema iterativo característico de la turbo decodificación, se considera un canal AWGN y se considera además que las probabilidades a priori $p_t(c_t)$ para cada bit no son constantes, sino que varían por el intercambio de información entre los decodificadores.

Para el cálculo de la métrica de una trayectoria dentro del enrejado trellis, el algoritmo maximiza el logaritmo de la probabilidad a posteriori $P_r\{\mathbf{c}, \mathbf{r}\}$:

$$\log P_r\{\mathbf{c}, \mathbf{r}\} = \log P_r\{\mathbf{c}\} + \log P_r\{\mathbf{r}|\mathbf{c}\}. \tag{3.5}$$

La métrica de una rama en el enrejado trellis en un instante t queda dada por la siguiente expresión:

$$v_t^{c_t} = \sum_{i=0}^{n-1} (r_{t,i} - x_{t,i})^2 - \log p_t(c_t) \tag{3.6}$$

La métrica de una trayectoria $\mathbf{x}$ a través del enrejado trellis, se realiza mediante la siguiente expresión:

$$\mu_t^x = \sum_{t'=1}^{t} v_{t'}^{c_{t'}} = \mu_{t-1}^x + v_t^{c_t} \tag{3.7}$$

El decodificador SOVA selecciona la trayectoria $\widehat{X}$ con la menor métrica $\mu_{\tau,min}$, y la cataloga como la trayectoria de mayor verosimilitud, (*ML*, Maximum Likelihood). Adicional a la trayectoria *ML*, el algoritmo considera además otra trayectoria conocida trayectoria del competidor, la cual se define como la trayectoria con la menor métrica de entre todas las trayectorias que llegan al restante de símbolos de la trayectoria *ML* en el instante t. Por ejemplo, si en el instante t, se tiene que el símbolo *ML* corresponde a un bit "1" como entrada, entonces la trayectoria del competidor será aquella que presente la menor métrica para el símbolo complementario de entrada "0" y viceversa. A esta trayectoria se le denota como $\mu_{t,c}$.

Asumiendo que en el instante t, la trayectoria *ML* corresponde a un bit "1" recibido, las probabilidades de que el bit recibido en el instante t sea "1" o "0", son proporcionales al peso de la trayectoria *ML* y de la trayectoria competidora, respectivamente.

$$P_r(c_t = 1|\mathbf{r}_1^\tau) \backsim e^{-\mu_{\tau,min}},\ P_r(c_t = 0|\mathbf{r}_1^\tau) \backsim e^{-\mu_{t,c}}. \tag{3.8}$$

El algoritmo calcula el valor de $\Lambda(c_t)$ y considerando la simplificación de la ecuación 3.8, la ecuación 3.3 queda:

$$\Lambda(c_t) = log\frac{P_r(c_t = 1|\mathbf{r}_1^\tau)}{P_r(c_t = 0|\mathbf{r}_1^\tau)} \backsim log\frac{e^{-\mu_{\tau,min}}}{e^{-\mu_{t,c}}} = \mu_{\tau,min} - \mu_{t,c}, \tag{3.9}$$

finalmente, el cálculo de $\Lambda(c_t)$ queda:

$$\Lambda(c_t) = \mu_t^0 - \mu_t^1 = (-1)^{c_t}(\mu_{\tau,min} - \mu_{t,c}), \tag{3.10}$$

donde:
μ_t^0 y μ_t^0 son las métricas de las trayectorias de menor peso para el bit recibido "0" y "1" respectivamente, en el instante t.

En resumen, la salida "suave" se puede calcular como la diferencia entre las métricas de la trayectoria de menor peso que tenga el valor "0" como bit recibido, y la trayectoria de menor peso que tenga el valor "1" como bit recibido. El signo de esta diferencia determina la decisión "dura" del bit recibido en el instante t, y el valor absoluto corresponde a la medida de credibilidad de la decisión tomada, es decir, la información "suave" que es retroalimentada al otro decodificador, el cual la tomará como información a priori.

El algoritmo SOVA se puede resumir de la siguiente manera:

- Recursión hacia adelante: En la recursión hacia adelante, se opera de manera similar al algoritmo de Viterbi normal, se analiza las transiciones de estado dentro del enrejado, se comparan las métricas de las trayectorias y se determina la mejor trayectoria para cada nodo. Esto se repite hasta llegar al final del enrejado, es decir, el tiempo se incrementa desde $t = 0$ hasta el tiempo correspondiente al final del enrejado. En este punto se elige la trayectoria *ML*.

- Recursión hacia atrás: Se arranca el recorrido del enrejado desde el estado final, de aqui la importancia de que tanto el estado final, como el inicial sean perfectamente conocidos. Se calculan las métricas de cada rama que llega a un nodo en el instante t (transiciones de estado), calculando las métricas de las trayectorias entrantes a cada nodo y se almacena la trayectoria sobreviviente en cada nodo. Este proceso se repite hasta llegar al inicio del enrejado, es decir, el índice del tiempo se decrementa desde el final del enrejado, hasta llegar a $t = 0$.

- Cálculo de la información "suave": Como se mencionó anteriormente, para obtener la información "suave" se requiere la métrica de la trayectoria *ML* (ya obtenida con la recursión hacia adelante) y la métrica de la trayectoria competidora, o como se mencionó anteriormente en la ecuación 3.10, se requiere calcular las métricas de la trayectoria de menor peso que tengan como entrada el valor de bit "0" y "1" respectivamente, y calcular su diferencia. Esto se hace analizando a partir del instante $t = 0$, continuando en cada instante t, y de esta manera se identifica la trayectoria *ML*.

  $$\mu_t^i = \mu_{\tau,min}.$$

  Para hallar la métrica de la trayectoria μ_t^c, donde $c = i \oplus 1$ (la notación $\oplus$ corresponde a la operación suma módulo 2), se emplea la siguiente ecuación [5]:

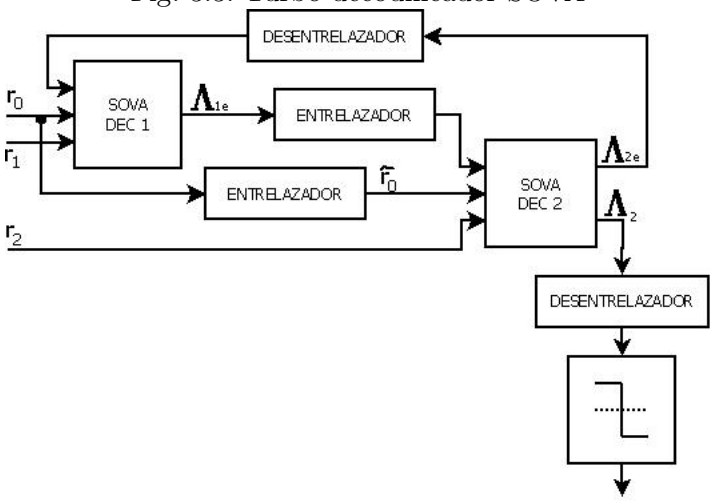
  $$\mu_{t,c} = min\left\{\mu_{t-1}^f(l') + v_t^c(l', l) + \mu_t^b(l)\right\}, \tag{3.11}$$

donde:
l, l' son todos los posibles estados dentro del enrejado trellis.
$\mu_{t-1}^{f}(l')$ es la métrica de la trayectoria sobreviviente en el instante $(t-1)$ y nodo l', recorrida hacia adelante (*foward*). También se puede interpretar como la trayectoria ML en el instante $t-1$.
$v_t^c(l', l)$ es la métrica de la rama que va del nodo l' al nodo l en el instante t, para el bit $c = i \oplus 1$ (complemento del bit ML).
$\mu_t^b(l)$ es la métrica de la trayectoria sobreviviente en el instante t y nodo l, en la dirección hacia atrás (*backward*).

La ecuación 3.11 indica que la métrica μ_t^c depende la métrica de la trayectoria sobreviviente en el instante $t-1$ (foward) y se considera que para la transición de los instantes $t-1$ a t (nodos l' a l respectivamente) se tiene a la entrada el bit complemento del ML estimado. Se toma dicha métrica de transición y se completa con la métrica de la trayectoria sobreviviente al nodo l (*backward*). Para esta ecuación, se analizan las trayectorias *backward*, *forward* y las transiciones de estado que impliquen un bit de entrada $c = i \oplus 1$ y se toma como trayectoria competidora aquella que presente la menor métrica, en ese sentido se debe interpretar el mínimo.

Para el caso de un turbo decodificador típico, se realiza un intercambio de información "suave" entre los decodificadores, esto refina el proceso de decodificación haciendo más verosimil la información que cada decodificador toma. Este intercambio de información entre ambos decodificadores se puede observar en la figura 3.3 [5].

Fig. 3.3: Turbo decodificador SOVA



En esta figura se observa el efecto "turbo", dado que la salida del primer decodificador Λ_{1e} se entrelaza y se retroalimenta hacia el segundo decodificador, el cual la toma como su información a priori. A su vez la salida del segundo decodificador, Λ_{2e}, es desentrelazada y enviada hacia el primer decodificador, el cual la toma respectivamente como su información a priori para empezar así la siguiente iteración. Estas informaciones a priori afectan el cálculo de la métrica de cada rama, como se muestra en la ecuación 3.6. Cada nueva iteración afecta el valor de las probabilidades a priori debido al intercambio de información entre los decodificadores.

Este proceso iterativo se repite hasta alcanzar un número predeterminado de iteraciones, o hasta que se alcance un cierto criterio de calidad en la información decodificada,

entendiéndose calidad como un número máximo de errores permitido.

En el instante t, el estimado *ML* para el símbolo c_t se puede obtener a partir de la métrica de trayectoria *ML*, $\mu_{\tau,min}$, de la siguiente manera:

$$\begin{aligned} \mu_{t,c_t} &= \mu_{\tau,min} \\ &= \sum_{t'=0}^{t-1} v_{t'} + v_t^{c_t} + \sum_{t'=t+1}^{\tau} v_{t'} \qquad (3.12) \\ &= \mu'_t + v_t^{c_t} \qquad (3.13) \end{aligned}$$

en donde:

$$\mu'_{t,1} = \sum_{t'=0}^{t-1} v_{t'} + \sum_{t'=t+1}^{\tau} v_{t'}. \qquad (3.14)$$

Con base en la ecuación anterior, se puede escribir la salida "suave" del primer decodificador como:

$$\Lambda_1(c_t) = (-1)^{c_t}\{(\mu'_{t,1} + v_{t,1}^{c_t}) - (\mu''_{t,1} + v_{t,1}^{c})\}. \qquad (3.15)$$

En donde:

- $\mu'_{t,1}$ se calcula de acuerdo a la ecuación 3.14, para la trayectoria *ML*.
- $v_{t,1}^{c_t}$ es la métrica de la rama, del instante $t-1$ al instante t, para el símbolo *ML*.
- $\mu''_{t,1}$ se calcula de acuerdo a la ecuación 3.14, para la trayectoria del mejor competidor al símbolo *ML*.
- y, finalmente, $v_{t,1}^{c}$ es el peso de la rama, del instante $t-1$ al instante t, para el símbolo complemento al símbolo *ML*.

Reorganizando la ecuación 3.15 y normalizando los valores de voltaje en la modulación empleada, de modo que a un bit "1" se le asigna un nivel de 1 Volt, y a un bit "0" se le asigna un valor de -1 Volt, quedando de la siguiente manera [5]:

$$\Lambda_1(c_t) = \log \frac{p_t^1(1)}{b} + 4r_{t,0} + \Lambda_{1e}(c_t). \qquad (3.16)$$

En donde:

- $p_t^1(1)$ es la probabilidad a priori para un "1".
- $p_t^1(0)$ es la probabilidad a priori para un "0".
- $\Lambda_{1e}(c_t)$ es la información extrínseca del decodificador 1, que se calcula con la siguiente ecuación:

$$\Lambda_{1e}(c_t) = (-1)^{c_t}\{(\mu'_{t,1} - \mu''_{t,1} - 2\sum_{i=1}^{n-1}(x_{t,i}^{c_t} - x_{t,i}^{c})\}. \qquad (3.17)$$

La información extrínseca del primer decodificador es pasada a través del entrelazador, y recibida por el segundo decodificador, el cual la toma como un estimado de la información a priori.

$$\tilde{\Lambda}_{1e}(c_t) = \log \frac{p_t^2(1)}{p_t^2(0)}, \tag{3.18}$$

en donde $p_t^2(1)$ y $p_t^2(0)$ son las informaciones a priori para "1" y "0" respectivamente, a la entrada del segundo decodificador, en el instante t.

El segundo decodificador calcula su información "suave" $\Lambda_2(c_t)$ de manera similar a la ecuación 3.15. Para el cálculo de $\Lambda_2(c_t)$ se sustituye la información extrínseca proveniente del primer decodificador, quedando de la siguiente manera:

$$\Lambda_2(c_t) = \tilde{\Lambda}_{1e}(c_t) + 4\tilde{r}_{t,0} + \Lambda_{2e}(c_t). \tag{3.19}$$

Finalmente, para la siguiente iteración, en el primer decodificador se sustituye la información extrínseca proveniente del segundo decodificador.

$$\Lambda_1(c_t) = \tilde{\Lambda}_{1e}(c_t) + 4r_{t,0} + \Lambda_{1e}(c_t). \tag{3.20}$$

De este modo, al aumentar el número de iteraciones en el proceso de decodificación, la verosimilitud de la información que entreguen los decodificadores será más alta, hasta que se decida tomar una decisión final, o decisión "dura" del bit decodificado. Lo recomendable es hacer dicha decisión "dura" con base en $\Lambda_2(c_t)$.

En resumen, el algoritmo SOVA presenta las siguientes características:

- Minimiza la probabilidad P_e de la secuencia en conjunto.
- Requiere que los estados inicial y final dentro del enrejado sean conocidos.
- No requiere conocer la varianza del ruido en el canal de comunicaciones.
- La métrica de la trayectoria sobreviviente se toma de la recursión hacia adelante, de manera similar al algoritmo de Viterbi tradicional. Esta trayectoria se le denomina trayectoria *ML*.
- La métrica del "mejor competidor" se obtiene de una combinación de las métricas de las recursiones hacia adelante, hacia atrás y de la transición de estados.
- La información "suave" se interpreta como la credibilidad de que el símbolo $c_t = X$ sea el peso del competidor, menos el peso del *ML*; si el *ML* tiene poco peso y el competidor un peso alto, la decisión sobre el *ML* será correcta.
- Es necesario que el enrejado trellis sea terminado en el estado nulo, para que de esta manera, el recorrido hacia atrás pueda inicializarse correctamente. A este proceso se le conoce como aterrizado del enrejado trellis.

Para reducir la complejidad de este algoritmo, es posible calcular en paralelo las recursiones hacia adelante y hacia atrás y posteriormente, calcular la información "suave".

3.3.2. Algoritmo MAP

El algoritmo MAP (*Maximum-a posteriori*), propuesto en 1974 por Bahl, Cocke, Jelinek y Raviv [24], también conocido como BCJR por las iniciales de sus autores, es un algoritmo empleado para la estimación de parámetros pseudoaleatorios cuando se conoce sus probabilidades a priori. En particular, en teoría de códigos se emplea para determinar el bit, o símbolo, con mayor verosimilitud, dada una secuencia recibida en presencia de ruido. El algoritmo requiere que el codificador pueda modelarse como una cadena de Markov, en donde la salida del mismo ($\mathbf{v}_t$, secuencia codificada) y la transición de estado ($S_t \rightarrow S_{t+1}$) dependen únicamente de la secuencia de entrada. De este modo, se puede analizar la salida del codificador en el instante **t** como una probabilidad:

$$q_t(\mathbf{x}_t|l', l) = P_r\left\{\mathbf{v}_t|S_{t-1} = l', S_t = l\right\}. \tag{3.21}$$

Donde **x** representa la señal modulada. Considerando que se trata de un canal AWGN, las probabilidades de transición se definen como:

$$P_r\left\{\mathbf{r}_1^\tau|\mathbf{x}_1^\tau\right\} = \prod_{j=1}^{\tau} R(\mathbf{r}_j|\mathbf{x}_j) = \prod_{j=1}^{\tau}\prod_{i=0}^{n-1} P_r(r_{j,i}|x_{j,i}), \tag{3.22}$$

mientras que:

$$P_r\left\{r_{j,i}|x_{j,i} = -1\right\} = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(r_{j,i}+1)^2}{2\sigma^2}} \tag{3.23}$$

$$P_r\left\{r_{j,i}|x_{j,i} = +1\right\} = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(r_{j,i}-1)^2}{2\sigma^2}}, \tag{3.24}$$

en donde σ^2 es la varianza del ruido del canal AWGN.

Para calcular la relación logarítmica de verosimilitud de la ecuación 3.3, dada la secuencia recibida $\mathbf{r}_1^\tau$, el algoritmo MAP calcula las probabilidades a posteriori de la siguiente manera:

$$P_r\left\{c_t = 0|\mathbf{r}_1^\tau\right\} = \sum_{(l',l)\in B_t^0} Pr\left\{S_{t-1} = l', S_t = l|\mathbf{r}_1^\tau\right\} \tag{3.25}$$

$$= \sum_{(l',l)\in B_t^0} \frac{Pr\left\{S_{t-1} = l', S_t = l|\mathbf{r}_1^\tau\right\}}{P_r\left\{\mathbf{r}_1^\tau\right\}}, \tag{3.26}$$

y de manera similar

$$P_r\left\{c_t = 1|\mathbf{r}_1^\tau\right\} = \sum_{(l',l)\in B_t^1} Pr\left\{S_{t-1} = l', S_t = l|\mathbf{r}_1^\tau\right\} \tag{3.27}$$

$$= \sum_{(l',l)\in B_t^1} \frac{Pr\left\{S_{t-1} = l', S_t = l|\mathbf{r}_1^\tau\right\}}{P_r\left\{\mathbf{r}_1^\tau\right\}}. \tag{3.28}$$

Donde B_t^0 y B_t^1 son la serie de transiciones $S_{t-1} = l' \rightarrow S_t = l$ generadas por un bit de entrada $c_t = 0$, o $c_t = 1$, respectivamente.

Para calcular la probabilidad condicional $Pr\{S_{t-1} = l', S_t = l|\mathbf{r}_1^\tau\}$, se emplea la definición de probabilidad condicional, para tener entonces:

$$Pr\{S_{t-1} = l', S_t = l|\mathbf{r}_1^\tau\} = \frac{Pr\{S_{t-1} = l', S_t = l, \mathbf{r}_1^\tau\}}{P\{\mathbf{r}_1^\tau\}}. \tag{3.29}$$

La ventaja de modelar el proceso de codificación como un proceso de Markov, permite reescribir el numerador de la ecuación anterior como:

$$Pr\{S_{t-1} = l', S_t = l, \mathbf{r}_1^\tau\} = \alpha_{t-1}(S_{t-l} = l')\gamma_t(S_{t-1} = l', S_t = l)\beta_t(S_t = l). \tag{3.30}$$

Sustituyendo las ecuaciones 3.26 y 3.28 en la ecuación 3.3 tenemos entonces:

$$\begin{aligned}
\Lambda(c_t) &= log\frac{P_r(c_t = 1|\mathbf{r}_1^\tau,)}{P_r(c_t = 0|\mathbf{r}_1^\tau)} \\
&= log\frac{\sum\limits_{(l',l)\in B_t^1} \frac{Pr\{S_{t-1}=l', S_t=l|\mathbf{r}_1^\tau\}}{P_r\{\mathbf{r}_1^\tau\}}}{\sum\limits_{(l',l)\in B_t^0} \frac{Pr\{S_{t-1}=l', S_t=l|\mathbf{r}_1^\tau\}}{P_r\{\mathbf{r}_1^\tau\}}} \\
&= log\frac{\sum\limits_{(l',l)\in B_t^1} Pr\{S_{t-1} = l', S_t = l|\mathbf{r}_1^\tau\}}{\sum\limits_{(l',l)\in B_t^0} Pr\{S_{t-1} = l', S_t = l|\mathbf{r}_1^\tau\}} \\
&= log\frac{\sum\limits_{(l',l)\in B_t^1} \alpha_{t-1}(S_{t-l} = l')\gamma_t^1(S_{t-1} = l', S_t = l)\beta_t(S_t = l)}{\sum\limits_{(l',l)\in B_t^0} \alpha_{t-1}(S_{t-l} = l')\gamma_t^0(S_{t-1} = l', S_t = l)\beta_t(S_t = l)}.
\end{aligned} \tag{3.31}$$

En donde:

- $\alpha(S_{t-1} = l')$ representa la probabilidad de que el estado actual sea $S_{t-1} = l'$ dado el vector $\mathbf{r}$, el estado actual dentro del enrejado depende exclusivamente de los valores anteriores del vector $\mathbf{r}$, no de valores futuros de $\mathbf{r}$; esta métrica se obtiene al recorrer el enrejado hacia adelante.

- $\gamma(S_{t-1} = l' \to S_t = l)$ representa la probabilidad de transición del estado $S_{t-1} = l'$ al estado $S_t = l$, dado que en el instante t se está en el estado $S_{t-1} = l'$. A este valor se le relaciona con la métrica de la rama asociada a la transición de estados antes mencionada.

- Finalmente, el valor $\beta(S_t) = l$, o métrica de recursión hacia atrás, representa la probabilidad de que el estado final sea $S_t = l$, sin considerar que el estado inicial sea $S_{t-1} = l'$. Se le relaciona con la métrica de la recursión hacia atrás del enrejado.

El valor $\alpha(S_t = l)$ se obtiene de manera iterativa, empleando la siguiente expresión:

$$\alpha_t(S_t = l) = \sum_{l''} \alpha_{t-1}(l'') \sum_{i\in(0,1)} \gamma_t^i(l', l). \tag{3.32}$$

Se debe considerar que para $t = 0$, se tiene como condición inicial que $\alpha_0(0) = 1$ y $\alpha_0(l) = 0$, con $l \neq 0$.

En donde l'' es uno de los posibles estados dentro del enrejado en el instante $t-1$. El valor $\beta_t(l)$ se calcula de la siguiente manera:

$$\beta_t(l) = \sum_{l''} \beta_{t+1}(l'') \sum_{i\in(0,1)} \gamma^i_{t+1}(l, l''). \tag{3.33}$$

Esto para valores de $t = \tau-1, \cdots, 1, 0$. Las condiciones iniciales son: $\beta_\tau(0) = 1$ y $\beta_\tau(l) = 0$, para $l \neq 0$.
Finalmente, el valor $\gamma(S_{t-1} = l' \to S_t = l)$ se calcula con la siguiente expresión:

$$\gamma^i_t(S_{t-1} = l', S_t = l) = \begin{cases} p_t(i) = \exp(-\frac{\sum_{j=0}^{n-1}(r^i_{t,j} - x^i_{t,j}(l))^2}{2\sigma^2}) & \text{para } (l', l) \in B^i_t \\ 0 & \text{en cualquier otro caso.} \end{cases} \tag{3.34}$$

En donde $p_t(i)$ representa la probabilidad a priori de que $c_t = i$ y el valor $x^i_{t,j}(l)$ representa la salida del codificador asociada con la transición de $S_{t-1} = l'$ a $S_t = l$, dada la entrada $c_t = i$.

Cuando se trabaja en un esquema iterativo, se realiza un intercambio de información entre ambos decodificadores, la salida de uno, información a posteriori es tomada por el otro como información a priori, para aumentar la credibilidad de la decodificación realizada, de una manera muy similar a como se muestra en la figura 3.3.

Considerando las ecuaciones 3.32, 3.33 y 3.34, la ecuación 3.31 queda reescrita como[5]:

$$\Lambda_1(c_t) = \log \frac{\sum_{l',l=0}^{M_s-1} \alpha_{t-1}(l') p_t(1) \exp(-\frac{\sum_{j=0}^{n-1}(r_{t,j} - x^1_{t,j}(l))^2}{2\sigma^2})}{\sum_{l',l=0}^{M_s-1} \alpha_{t-1}(l') p_t(0) \exp(-\frac{\sum_{j=0}^{n-1}(r_{t,j} - x^0_{t,j}(l))^2}{2\sigma^2})}. \tag{3.35}$$

Donde el parámetro $(n-1)$ representa la cantidad de bits de paridad.

Esta ecuación se puede simplificar al separar de la ecuación anterior los términos $p_t(0)$ y $p_t(1)$ buscando expresar la información a priori por separado y considerando que se trabaja con códigos sistemáticos, los términos $x^i_{t,0}$, $i = 0, 1$ son, respectivamente $x^1_{t,0} = 1$ y $x^0_{t,0} = -1$, ya que se trata de los bits sistemáticos. De este modo, la ecuación 3.35 queda [5]:

$$\Lambda_1(c_t) = \log \frac{p_t(1)}{p_t(0)} + \frac{2}{\sigma^2} r_{t,0} + \Lambda_{1e}(c_t). \tag{3.36}$$

En donde $\Lambda_{1e}(c_t)$ se calcula de la siguiente manera:

$$\Lambda_{1e}(c_t) = \log \frac{\sum_{l',l=0}^{M_s-1} \alpha_{t-1}(l') \exp\left(-\frac{\sum_{j=0}^{n-1}(r_{t,j} - x^1_{t,j}(l))^2}{2\sigma^2}\right)}{\sum_{l',l=0}^{M_s-1} \alpha_{t-1}(l') \exp\left(-\frac{\sum_{j=0}^{n-1}(r_{t,j} - x^0_{t,j}(l))^2}{2\sigma^2}\right)}. \tag{3.37}$$

M_s es el número de posibles estados en el codificador.

Para la primera iteración, las probabilidades a priori $p_t(1)$ y $p_t(0)$ en el primer decodificador se consideran equiprobables, es decir $p_t(1) = p_t(0) = 0{,}5$. Para diferenciar las probabilidades a priori del primer y segundo decodificador se añade el superíndice respectivo: $p^1_t(1), p^1_t(0)$ para el primer decodificador y $p^2_t(1), p^2_t(0)$ para el segundo decodificador.

Para el segundo decodificador, las probabilidades $p^2_t(1), p^2_t(0)$ se obtienen del primer decodificador:

$$\log \frac{p_t^2(1)}{p_t^2(0)} = \Lambda_{1e}(c_t). \tag{3.38}$$

De manera similar, para iteraciones posteriores en el primer decodificador, la información a priori se obtiene del segundo decodificador.

$$\log \frac{p_t^1(1)}{p_t^1(0)} = \Lambda_{2e}(c_t). \tag{3.39}$$

No se debe perder de vista que debido al entrelazador, el valor $\Lambda_{1e}(c_t)$ debe ser entrelazado antes de pasar al segundo decodificador, y el valor $\Lambda_{2e}(c_t)$ debe ser pasado a través del desentrelazador para poder ser empleado en el primer decodificador, ver figura 3.3.

Reescribiendo los valores $\Lambda(c_t)$, considerando las informaciones extrínsecas provenientes del otro decodificador se tiene entonces:

$$\Lambda_2(c_t) = \widetilde{\Lambda}_{2e}(c_t) + \frac{2}{\sigma^2}\widetilde{r}_{t,0} + \Lambda_{2e}(c_t) \tag{3.40}$$

$$\Lambda_1(c_t) = \widetilde{\Lambda}_{2e}(c_t) + \frac{2}{\sigma^2}r_{t,0} + \Lambda_{1e}(c_t). \tag{3.41}$$

Las principales características del algoritmo MAP se resumen en los siguientes puntos:

- Está enfocado a minimizar la probabilidad de símbolo erróneo.
- Requiere que los estados inicial y final dentro del enrejado sean conocidos.
- Requiere conocer la varianza del ruido en el canal de comunicaciones.
- La fortaleza del algoritmo radica en analizar minuciosamente dentro del enrejado las probabilidades de transición entre estados, las cuales van ligadas a la varianza del ruido del canal de comunicaciones, aunque cuando se trabaja con canales que presentan SNR altas, dicha varianza pierde importancia.
- De manera similar al algoritmo SOVA, requiere de un recorrido hacia adelante y de un recorrido hacia atrás del enrejado.
- Si el estado final dentro del enrejado no se conoce, lo cual es bastante común, se recomienda emplear $\beta_\tau(l)$/número de posibles estados del codificador.
- Aunque computacionalmente es más demandante que el algoritmo SOVA, presenta un mejor desempeño (0.55 dB para un BER de 10^{-5}, con un entrelazador de 4096 elementos y 18 iteraciones en la decodificación [5]).

3.3.3. Variantes de los algoritmos SOVA y MAP

Cuando se recorre el enrejado para una trama de un determinado tamaño, las decisiones se realizan con un retraso que está en función de dicho tamaño de trama. Para el SOVA estándar, se requiere almacenar los resultados de ambos recorridos (*backward* y *forward*), lo cual se puede volver costoso en términos de memoria requerida y retraso inherente. Para reducir lo anterior, existe una versión alterna del algoritmo SOVA, llamada SOVA de ventana móvil (*Sliding Window SOVA*), el cual se resume en los siguientes pasos [5]:

- Se divide la trama original en subtramas de tamaño D, para decodificar cada subtrama, las recursiones se extienden hasta la siguiente subtrama para poder generar métricas con buena verosimilitud.
- En el recorrido hacia adelante, se inicia en el instante $t = 0$, se calculan las métricas para cada nodo y cada instante de tiempo, y se almacenan; en el instante $t = 2D$, se selecciona la trayectoria con la menor métrica como la trayectoria ML y se calculan las decisiones "duras" para la primera subtrama.
- El recorrido hacia atrás arranca del instante $t = 2D$, se analizan las posibles trayectorias y no se almacena nada hasta el instante D, en este momento ya se tienen métricas con una aceptable verosimilitud. Se continua el recorrido hacia atrás hasta el instante $t = 0$.
- Cuando el recorrido hacia atrás alcanza el instante $t = D$, el decodificador ya puede comenzar a entregar sus decisiones sobre la primera subtrama conforme el recorrido hacia atrás siga avanzando, basándose en los recorridos *forward* y *backward*.
- Así, analizando las dos primeras subtramas se puede entregar un estimado de la primera subtrama.
- El proceso se repite, ahora el recorrido *forward* se extiende a la tercera subtrama, y el recorrido *backward* inicia desde la tercera subtrama, y así poder entregar estimados de la segunda subtrama.
- Aunque el recorrido *backward* se tenga que realizar dos veces (cada subtrama se recorre dos veces en dicho sentido), el retraso es mucho menor y los requerimientos de memoria son menores, la recursión *forward* requiere únicamente almacenar $2D.M_s$ métricas y la recursión backward solo M_s métricas (la memoria se reutiliza), donde M_s representa el número de estados dentro del enrejado trellis.

Para el algoritmo MAP, buscando reducir el costo en complejidad, se manejan logaritmos de las métricas, obteniéndose un algoritmo que trabaja en el dominio de los logaritmos. Este algoritmo se conoce simplemente como algoritmo Log-MAP. Se toman logaritmos de las ecuaciones 3.32, 3.33 y 3.34, obteniéndose una expresión para Λ en términos de sumas de exponenciales. Para simplificar dichas sumas de exponenciales, se emplea el algoritmo Jacobiano [26] el cual se puede expresar como:

$$\log(e^{\delta_1} + e^{\delta_2}) = max(\delta_1, \delta_2) + log(1 + e^{-|\delta_1 - \delta_2|}) \tag{3.42}$$

La función $log(1 + e^{-|\delta_1 - \delta_2|})$ puede almacenarse en tablas, lo cual facilita el algoritmo al reducir a una búsqueda en dicha tabla, en vez de estar ejecutando la función exponencial en repetidas ocasiones.

Una versión alterna y subóptima del algoritmo Log-MAP, computacionalmente más simple, es el llamado algoritmo Max-Log-MAP, en el cual el valor $log(1 + e^{-|\delta_1 - \delta_2|})$ se descarta, aproximándose el valor $\log(e^{\delta_1} + e^{\delta_2})$ simplemente como $\log(e^{\delta_1} + e^{\delta_2}) = max(\delta_1, \delta_2)$.

3.4. Resumen

En el presente capítulo se analizaron los algoritmos más representativos para la turbo decodificación. Cada uno conlleva sus propias ventajas y desventajas, el algoritmo SOVA

es más simple de implementar comparado con el algoritmo MAP, sin embargo, el algoritmo MAP trae consigo una mejora en el desempeño, comparado con el algoritmo SOVA.

Para contrarrestar la complejidad del MAP, se implementan variantes del mismo que reducen la complejidad sin sacrificar mucho el desempeño.

Cualquiera de los algoritmos presentados, tiene su fundamento en una estrategia de intercambio de información entre los decodificadores RSC del turbo decodificador, lo cual hace destaca la importancia del entrelazador y asegurar que las secuencias de los códigos 1 y 2 tienen la menor correlación posible entre ellas.

Por otra parte, otro elemento clave dentro del desempeño de un turbo código es el número de iteraciones que se realice en el proceso de la decodificación. Cada iteración adicional conlleva una mejora en la credibilidad de la decisión final a tomar, sin embargo, a partir de la iteración 8 aproximadamente para casi cualquier configuración de turbo código elegida, dicha ganancia se vuelve marginal mientras que el tiempo necesario para realizar el proceso iterativo sigue creciendo, por lo que, con base en las diferentes pruebas realizadas a lo largo de este trabajo de doctorado, se sugiere que un número recomendable de iteraciones es alrededor de 10, esta cantidad puede evidentemente variar de acuerdo a la calidad de la información recibida. Existen diversas estrategias para detener el proceso iterativo en la decodificación, algunas de ellas son:

- Detener el proceso si la diferencia entre la información recuperada entre la iteración anterior y la actual es menor a un umbral predeterminado.
- Detener el proceso si se ha alcanzado una tasa de bit en error objetivo.
- Detener el proceso si se ha alcanzado un número tope de iteraciones.

Estas estrategias pueden variar en cuanto a la complejidad de su implementación, aunque en general no suponen un aumento significativo en la complejidad total del decodificador. Una estrategia muy recomendable en la práctica puede ser una que combine varios de los criterios antes citados.

4. EL PERFORADO DE LOS CÓDIGOS

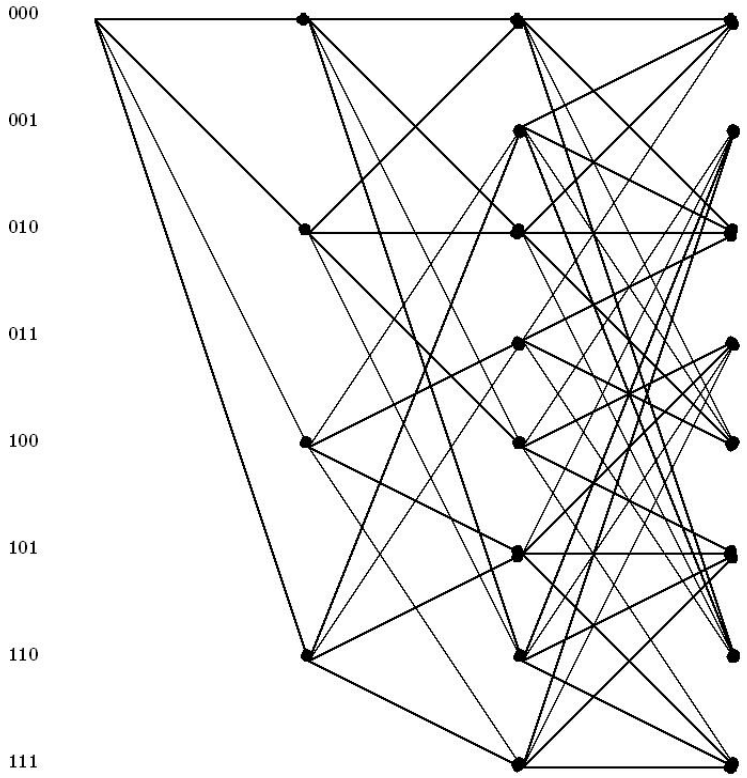
Se define el perforado como el proceso mediante el cual se cancelan de manera sistemática determinados símbolos codificados para de este modo, ser eliminados o no ser transmitidos.

La idea subyacente en el perforado es, a partir de un código de una tasa de codificación en definida, eliminar determinados símbolos para generar códigos de tasa más alta (menor redundancia). Una propiedad importante es que, dado que siempre se perfora la misma *i-ésima* coordenada en cada palabra codificada, el código resultante sigue siendo lineal [9].

4.1. Codigos convolutivos perforados

El perforado es muy empleado para el diseño de códigos convolutivos, una de las primeras referencias se puede encontrar en 1979 [27]. Además de modificar la tasa del código, el objetivo principal del perforado en los códigos convolutivos tiene que ver con el proceso de la decodificación. La figura 4.1 representa el enrejado trellis de un codificador de tasa $R = 2/3$ propuesto.

Fig. 4.1: Enrejado trellis para un codificador de tasa R=2/3



Como se puede observar en la figura 4.1, la complejidad de este enrejado es mucho más alta comparada con aquella de la figura 2.6. Como se menciono antes, la idea del perforado es construir un código de tasa alta a partir del perforado de otro código de menor tasa, al cual generalmente se le conoce como código base, código origen, o también código madre. En base a esta comparación de complejidad de los dos enrejados, la decodificación de un código de tasa alta es mucho más simple si dicho código se construye mediante el perforado que si se construyese el código de la tasa alta de manera directa. En general, para códigos de tasa $R = b/n$ con $b > 1$, conforme aumenta el valor de b, también aumenta el número de comparaciones que es necesario realizar en el algoritmo de decodificación para calcular las métricas, dado que en cada nodo existen 2^b ramas que salen y entran en cada instante de tiempo. Para decisión dura, se emplea el algoritmo de Viterbi y la métrica a considerar es la métrica de Hamming mientras que para decisión suave, la métrica a emplear se define en el algoritmo usado para la decodificación.

El proceso de perforado se realiza mediante una matriz de perforado, la cual se define como una matriz binaria de k x p elementos, en la cual un valor "0" define un elemento que será eliminado mientras que un valor "1" representa un elemento que se conserva. Por lo general, la regla de perforado sigue un patrón periódico, conocido como periodo de perforado el cual se abrevia como p.

Como ejemplo, sea la siguiente matriz de perforado:

$$P = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

la cual puede ser usada para, a partir de un código convolutivo de tasa $R = 1/2$, generar otro código de tasa $R = 2/3$. Al código original a partir del cual se construyen otros códigos mediante el perforado, se le conoce como código madre. En este ejemplo, el código madre es el de $R = 1/2$. Con base en la figura 2.3, la salida del código madre se puede escribir como:

$$\bar{v} = [(z_0^1, z_0^2), (z_1^1, z_1^2), (z_2^1, z_2^2), \ldots, (z_\tau^1, z_\tau^2), \ldots]$$

Con la matriz de perforado antes descrita, se obtiene un nuevo código de tasa $R = 2/3$, cuya salida es:

$$\bar{v} = [(z_0^1, z_0^2), (z_1^1), (z_2^1, z_2^2), (z_3^1), \ldots]$$

Para la decodificación de este nuevo código, se insertan símbolos de relleno en las posiciones donde se realizaron perforados. Dado que nunca fueron transmitidos, dichos símbolos de relleno no se consideran para el cálculo de las métricas dentro del enrejado trellis. De este modo, la decodificación de cualquier código derivado del mismo código madre se puede realizar con el mismo enrejado trellis. Lo anterior permite tener familias de códigos de tasa mayor al del código original en donde cualquiera de dichos códigos puede ser decodificado con el mismo enrejado trellis. La metodología para la construcción de la matriz P de perforado, se analiza a detalle en la sección 4.1.3.

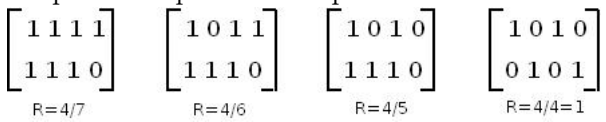
El proceso de perforado se debe realizar teniendo en cuenta que, aunque la complejidad del decodificador asociado se mantiene constante en comparación con el código base, el proceso de perforar implica enviar menos redundancia y por consiguiente, la distancia libre del código que se obtiene es menor. Se debe establecer un adecuado equilibrio entre cantidad de redundancia añadida a la información contra el desempeño del código por la reducción de la distancia libre.

Particular atención se debe tener en que cuando se realice el perforado, el código resultante no sea un código catastrófico. El código resultante debe conservar la característica de ser un código lineal.

4.1.1. Códigos convolutivos perforados de tasa compatible

Para optimizar el proceso de perforado, en 1988 J. Hagenauer [28] propone una estrategia de perforado llamada de tasa compatible. Esta estrategia consiste en que los patrones de perforado para tasa alta (menor redundancia) estén inmersos o formen parte dentro de los patrones de perforado de tasas inferiores. Lo anterior puede explicarse mejor mediante la figura 4.2

Fig. 4.2: Hipotéticos patrones de perforado de tasa compatible



Como se puede observar, el patrón de perforado de tasa $R = 1$ se encuentra incluido en el patrón de perforado de tasa $R = 4/5$ y este a su vez, inmerso en el de tasa $R = 4/6$ y así sucesivamente hasta llegar al código de tasa $R = 4/7$. Para determinar con base en la matriz de perforado la tasa de codificación, el periodo de la matriz de perforado (coincidente con el número de columnas de la matriz) representa el número de bits o símbolos a la entrada y el peso de Hamming de la matriz representa el número de bits o símbolos a la salida, de este modo, la tasa de codificación se puede calcular de una manera sencilla como $R = \frac{\text{num de columnas}}{\text{peso de Hamming de la matriz}}$.

Algunas de las posibles aplicaciones del concepto de códigos de tasa compatible son:

- Sistemas donde la tasa de transmisión se ajusta al seleccionar un patrón de perforado en particular manteniendo sin cambios el decodificador.
- Sistemas en los cuales se realiza una transmisión con tasa de codificación adaptable, de acuerdo a las condiciones de ruido o calidad de la información deseada, adaptando la tasa de codificación de canal empleada.
- Sistemas en donde la información sea jerarquizada y codificada de acuerdo a dicha jerarquización en función de su importancia.
- Sistemas híbridos FEC-ARQ (Forward Error Correction, Automatic Request Query) [29] en los cuales, para la información recibida que aun contenga errores, se envía una petición al codificador para que este envíe la redundancia necesaria para pasar al siguiente nivel de redundancia sin reenviar toda la información. Esto es, si el código de tasa más alta no tiene la capacidad necesaria para corregir determinado número de errores, entonces se envían únicamente los símbolos que previamente fueron perforados, necesarios para tener un código de menor tasa.

El perforado de tasa compatible no es el único medio para realizarlo, se pueden seguir diferentes criterios que van desde seleccionar las posiciones a ser perforadas de manera pseudoaleatoria hasta seguir criterios basados en la distancia libre de los códigos resultantes. En este sentido, el trabajo de P. Thitimajshima [14] realiza un estudio muy completo sobre códigos convolutivos sistemáticos perforados.

4.1.2. Turbo códigos perforados

La idea de extender el perforado hacia los turbo códigos viene de manera natural al tener sistemas con restricciones en el ancho de banda a usar y con malas condiciones de ruido. A continuación se mencionan algunas de las metodologías más importantes en la literatura científica:

- Ö. Açikel y W. E. Ryan [30] proponen una metodología basada en una búsqueda computacional tratando de optimizar la tripleta patrón de perforado, entrelazador y códigos *RSC* empleados en el turbo codificador, el parámetro de optimización es la distancia libre. Dada la complejidad de la tarea, acota esta búsqueda fijando los codificadores *RSC* a emplear, parte de un entrelazador pseudoaleatorio el cual es modificado buscando minimizar las salidas de peso 2 y 3 del turbo codificador. Finalmente, los patrones de perforado que se generan son adecuados para el par entrelazador optimizado - códigos *RSC* elegidos. Lo sobresaliente de esta metodología es que parte de una optimización de todos los elementos que componen el turbo codificador buscando que la degradación en términos de la distancia libre del turno código resultante sea la menor posible.

- F. Babich, G. Montorsi y F. Vatta [31] proponen una metodología para la construcción de patrones de perforado de tasa compatible partiendo de tres criterios y finalmente, eligiendo un criterio en donde se selecciona la posición a ser perforada como aquella que presente la mejor pareja (d_w, N_w), donde d_w representa el w-ésimo término del polinomio de ponderación de pesos del código resultante y N_w representa la cantidad de palabras código con un peso de Hamming de d_w, la mejor pareja significa el valor máximo de d_w combinado con el menor valor de N_w. Para el cálculo de las parejas (d_w, N_w), se considera un entrelazador uniforme como parte de la metodología de construcción. El entrelazador uniforme se define como una idealización de entrelazador que mapea una entrada de longitud k y peso w, a cualquiera de las $\begin{pmatrix} k \\ w \end{pmatrix}$ posibles permutaciones de manera equiprobable [32]. La ventaja que conlleva esta idealización es que facilita el cálculo del desempeño del turbo código resultante.

- E. Rosnes y Φ. Ytreyus [33], en su propuesta diseñan patrones de perforado de tasa compatible, enfocados a trabajar en la zona de piso de ruido (*error floor*). para su construcción, se basa en obtener un listado de las palabras código del turbo código en análisis, dicha lista requiere ser recalculada conforme el perforado se realiza. La posición a ser perforada es aquella que aparezca en el menor número de palabras código de bajo peso (afectar lo menos posible la distancia libre).

En general, se recomienda no perforar los símbolos correspondientes a la salida sistemática del turbo codificador original, ya que lo anterior puede degradar la probabilidad a posteriori de los decodificadores [30] o en determinado caso, puede llevar a la construcción de un código catastrófico. Otro elemento que muchos autores no consideran con la suficiente importancia es el entrelazador usado en el turbo codificador, en la mayoría de los casos se asume una idea de un entrelazador uniforme [32] y a partir del mismo se procede con el algoritmo. Esta abstracción del entrelazador es usada para facilitar el cálculo del espectro de distancias de los códigos resultantes y finalmente, ganar tiempo en la obtención de los patrones de perforado. La principal desventaja que esta abstracción representa es que, al

asumir un entrelazador con desempeño promedio, se dejan de lado aquellos entrelazadores que buscan maximizar la distancia libre del turbo código y descartan los beneficios que los mismos representan.

El considerar de manera muy puntual el entrelazador para la construcción de los patrones de perforado tiene como desventaja una pérdida de generalidad, el patrón de perforado que se obtiene es válido exclusivamente para el turbo código en análisis, pero se gana en el hecho de que dicho patrón puede tener un mejor desempeño en términos de la distancia libre del turbo código resultante, y si se considera en particular la zona de piso de ruido del desempeño de un turbo código, cualquier mejora en la distancia libre manteniendo la complejidad del sistema sin cambio, es sin duda alguna una ganancia recomendable.

La propuesta que se desarrolló el presente trabajo [34], se muestra a continuación:

4.1.3. Construcción de turbo códigos perforados

La propuesta realizada parte de las siguientes premisas:

- Considerando aplicaciones prácticas de comunicaciones inalámbricas, se desea trabajar con tramas pequeñas.
- Esta metodología se enfoca a SNR altas, buscando contrarrestar un poco el efecto de degradación de piso de ruido en los turbo códigos construidos (BER).

Las condiciones anteriores, buscando tener una metodología atractiva pensando en posibles implementaciones con restricciones en la latencia máxima permitida, limitantes en la memoria empleada y/o sistemas con requerimientos estrictos en la calidad de la información decodificada.

A partir de las condiciones antes citadas, la metodología propuesta queda de la siguiente manera:

1. El primer elemento a elegir para la construcción del turbo codificador base son los codificadores RSC a ser usados. En este sentido, se eligen los propuestos en el estándar UMTS [22] como punto de referencia.

2. Una vez elegidos los codificadores RSC a ser empleados, el siguiente paso es la elección del entrelazador. En este punto, dado que se cuenta de antemano con la premisa de trabajar con tramas pequeñas, nuestra recomendación es elegir un entrelazador del tipo determinístico, cuya construcción vaya ligada a los codificadores RSC elegidos previamente, que trate de maximizar la distancia libre del turbo código resultante y finalmente, que bajo condiciones de perforado, ayude a que la degradación del desempeño (BER) del turbo código resultante sea lo menor posible. En caso de trabajar con tramas de tamaño mayor a 1024 símbolos, cualquier entrelazador de tipo pseudoaleatorio que garantice romper la correlación entre las entradas de ambos codificadores es aceptable.

3. Una vez con los codificadores RSC y entrelazador elegidos, los bits de paridad de los codificadores RSC 1 y 2 son los únicos candidatos a ser perforados. Los bits correspondientes a la salida sistemática del turbo codificador se descartan a ser perforados por una parte para no degradar la distancia libre en la zona de piso de ruido y por otra, evitar que con el proceso del perforado se pudiese construir un

código catastrófico. Siguiendo una restricción de tasa compatible [28], se analizan cada una de las posibles posiciones candidatas a ser perforadas, aunque lo anterior pudiese resultar en un procedimiento tardado, esto garantiza que se elija la posición mas adecuada dando lo anterior como resultado un mejor desempeño en el turbo código resultante. El análisis antes mencionado consiste en calcular el polinomio de ponderación de pesos (duplas distancia libre - multiplicidad (d_w, N_w), con $w = 1, 2, \ldots$). La elección de la posición a ser perforada se hace con base en la elección de la mejor pareja (d_w, N_w), esta calidad de mejor es en el sentido de la mayor d_w con la menor N_w, una alta multiplicidad de las palabras de bajo peso es la principal responsable del efecto de piso de ruido. En caso de haber más de una posición con la mejor pareja (d_w, N_w), se procede a analizar el siguiente elemento del listado, es decir, se analiza la pareja (d_{w+1}, N_{w+1}), esto se realiza de manera iterativa hasta que se pueda elegir finalmente la posición a ser perforada.

El paso 3 se realiza de manera iterativa hasta alcanzar la tasa de codificación objetivo. En este momento, se cuenta con toda una familia de patrones de perforado desde la tasa original ($R = 1/3$) hasta la tasa objetivo. Un parámetro que no se incluye dentro de la metodología pero que sin duda es importante, es el tamaño de la matriz de perforado, es decir, la elección del periodo de perforado, p. En la figura 4.3 se muestra dicho parámetro. El renglón superior adicional corresponde a la salida sistemática, correspondiendo a todas sus posiciones en "1" ya que no se perfora ninguna de esas posiciones por las razones ya expuestas.

Fig. 4.3: Propuesta matriz de perforado de tasa R=4/6=2/3, p=4

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Un periodo de perforado pequeño ($4 \leq p \leq 10$) facilita el tiempo de obtención de los patrones de perforado correspondientes, un periodo mayor hace que el tiempo de cálculo, concretamente de los pares (d_w, N_w) sea más tardado. Se recomienda que la elección de este parámetro se realice de acuerdo a las necesidades del sistema donde se desee implementar (número de patrones de perforado que se desea construir, limitantes de memoria del sistema para almacenar el banco de patrones, etc).

Para el cálculo de las parejas (d_w, N_w), se emplea el algoritmo propuesto por R. Garello, P. Pierleoni y P. Benedetto [35], el cual permite incorporar de manera particular tanto los codificadores *RSC*, el entrelazador así como el patrón de perforado utilizados, obteniéndose de este modo, datos muy confiables sobre los valores de las parejas (d_w, N_w) obtenidas.

Para efectos de comparar esta metodología con algunas existentes [33], [31], se fijan los siguientes escenarios de prueba:

- Para comparar la metodología con la metodología propuesta por E. Rosnes y O. Ytrehus, [33] se fija un tamaño de trama $N = 100$ y se elige el entrelazador *UMTS*. Sin embargo, los autores mencionados no presentan los patrones de perforado que construyen, indicando únicamente las gráficas del desempeño de los mismos. Uno de los escenarios que ellos proponen como prueba de su metodología es el antes descrito, con lo cual, nuestro punto de comparación consiste en analizar la distancia libre que presentan a una determinada tasa de codificación. La tabla 4.1 muestra el

comparativo de distancias libres obtenidas para una tasa de codificación $R = 0{,}8$. El valor de distancia libre en nuestro caso se obtuvo con la ayuda del algoritmo propuesto por R. Garello, P. Pierleoni y P. Benedetto [35].

Tab. 4.1: Comparativo de valores de distancia libre, $R = 0{,}8$

Metodología	Valor de d_{free}
Rosnes & Itrehus	3.5
Metodología propuesta	4

- Para el comparativo con la metodología propuesta por F. Babich, G. Montorsi y F. Vatta [31], se elige nuevamente un tamaño de trama $N = 100$, se seleccionan 2 entrelazadores, entrelazador *PSI* y entrelazador pseudoaleatorio. Dado que no es un entrelazador muy grande, el patrón de perforado se construye con un periodo $p{=}N{=}100$, para lograr un empate uno a uno entre el patrón de perforado y los elementos del entrelazador. En este caso, los autores si proporcionan los patrones de perforado obtenidos para una tasa de codificación $R = 2/3$. La figura 4.4 muestra el desempeño, en términos del *BER* para los entrelazadores antes mencionados, empleando la metodología propuesta por F. Babich et.al [31] y nuestra metodología [34]. Las etiquetas en la figura significan:

  - EP, BM: Significa Entrelazador Pseudoaleatorio, patrón de perforado F. Babich, G. Montorsi y F. Vatta.
  - EP, NM: Significa Entrelazador Pseudoaleatorio, patrón de perforado construido con nuestra metodología propuesta.
  - EPSI, BM: Significa Entrelazador PSI, patrón de perforado F. Babich, G. Montorsi y F. Vatta
  - EPSI, NM: Significa Entrelazador PSI, patrón de perforado construído con nuestra metodología propuesta.

En ambos casos, se asume un canal con ruido *AWGN* y una fuente aleatoria de datos.

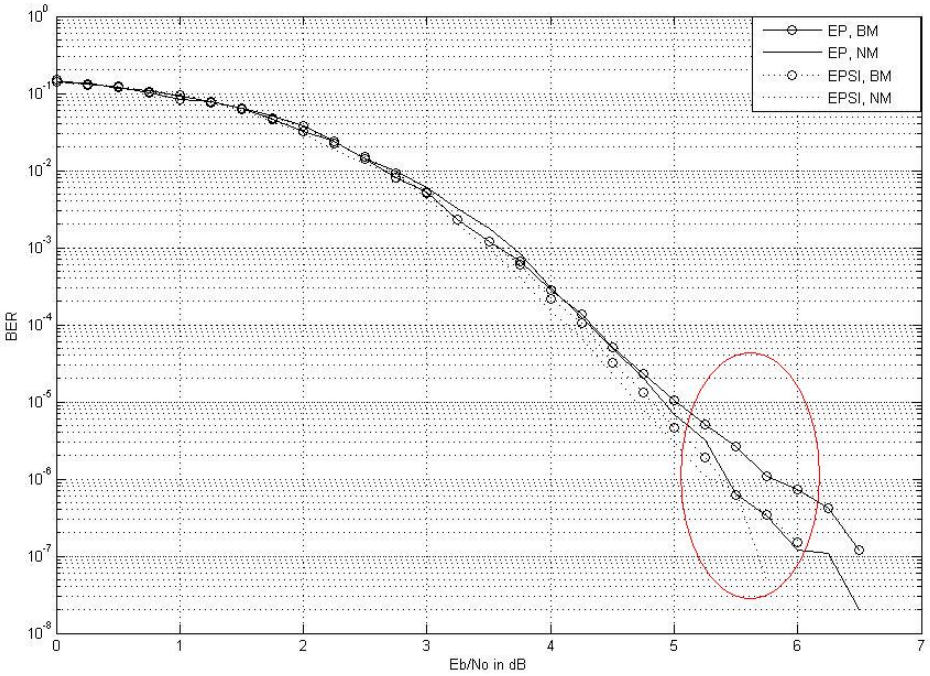
Con base en la tabla 4.1, aunque la mejora en la distancia libre obtenida a partir de usar nuestra metodología parece poco significativa, representa una mejora del casi 15 % sobre el valor leído del trabajo de E. Rosnes y O. Ytrehus [33]. La lectura de distancia libre en ambos casos es baja debido al tamaño del entrelazador empleado. Para entrelazadores mas grandes, la diferencia entre ambas lecturas de distancia libre resultaría mas evidente.

Como se puede observar en la figura 4.4, nuestra metodología presenta una ligera mejora en el desempeño del turbo código construido, particularmente observable en la zona de piso de ruido. La complejidad de nuestra metodología [34] comparada con las otras metodologías es mayor, considerando como referencia el tiempo necesario para obtener el patrón de perforado, sin embargo, dado que los patrones de perforado no se requieren calcular en tiempo real al momento del enlace de comunicaciones, esta complejidad mayor no representa una desventaja.

Las mejoras obtenidas con la metodología propuesta en este trabajo de tesis son, principalmente debidas a:

- Construcción del turbo codificador de manera ligada, los entrelazadores seleccionados consideran en su metodología los codificadores *RSC* seleccionados.

Fig. 4.4: Comparativo entre metododología Babich et. al y la propuesta en este trabajo, $R = 2/3$



- Cálculo de las parejas (d_w, N_w) de manera exacta: Evitar considerar al entrelazador como uniforme, conlleva explotar todas las posibles ventajas que el entrelazador pueda aportar, además de tener una decisión más confiable en cuanto se presentan varias posiciones candidatas a ser perforadas con similares pares (d_w, N_w). De manera adicional, la metodología propuesta por R. Garello, P. Pierleoni y P. Benedetto [35] es empleada como punto de referencia por otras metodologías lo que habla de su precisión, particularmente en entrelazadores pequeños.

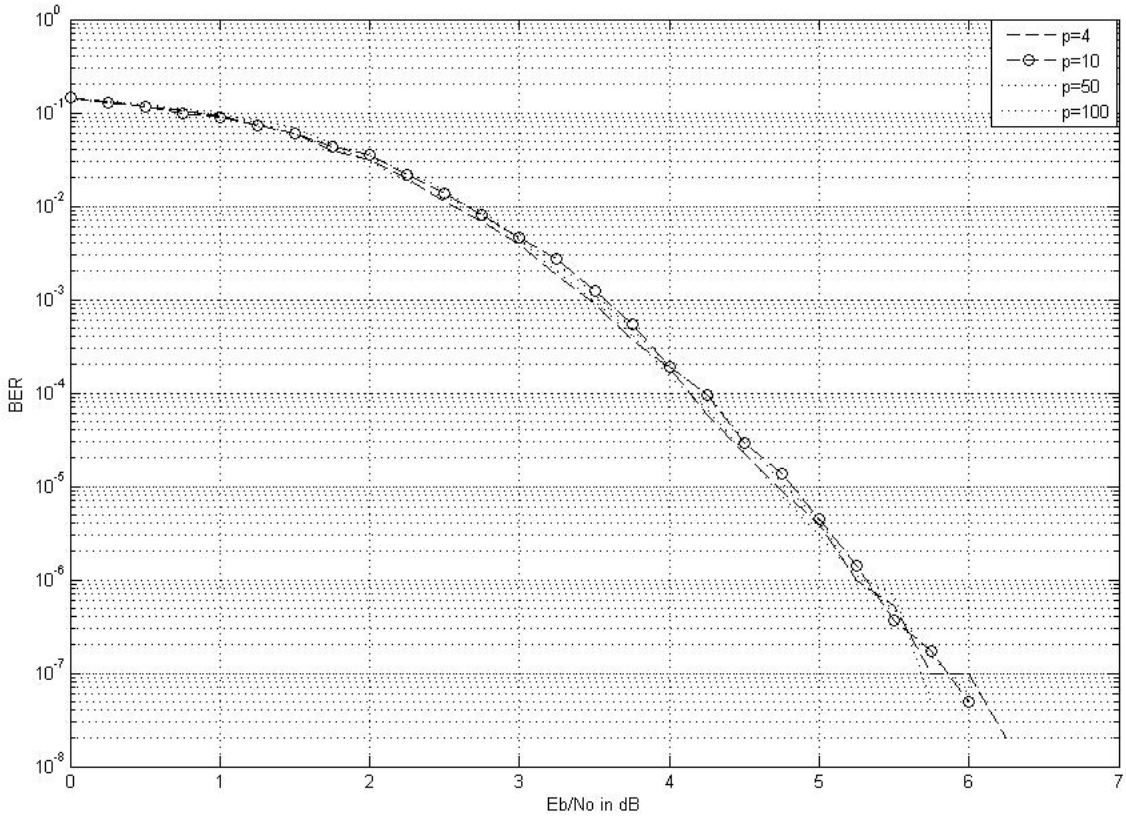
- Considerar como posiciones candidatas a ser perforadas todas las posiciones correspondientes a las salidas de paridad de ambos codificadores *RSC*.

Dada la naturaleza de la metodología, los patrones de perforado que se obtienen están totalmente ligados al turbo codificador construído, no es recomendable aplicarlos a otra configuración de turbo codificador. Se pierde generalidad con esta metodología, pero se gana en desempeño en la zona de piso de ruido.

Otro parámetro que debe ser considerado, es el periodo de perforado p de los patrones de perforado. Un periodo de perforado que tienda a ser tan grande como el tamaño del entrelazador es el que mejor desempeño, en términos de la distancia libre, presenta. Sin embargo, se debe considerar de manera muy puntual otras necesidades del sistema como es la memoria requerida; una matriz de perforado muy grande requiere de muchas localidades de memoria para ser almacenada, mientras que empleando periodos pequeños $(p \leq 12)$ se mantiene razonablemente la cantidad de memoria necesaria. Por otra parte, la ganancia obtenida empleando tamaños de p cercanos o iguales al tamaño del entrelazador no termina de justificar la mayor cantidad de memoria empleada. La figura 4.5 muestra

el comportamiento del desempeño en función del periodo de perforado, para una tasa de codificación $R = 2/3$

Fig. 4.5: Comparativo para diferentes valores de p, $R = 2/3$



Aunque en la figura anterior es difícil visualizar una diferencia notoria para los valores de p analizados, en la siguiente tabla se muestran los tres primeros valores de las parejas (d_w, N_w), obtenidas con el algoritmo propuesto por R. Garello, P. Pierleoni y P. Benedetto [35].

Tab. 4.2: Comparativo de valores de las parejas (d_w, N_w), $R = 2/3$

Periodo de perforado	Valores de las parejas (d_w, N_w), (d_{w+1}, N_{w+1}), (d_{w+2}, N_{w+2})
4	(5,5), (6,18), (7,70)
10	(6,17), (7,72), (8,300)
50	(7,71), (8,281), (9,1015)
100	(7,45), (8,264), (9,998)

La citada poca diferencia en la figura 4.5 se puede justificar con base en la tabla 4.2; aunque periodos de perforado mayores de p traen como consecuencia una mejor distancia libre, un efecto no deseado es una mayor multiplicidad de los términos N_w; tómese como referencia las parejas para los valores de $p = 4$ y $p = 10$, la distancia libre mejora en una unidad sin embargo, la multiplicidad pasa de 5 a 17, lo cual genera que en la gráfica

del desempeño, no haya una diferencia notoria entre las líneas correspondientes a los diferentes valores de p.

Con base en la figura 4.5 mostrada y en los valores de la tabla 4.2, se recomienda emplear periodos de perforado $p \leq 10$, usar valores más altos traerá una mejora muy poco significativa y en contraparte, el tiempo necesario para obtener los patrones de perforado correspondientes se incrementará a medida que el valor de p lo haga, además de que la cantidad de memoria requerida para almacenar dichos patrones de perforado será más grande (matrices de perforado de mayor longitud).

4.2. Resumen

En el presente capítulo se analizó el concepto y diseño del perforado en los turbo códigos. Perforar un turbo código es un modo de alcanzar tasas de codificación superiores (menor redundancia) sin aumentar la complejidad del turbo decodificador asociado. Por otra parte, reducir la redundancia añadida significa una menor capacidad de corrección de errores, lo cual se refleja principalmente en una reducción de la distancia libre en el turbo código asociado.

La restricción de tasa compatible añade una variable más al tema de la turbo decodificación, sin embargo, las posibles aplicaciones de la construcción de turbo códigos perforados de tasa compatible son extensas, similares a las de códigos convolutivos perforados de tasa compatible, pero con la ventaja de tener un esquema de codificación de canal con una capacidad de corrección de errores más alta con las consecuentes ventajas que esto representa.

La propuesta que se realiza en esta tesis para la construcción de turbo códigos perforados busca ser integral, considerar todos los elementos del turbo codificador de manera particular para, de este modo, tener un turbo código con la menor degradación posible en su desempeño. El patrón de perforado obtenido es exclusivo para la combinación de elementos de la cual se parta. Se pierde generalidad con este paso, sin embargo, se aprovechan todas las características de los elementos constitutivos del turbo codificador y se tiene como consecuencia, un mejor desempeño comparado con otras metodologías.

Aunque la metodología propuesta tiene una complejidad mayor, considerando tiempo de obtención de los patrones de perforado como parámetro de medición, se tiene a favor que la obtención de los patrones de perforado no es necesario calcularlos en tiempo real durante la transmisión, con lo cual el tiempo de obtención se vuelve un parámetro irrelevante.

5. MÉTODO PROPUESTO DE DISEÑO DE UN TURBO CÓDIGO CONSIDERANDO TODOS SUS COMPONENTES

Uno de los objetivos más importantes alrededor de un turbo codificador es: ¿como construir un turbo codificador? dado que actualmente, no existe un único turbo codificador que trabaje mejor que cualquier otro bajo cualquier ambiente. El cómo construir un turbo codificador que presente un buen desempeño, medido en términos de la probabilidad de error, va relacionado directamente con las condiciones del sistema en donde se desea utilizar.

Entre las condiciones más importantes se pueden considerar:

- Condiciones de razón señal a ruido SNR predominantes en el sistema: En función de las condiciones de ruido predominantes en el sistema, se haría la elección de elementos como los codificadores convolutivos RSC y en particular, para condiciones de SNR altas, debido al efecto de piso de ruido característico de los turbo códigos, la elección del entrelazador debe ser otro factor a tener en consideración.

- Limitantes de memoria disponible: Un sistema donde se cuente con memoria limitada para el codificador de canal, puede requerir que se empleen tramas pequeñas, lo cual conlleva a consideraciones para la elección del tamaño y diseño del entrelazador a ser empleado en el turbo codificador.

- Latencia del sistema: Si el sistema pudiese soportar aplicaciones que demanden retrasos muy pequeños (aplicaciones en tiempo real, por ejemplo), esto limita el tamaño del entrelazador a emplear, entrelazadores grandes requieren mayores tiempos de procesamiento, particularmente en el proceso de decodificación del turbo código correspondiente.

Bajo esta perspectiva, la propuesta que se realiza en esta tesis va enfocada a proporcionar un método sobre el como construir un turbo codificador.

5.1. Elección de los codificadores convolutivos RSC

El punto del cual se recomienda partir para la construcción del turbo codificador es la elección de los codificadores que constituirán el turbo código. Entre los aspectos que se deben considerar destacan:

- Longitud de restricción (K) de los codificadores convolutivos RSC: Entre mayor sea este parámetro, mejor es la distancia libre que los codificadores RSC presentan [14], sin embargo, la complejidad del trellis asociado aumenta en la misma proporción y por consiguiente, el tiempo de procesamiento requerido para la codificación y para la decodificación aumenta. En la referencia antes citada se presentan tablas con espectros de distancias para valores de K entre 3 y 9.

- Región de *SNR* donde se desea operar el turbo codificador: Cuando se trabaja con *SNR* bajas, es recomendable trabajar con codificadores *RSC* con valores más altos de K, dado que en esa zona, el fenómeno del piso de ruido depende de la distancia libre del turbo codificador construido. Sin embargo, para no aumentar en demasía la complejidad del trellis asociado a los codificadores *RSC* elegidos, se recomienda elegir un valor de K promedio, que pueda dar un buen desempeño tanto a *SNR* altas como bajas. En este sentido, el estándar *UMTS* [22] define los siguientes codificadores *RSC*:

  $$g_1(D) = 1 + D^2 + D^3$$

  $$g_2(D) = 1 + D + D^3$$

  En consecuencia se recomienda partir de estos codificadores *RSC* para la construcción del turbo codificador.

Una vez elegidos los codificadores *RSC* a ser usados, el siguiente paso es la elección del entrelazador.

5.2. Elección del entrelazador

La elección del entrelazador es otro de los puntos claves que determinan el desempeño de un turbo codificador, el entrelazador es el responsable de:

- Asegurar que las secuencias de entrada a los respectivos codificadores convolutivos tengan la menor correlación posible. Lo anterior ocasiona que en la decodificación, el intercambio de información entre los decodificadores *RSC* sea más eficiente.

- Construir un código de bloque de tamaño N. La presencia del entrelazador conlleva que el turbo codificador se vea en conjunto como un gran código de bloque de tamaño N, obteniéndose de este modo un código con una mucho mejor distancia libre comparado con los codificadores *RSC* a partir de los cuales se construye el turbo codificador. A esta ganancia se le conoce como ganancia de entrelazado, que se ve reflejada en una disminución de la probabilidad de bit erróneo en un factor de N [5]. Sin embargo, mientras mas grande sea el tamaño del entrelazador, los requerimentos de memoria crecen en el decodificador así como el tiempo necesario para decodificador una trama, con lo cual trabajar con tamaños de entrelazador muy grandes lo hace difícil de llevar a una implementación.

- Cuando se trabaja con tramas pequeñas, el entrelazador debe, de manera adicional, buscar descomponer las entradas de bajo peso, responsables de una distancia libre baja en el turbo código resultante.

Para lograr los puntos anteriores, es recomendable tomar como punto de partida el tamaño del entrelazador o el tamaño de trama a ser empleado. El tamaño de la trama a ser utilizada estará limitado por factores como la latencia máxima permitida en el sistema donde se desea implementar el turbo codificador, la memoria disponible entre otros.

Si se desea trabajar con tramas grandes, un entrelazador de naturaleza pseudoaleatoria es una buena opción, ya que para valores de $N > 1024$, este tipo de entrelazadores garantizan en buena medida que las funciones que debe de cumplir un entrelazador sean cumplidas (eliminar o al menos reducir la correlación de las secuencias de entrada a los

codificadores). Una recomendación para garantizar en mejor medida que el entrelazador pseudoaleatorio elegido sea adecuado, es construir un entrelazador *S-random* [18].

Por otra parte, si se desea trabajar con tramas pequeñas, un entrelazador tipo determinístico es la elección más adecuada. Como se menciona en el capítulo 2, existen diversas estrategias para la construcción de entrelazadores determinísticos, en general, para los entrelazadores determinísticos, se recomienda que la estrategia de selección considere:

1. Codificadores *RSC*, de este modo en el diseño global, el entrelazador quedará ligado a los codificadores y puede obtenerse una mejor distancia libre en el turbo código resultante. Se pierde generalidad con esta estrategia, pero se obtiene un turbo código con un mejor desempeño comparado con el caso de no considerar los codificadores *RSC*.

2. Si se desea trabajar con *SNR* altas, se propone que el entrelazador busque maximizar la distancia libre del turbo código resultante.

3. Que el entrelazador elegido pueda trabajar bajo condiciones de perforado, es decir, la distancia libre del turbo código no decaiga tanto por el efecto del perforado mismo. En este punto se propone probar diferentes entrelazadores candidatos para seleccionar aquel que menor degradación presente para una tasa de codificación dada.

En este sentido, un entrelazador que a lo largo de esta tesis y bajo diversos esquemas de prueba dio buenos resultados en términos de la distancia libre del turbo código resultante y de las curvas *Pe* vs *BER* obtenidas, es el entrelazador *DRP* [21] por lo cual se recomienda su uso para tramas pequeñas.

Finalmente, el último parámetro para la construcción de un turbo codificador de tasa adaptable a la aplicación, es la matriz de perforado, si es que se requiere trabajar con otras tasas de codificación mayores a $R = 1/3$.

5.3. Elección del patrón de perforado

Para la construcción de los patrones de perforado, se recomienda en general no perforar las posiciones correspondientes a la salida sistemática, para evitar construir un turbo código catastrófico. Por otra parte, para la estrategia de perforado que se elija se debe considerar tanto los codificadores *RSC* ya escogidos como el entrelazador empleado en el turbo codificador, de tal suerte que las posiciones más adecuadas a ser perforadas procuren que la distancia libre del turbo código se degrade lo menos posible.

Se recomienda emplear la estrategia de perforado propuesta en el capítulo 4 [34], la cual considera de manera integral los parámetros antes descritos, además de considerar una restricción de tasa compatible para la obtención de los patrones de perforado.

Aunque esta metodología esta diseñada pensando en turbo codificadores que empleen tramas pequeñas, al buscar tener una buena pareja (d_w, N_w), la metodología también puede ser aplicada a turbo codificadores con tramas más grandes. La desventaja en este caso, sería el tiempo necesario para el cálculo de las parejas (d_w, N_w) dentro de la metodología, lo anterior debido en gran medida al algoritmo empleado para esta tarea [35], sin embargo, se insiste en la importancia de contar con valores (d_w, N_w) precisos para que se pueda tomar una decisión confiable en la construcción del patrón de perforado.

Si el periodo p de la matriz de perforado es lo suficiente grande ($p > 20$) combinado con un entrelazador con $N > 1024$, se puede realizar el perforado basándose en una elección pseudoaleatoria entre las posiciones de paridad correspondientes a los codificadores 1 y 2 respectivamente. Se excluye directamente perforar las posiciones correspondientes a la salida sistemática del turbo codificador.

5.4. Ejemplo de construcción de un turbo codificador

En la figura 5.1 se hace un breve resumen de la metodología para la construcción del turbo código, considerando todas sus componentes.

Fig. 5.1: Diagrama de flujo para la construcción de un turbo codificador considerando todos sus componentes

Para dejar claros los puntos anteriores, se realizará la construcción de un turbo codificador para una tasa de codificación $R = 2/3$.

1. Se seleccionan como codificadores *RSC* los propuestos en el estándar UMTS [22], al ser codificadores que trabajan bien tanto para SNR altas como bajas.

2. Pensando en que se pudiese aplicar en algún sistema de comunicaciones inalámbrico, se selecciona un tamaño de trama de 256 elementos, el entrelazador seleccionado es el entrelazador *DRP* [21], debido a su buen desempeño para tramas pequeñas ($N < 1024$).

3. Para la construcción de los patrones de perforado, se parte de la elección del periodo de perforado, eligiendo $p = 16$ para de este modo contar con un banco de matrices de perforado para tasas desde $R = 16/47$ hasta $R = 16/24 = 2/3$. La tabla 5.1 muestra algunos de los patrones de perforado obtenidos para diferentes tasas de codificación.

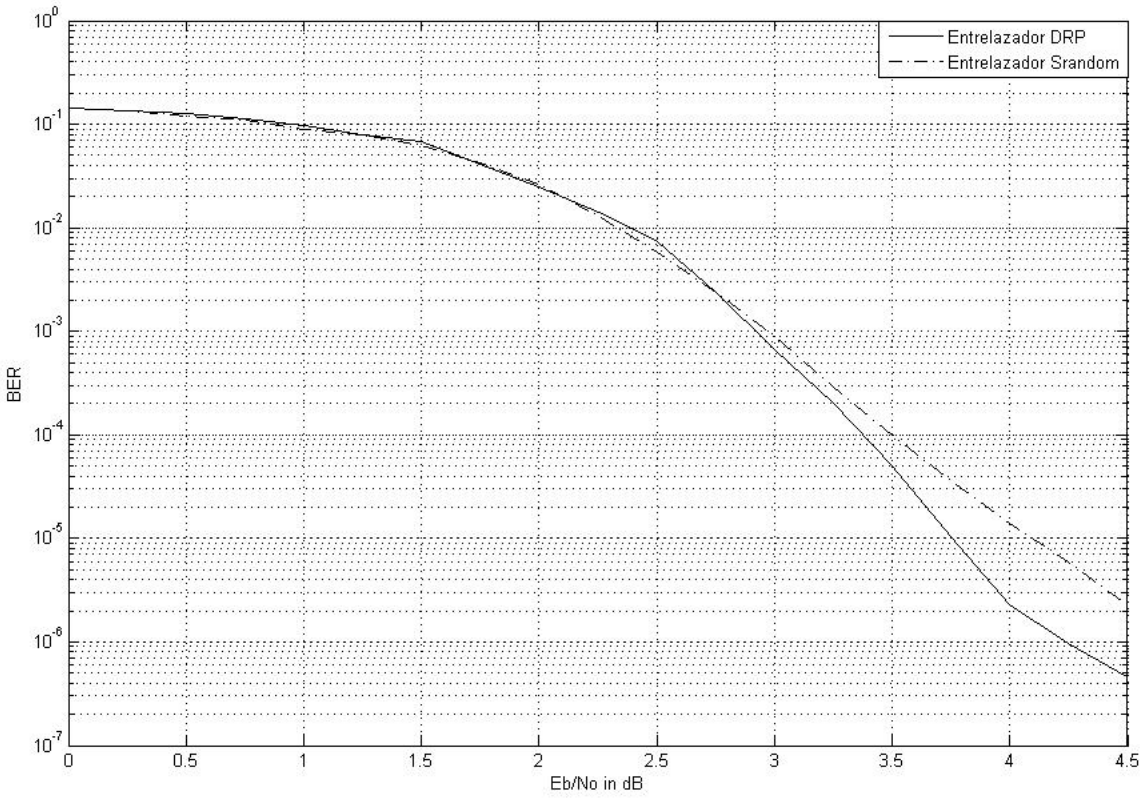
En la figura 5.2 se muestra el desempeño del turbo codificador construido en este ejemplo mientras que el segundo corresponde a un turbo codificador con R=2/3, codificadores

Tab. 5.1: Patrones de perforado obtenidos

Tasa de codificación	Patrón de perforado (octal)
R=16/44	(77773, 176677)
R=16/40	(67173, 176277)
R=16/36	(61173, 172077)
R=16/32=1/2	(21172, 162067)
R=16/28	(21072, 120063)
R=16/24=2/3	(1052, 20023)

RSC los propuestos en el estándar UMTS [22], entrelazador aleatorio con $N = 256$, un patrón de perforado con $p = 16$ donde las posiciones a ser perforadas se eligen de manera pseudoaleatoria entre la salida de paridad del primer y segundo codificador *RSC*. La salida sistemática no se considera para evitar construir un posible código catastrófico. Se remarca la diferencia del desempeño entre ambos escenarios, para hacer notar la mejora que se obtiene en el desempeño del turbo código obtenido aplicando la metodología de diseño propuesta.

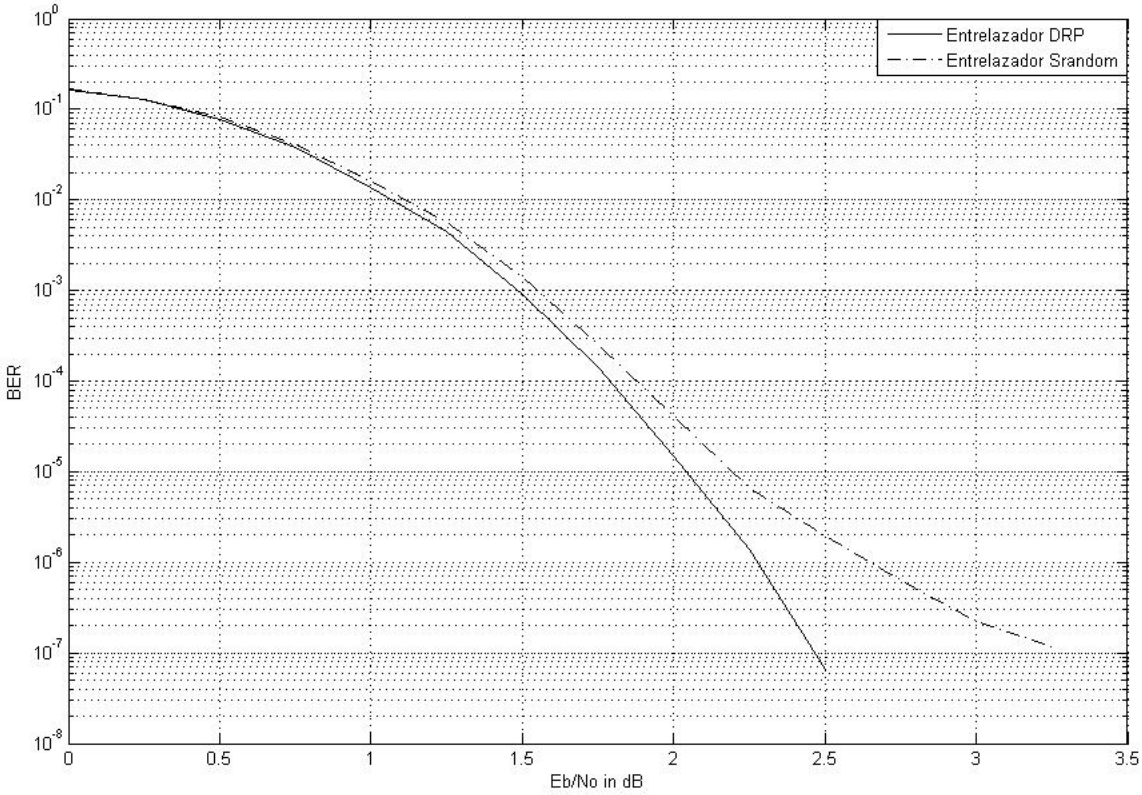
Fig. 5.2: Comparativo del desempeño para la metodología propuesta, *R=2/3*



Finalmente, la figura 5.3 muestra el desempeño de dos turbo codificadores para una tasa de codificación $R = 1/3$. En este caso, al no haber perforado, la diferencia en el desempeño de los dos turbo codificadores es notoria desde la zona de caída de agua

(*Waterfall region*), correspondiente a *SNR* medias. Lo anterior debido a la mejor distancia libre que presenta el entrelazador *DRP* comparado con un entrelazador aleatorio.

Fig. 5.3: Comparativo del desempeño para la metodología propuesta, *R=1/3*



5.5. Resumen

En el presente capítulo se presentó la metodología propuesta en esta tesis para el diseño de un turbo codificador considerando todas sus componentes. La mejora obtenida en el desempeño, particularmente para tasas de codificación altas (menor redundancia) es importante considerando que la complejidad del turbo codificador se mantiene constante además de mejorar el desempeño en especial en la zona de piso de ruido del turbo codificador diseñado.

La mejora en el desempeño del turbo codificador empleando la construcción ligada se vuelve más palpable en sistemas que requieran o permitan bajas tasas de errores durante el proceso de la transmisión.

6. CONCLUSIONES

En el presente trabajo de tesis se analizaron los conceptos de turbo codificación, turbo decodificación, entrelazado y perforado para turbo códigos. Aunque en el trabajo original de los turbo códigos [4] se propone un entrelazado de naturaleza pseudoaleatoria, esto no es recomendable cuando se trabaja con tamaños de tramas $N < 1024$ elementos. Esta diferencia es resaltada en este trabajo de tesis.

La construcción de un turbo codificador donde los elementos sean elegidos de manera ligada es un punto que hasta el momento solo existía como sugerencia [15], sin embargo no se había abordado de manera formal hasta el presente trabajo debido al número de variables involucradas en la construcciòn de un turbo codificador.

Para poder realizar lo anterior, se establecen ciertas condiciones bajo las cuales se desea opere el turbo codificador y a partir de estas condiciones se establecen criterios a seguir para la construcción concatenada. La metodología propuesta [36] se enfoca a situaciones de SNR altas, región conocida como piso de ruido en la cual el turbo código parece alcanzar un tope en su desempeño, siendo dificil lograr pequeñas mejoras en el mismo. Al mismo tiempo, se busca que la metodología propuesta sea factible de implementar en sistemas de comunicaciones con limitantes en aspectos como memoria disponible, latencia máxima permitida, por mencionar algunos. Para esto se seleccionan tramas pequeñas ($N < 1024$) y se elige como parámetros a cuidar las duplas distancia libre - multiplicidad (d_{free}, N_{free}) del turbo código resultante. El resultado es una metodología novedosa que permite mejorar el desempeño del turbo código resultante para las condiciones antes mencionadas sin que la complejidad del sistema aumente. Además, esta idea puede aplicarse a otras regiones de SNR logrando con esto turbo códigos con mejor desempeño. La combinación de elementos del turbo codificador que resultan de la aplicación de esta metodología garantiza un mejor desempeño que eligiendo los elementos de manera aislada para turbo codificadores con características equivalentes.

Otro tema que se abordó en este trabajo de tesis fue la construcción de patrones de perforado para turbo códigos. La metodología propuesta [34], a diferencia de otras presentes en la literatura ([30], [31], [33]), aborda de una manera integral el problema del perforado, considerando de manera particular los elementos que constituyen el turbo codificador para obtener de este modo un patrón de perforado que afecte lo menos posible la dupla distancia libre - multiplicidad (d_{free}, N_{free}) del turbo código resultante, obteniéndose de este modo turbo códigos apropiados para trabajar en la zona de piso de ruido. Otra ventaja que esta metodología presenta es que los patrones de perforado que se construyen incorporan la característica de tasa compatible, lo cual los hace mas atractivos para ser aplicados en diferentes escenarios, en la sección 4.1.1 del presente trabajo de tesis se mencionan algunos escenarios posibles.

El combinar ambas estrategias da como resultado turbo códigos de tasa compatible con un buen desempeño, particularmente en la región de piso de ruido además de que los mismos resultan aplicables en sistemas como los antes mencionados lo cual los hace atractivos para su implementación.

Lo novedoso del presente trabajo de tesis es que:

- Aborda problemas que no se habían atacado previamente, tal es el caso de la construcción de un turbo codificador de manera ligada.
- Mejora en el enfoque de problemas ya analizados, como es el caso de la construcción de patrones de perforado para turbo códigos perforados de tasa compatible.

Los resultados obtenidos no son exclusivos para trabajar en la zona de piso de ruido, se hace particular énfasis en la misma debido a que es una zona crítica en el desempeño de un turbo código en donde las duplas (d_{free}, N_{free}) determinan el desempeño del turbo codificador. Los turbo codificadores construídos con las metodologías propuestas pueden trabajar con un buen desempeño para relaciones señal a ruido menores.

Por otra parte, las ideas aquí presentadas pueden extenderse a sistemas que empleen tramas de mayor longitud, se recomienda tener cuidado en emplear una metodología para calcular las duplas (d_{free}, N_{free}) mas rápida que la aquí usada [35] la cual, aunque es muy precisa en sus resultados, resulta lenta para tramas grandes.

6.1. Publicaciones y participaciones

Como resultados concretos del trabajo realizado, se presenta a continuación un listado con las publicaciones realizadas así como participaciones en diversos eventos.

Publicaciones

- Lazcano-Salas, S. and García-Ugalde, F.J. *Diseño de un turbo código en condiciones de SNR altas utilizando tramas pequeñas.* Ing. Invest. y Tecnol., Dic 2010, vol.11, no.4, p.461-469. ISSN 1405-7743.
- Lazcano-Salas, S. and García-Ugalde, F.J. *An improved methodology to design rate compatible punctured turbo codes* Signal, Image and Video Processing, 2010, Volume 4, Number 4, Pages 405-408.
- Lazcano-Salas, S. and García-Ugalde, F.J. *A comparative study for turbo code interleavers designed for short frame size and relatively high SNR channel*, to be published in IEEE Proceedins of 5th Int. Conf. on Signal Proc. and Comm. Systems, 12-14 Dec. 2011. Status: accepted.

Participaciones

- *Codificación de Imágenes con Turbo Códigos Perforados Usando Protección contra Errores No Homogénea (UEP).* Sexto Coloquio Nacional de Códigos, Criptografía y Áreas Relacionadas. Ciudad de México, Junio 16 al 18, 2004.
- *Diseño de Turbo códigos de tasa compatible para SNR altas.* Séptimo Coloquio Nacional de Códigos, Criptografía y Áreas Relacionadas. Toluca, Estado de México, Junio 7 al 9, 2006.
- *Análisis de entrelazadores para turbo códigos empleando tramas pequeñas.* Octavo Coloquio Nacional de Códigos, Criptografía y Áreas Relacionadas. Ciudad de México, Octubre 28 al 30, 2008.

- *Algoritmos iterativos de control de errores para sistemas de transmisión de datos con baja señal a ruido.* Seminarios SIAV, Señales, imágenes y ambientes virtuales, 13 marzo 2009.

- *Entrelazadores para turbo códigos; entre lo deseable y lo realizable.* Noveno Coloquio Nacional de Códigos, Criptografía y Áreas Relacionadas. Ciudad de México, Septiembre 28 al 30, 2011.

6.2. Trabajo a realizar

Este trabajo de tesis fue enfocado a entender el comportamiento de los turbo códigos bajo condiciones de relación señal a ruido (SNR) altas, combinado con el uso de tramas pequeñas, buscando su posible implementación en sistemas de comunicaciones inalámbricas y en general en sistemas que presenten restricciones en aspectos como latencia máxima permitida, memoria disponible para el turbo decodificador y sistemas donde se requiera un esquema de protección contra errores muy eficiente.

Combinando la perforación con la restricción de tasa compatible, la gama de posibles aplicaciones se hace más amplia, se pueden extender hacia áreas como:

- Sistemas donde se jerarquice la información de acuerdo a su importancia. Cada nivel de jerarquización va asociado con una tasa de codificación.

- Sistemas de tasa de codificación adaptable empleando turbo códigos. El banco de codificadores necesarios se sustituiría por un banco de patrones de perforado que hagan posible el cambio de tasa de codificación.

- Sistemas híbridos FEC - ARQ empleando turbo codificación.

- Sistemas de transmisión tipo internet, en donde el principal problema no sean condiciones severas de ruido sino pérdidas de paquetes. Dichas pérdidas de paquetes pueden manejarse como perforado de un turbo código y de este modo, corregir dicha problemática.

Una aplicación que puede resultar de particular interés es el uso de la turbo codificación en sistemas PLC (Power Line Communications), en donde las condiciones de ruido son muy severas y se requiere de un esquema de codificación de canal robusto que pueda corregir la mayor cantidad de errores en el sistema.

Por otra parte, el tema del análisis de entrelazadores para turbo códigos es un tema con una veta de trabajo muy amplia. Aunque existen bastantes trabajos sobre turbo códigos, continuamente aparecen nuevos entrelazadores construidos bajo diferentes perspectivas, algunos de estos nuevos entrelazadores presentan mejores resultados en términos del desempeño del turbo codificador construido como es el caso de los entrelazadores basados en polinomios de permutación (QPP interleavers) [37]. Resulta atractivo un posible estudio donde se caracterice y compare nuevos entrelazadores y entrelazadores usados en la actualidad, la caracterización abarcaría métricas como correlación entre las secuencias de entrada y salida al entrelazador, esparcimiento del entrelazador [19], desempeño del turbo codificador resultante con y sin condiciones de perforado y facilidad de implementación a nivel hardware, pensando en aplicaciones de comunicaciones inalámbricas.

Apéndice

A. ALGEBRA DE MÁXIMA CREDIBILIDAD

Las pruebas de hipótesis, procedimiento empleado para determinar la validez de una hípótesis, son empleados en diversas ramas que requieran de inferencia estadística. En la codificación de canal, constituyen el núcleo de muchos algoritmos de decodificación basados en máxima credibilidad.

Estas pruebas tienen su fundamento en el teorema de Bayes el cual establece cómo calcular la probabilidad condicional y conjunta de dos eventos A y B de la siguiente manera:

$$P(A|B)P(B) = P(B|A)P(A) = P(A,B). \tag{A.1}$$

Reacomodando la ecuación A.1, se tiene:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}. \tag{A.2}$$

Esta ecuación se puede interpretar como la probabilidad a posteriori (APP) del evento A condicionada al evento B, en términos de la probabilidad condicional $P(B|A)$ así como de las probabilidades apriori $P(A)$ y $P(B)$.

Para aplicaciones en comunicaciones, partiendo de un canal *AWGN*, la forma más usual del teorema de Bayes queda de la siguiente manera [2]:

$$P(d=i|x)P(B) = \frac{p(x|d=i)P(d=i)}{p(x)}. \tag{A.3}$$

y

$$P(x) = \sum_{i=1}^{M} p(x|d=i)P(d=i), \tag{A.4}$$

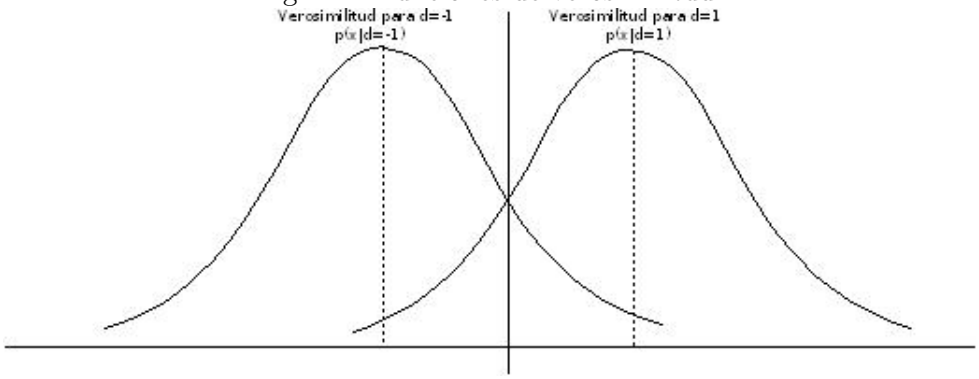
en donde $d=i$ representa el dato d, igual a una de las M posibles señales y $p(x|d=i)$ representa la función de densidad de probabilidad de la señal recibida (x) con ruido, condicionada a la señal d=i.

En la ecuación A.3, $p(x)$ representa un factor de escalamiento a partir de que su valor es el mismo para cualquier señal analizada. Esta ecuación puede interpretarse como el resultado de un experimento que involucra el recibir una señal y un cierto conocimiento estadístico de la ocurrencia de las posibles señales a las cuales la señal recibida pudiese pertenecer. Se conoce de antemano la probabilidad $P(d=i)$, es decir, se trata de una probabilidad a priori. Calcular la APP $P(d=i|x)$ puede interpretarse como un refinamiento del conocimiento a priori que se tenía.

En el caso de una señal binaria (+1, -1), es de particular interés para poder establecer un criterio de decisión sobre el posible valor de la señal recibida x.

Este criterio se le conoce como criterio de error mínimo, mejor conocido como maximum aposteriori (MAP) y se expresa de la siguiente manera:

Fig. A.1: Funciones de verosimilitud.



$$P(d=+1|x) \underset{H_2}{\overset{H_1}{\gtrless}} P(d=-1|x). \tag{A.5}$$

Esta ecuación puede leerse como: Se elige la hipótesis 1 (H_1) siempre que la APP $P(d=+1|x)$ sea mas grande que la APP $P(d=-1|x)$. En caso contrario, elegir la hipótesis 2 (H_2). La ecuación A.5 se expresa generalmente como una relación, y considerando el teorema de Bayes A.3, queda entonces:

$$\frac{p(x|d=+1)}{p(x|d=-1)} \underset{H_2}{\overset{H_1}{\gtrless}} \frac{P(d=-1)}{P(d=+1)}.$$

o equivalentemente

$$\frac{p(x|d=+1)P(d=+1)}{p(x|d=-1)P(d=-1)} \underset{H_2}{\overset{H_1}{\gtrless}} 1. \tag{A.6}$$

Para el análisis de esta ecuación, J. Hagenauer, E. Offer y L. Papke en 1996 [38] proponen trabajar con logaritmos, obteniendo una métrica muy importante la cual es conocida como relación logarítmica de máxima credibilidad, mejor conocida como LLR. Representa la salida suave del detector, representada como $L(d|x)$ de la siguiente manera:

$$L(d|x) = \log\left[\frac{P(d=+1|x)}{P(d=-1|x)}\right] = \log\left[\frac{p(x|d=+1)P(d=+1)}{p(x|d=-1)P(d=-1)}\right]. \tag{A.7}$$

$$L(d|x) = \log\left[\frac{p(x|d=+1)}{p(x|d=-1)}\right] + \log\left[\frac{P(d=+1)}{P(d=-1)}\right]. \tag{A.8}$$

$$L(d|x) = L(x|d) + L(d). \tag{A.9}$$

$L(x|d)$ representa la LLR de los valores observados de x bajo los supuestos de que se pudo haber transmitido $d=+1$ o $d=-1$, $L(d)$ es la LLR apriori del dato d.

El término $L(x|d)$ está asociado a las condiciones de ruido del canal, por lo que es usual encontrarlo expresado como $L_c(x)$.

B. ENTEROS MODULARES

El concepto de divisibilidad, fundamental en la teoría de números, constituye la base de las llamadas clases de congruencias. Una congruencia es en escencia, una afirmación acerca de la divisibilidad. La aritmética módulo es un sistema aritmético definido para las clases de congruencias de números enteros. La aritmética de enteros modulares tiene diversas aplicaciones, entre ellas música (escala de doce tonos que se repite periódicamente), criptografía, teoría de números y por supuesto, teoría de códigos por mencionar algunas.

Se presenta un bosquejo general de las bases para la aritmética módulo dada su importancia en general dentro de la teoría de códigos.

B.1. Clases de congruencias

Antes de definir de manera formal que es una clase de congruencia, es necesario definir algunos conceptos previos [39].

Definición B.1.1. Si un entero m, diferente de cero, divide a la diferencia a-b, se dice que a es congruente con b módulo m y se escribe $a \equiv b(\text{mod } m)$. Si a-b no es divisible entre m, se dice que a no es congruente con b módulo m y se escribe como $a \neq (\text{mod } m)$

Las congruencias tienen muchas propiedades en común con las igualdades, entre las cuales se tienen:

Si a, b, c, d, x, y son números enteros. entonces

- $a \equiv b$ (mod m), $b \equiv a(\text{mod } m)$ y $a - b \equiv 0$ son proposiciones equivalentes.
- Si $a \equiv b$ (mod m) y $b \equiv c$ (mod m), entonces $a \equiv c$ (mod m).
- Si $a \equiv b$ (mod m) y $c \equiv d$ (mod m), entonces $ax + cy \equiv bx + dy$ (mod m).
- Si $a \equiv b$ (mod m) y $c \equiv d$ (mod m), entonces $ac \equiv bd$ (mod m).
- Si $a \equiv b$ (mod m) y $d|m$, $d > 0$, entonces $a \equiv b(\text{mod } d)$

Al trabajar con enteros módulo m, se realizan en escencia las operaciones básicas de la aritmética, evitando los múltiplos de m y se trabaja únicamente con los valores $0, 1, \ldots, (m-1)$ es decir, con los residuos de m. En general, no se distingue entre un valor a y un valor $a + mx$, donde x es un entero cualquiera, es decir, todo entero es congruente módulo m para uno de los valores $0, 1, 2, \ldots, (m-1)$. Estos valores m constituyen un sistema completo de residuos módulo m.

Definición B.1.2. Si $x \equiv y(\text{mod } m)$ entonces y recibe el nombre de residuo x módulo m. Un conjunto $x_1, x_2, \ldots, x_m$ es un sistema completo de residuos módulo m si para todo entero y existe uno y solamente un x_j tal que $y \equiv x_j(\text{mod } m)$.

El estudio de la teoría de números, especialmente las clases de congruencia, aporta importantes elementos algebráicos para trabajar con la codificación de canal en general y con la teoría de códigos en particular. Concretamente, se trabajan con estructuras algebráicas como grupos y anillos, los cuales son definidos a continuación.

Definición B.1.3. Un grupo G es un conjunto de elementos $a, b, c, \ldots$ junto con una operación unívoca, $\oplus$ tal que:

1. El conjunto es cerrado bajo la operación;
2. El conjunto es conmutativo, es decir:

$$a \oplus b = b \oplus a$$

3. Cumple con la ley asociativa, es decir:

$$a \oplus (b \oplus c) = (a \oplus b) \oplus c$$

   para todos los elementos a, b, c en G

4. El conjunto tiene un elemento identidad único, e
5. Cada elemento tiene un inverso único en G.

La operación $\oplus$ se puede definir como la adición ordinaria, con la característica de que el resultado de la operación es otro elemento del grupo, es decir:

$$a \oplus b = (a + b)(\text{mod })m.$$

Para esta operación, su elemento identidad es el elemento neutro. Las tablas B.1 y B.2 muestran la operación $\oplus$ para enteros módulo 2 y modulo 6 respectivamente. Se resalta nuevamente, que el conjunto de enteros módulo 2 o conjunto binario, es el mas empleado en teoría de códigos pero no el único.

Tab. B.1: Operación suma en $\mathbb{Z}_2$

$\oplus$	0	1
0	0	1
1	1	1

Tab. B.2: Operación suma en $\mathbb{Z}_6$

$\oplus$	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	2	3	4	5	0
2	2	3	4	5	0	1
3	3	4	5	0	1	2
4	4	5	0	1	2	3
5	5	0	1	2	3	4

Tab. B.3: Operación producto en $\mathbb{Z}_2$

$\otimes$	0	1
0	0	0
1	0	1

Tab. B.4: Operación producto en $\mathbb{Z}_6$

$\otimes$	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	2	4	0	2	4
3	0	3	0	3	0	3
4	0	4	2	0	4	2
5	0	5	4	3	2	1

Otra importante operación es el llamado producto cruz, $\otimes$, el cual se define como el producto ordinario pero el resultado es a su vez elemento del grupo.

$$a \otimes b = (ab)(\text{mod })m$$

Para esta operación, su elemento identidad es el elemento unitario. Las tablas B.3 y B.4 muestran la operación $\otimes$ para enteros módulo 2 y enteros módulo 6 respectivamente.

El interés en la teoría de códigos se resume a los grupos finitos y que cumplan con las condiciones anteriores tanto para la operación $\oplus$ y la operación $\otimes$. A dichos grupos de interés se les conoce como anillos. El grupo mas empleado por su facilidad de implementación es el grupo binario, ya que sus dos elementos, 0 y 1 se pueden representar como un nivel bajo o alto de voltaje, respectivamente o simplemente como un apagado-encendido.

Definición B.1.4. Un anillo es un conjunto de al menos dos elementos, con dos operaciones binarias $\oplus$ y $\otimes$, tal que el conjunto en análisis es conmutativo bajo $\oplus$, cerrado bajo $\otimes$, además de que $\otimes$ es una operación asociativa y distributiva respecto a $\oplus$. El elemento identidad respecto a $\oplus$ recibe el nombre de cero del anillo. Si todos los elementos del grupo, excepto el cero del anillo son conmutativos bajo $\otimes$, entonces el anillo recibe el nombre de campo.

Bibliografía

[1] H. Nyquist, "Certain topics in telegraph transmission teory," *AIEE*, pp. 617–644, 1928.

[2] B. Sklar, "A primer on turbo code concepts," *IEEE Communications Magazine*, vol. 35, pp. 94–102, December 1997.

[3] C. E. Shannon, "A mathematical theory of communication," *Bell System Tech.*, vol. 27, pp. 379–423 and 623–656, 1948.

[4] C. Berrou, A. Glavieux, and P. Thitimajshida, "Near shannon limit error-correcting coding and decoding: Turbo codes," in *International Conference on Communications*, pp. 1064–1070, 1993.

[5] B. Vucetic and J. Yuan, *Turbo Codes: Principles and Applications*. Kluwer Academic Publishers, 2000.

[6] B. Sklar, *Digital Communications, Fundamentals and Applications*. Prentice Hall, second ed., 2001.

[7] R. H. Morelos-Zaragoza, *The Art of Error Correcting Coding*. John Wiley & Sons, second ed., 2006.

[8] G. D. Forney, *Concatenated Codes*. M.I.T. Press, Cambridge, MA, 1966.

[9] W. C. Huffman and V. Pless, *Fundamentals of Error-Correcting Codes*. Cambrigde University Press, 2003.

[10] W. W. Peterson and E. J. Weldon, *Error-correcting Codes, Secon Edition*. The MIT Press, 1972.

[11] M. Bossert, *Channel Coding for Telecommunications*. John Wiley & Sons, 1999.

[12] P. Elias, "Coding for noisy channels," *IRE Conv. Rec.*, vol. 3, pp. 37–46, 1955.

[13] J. L. Massey and M. K. Sain, "Inverses of linear sequential circuits," *IEEE Trans. Computers*, vol. AC-A2, pp. 330–337, April 1972.

[14] P. Thitimajshima, "Les codes convolutifs récursifs systématiques et leur aplication à la concaténation parallèle," Master's thesis, l´ Université de Bretagne Occidentale, 1993.

[15] M. Valenti, *Iterative Detection and Decoding for Wireless Communications*. PhD thesis, Virginia Polytechnic Institute and State University, July 1999.

[16] J. Hokfelt, O.Edfords, and T. Maseng, "On the theory and performance of trellis termination methods for turbo codes," *IEEE J. Sel. Areas Comm.*, vol. 19, pp. 838–847, May 2001.

[17] P. Robertson, "Illuminating the tructure of parallel concatenated reursive systematic (turbo) codes," in *GLOBECOM'94*, pp. 1298–1303, Nov. 1994.

[18] D. Divsalar and F. Pollara, "Multiple turbo codes," in *IEEE Military Communications Conference, MILCOM'95*, vol. 1, pp. 279–285, November 5-8 1995.

[19] S. Crozier, "New high-spread high-distance interleavers for turbo-codes," in *20th Biennal Symposium on Communcations*, pp. 3–7, Queen´s University, May 28-31 2000.

[20] D. Wang and H. Kobayashi, "On design of interleavers with practical size for turbo codes," in *IEEE International Conf on Comm. ICC 2000*, vol. 2, pp. 618–622, 2000.

[21] S. Crozier and P. Guinand, "High performance low-memory interleaver banks for turbo codes," in *IEEE Vehicular Technology Conference, VTC2001*, pp. 2394–2398, October 7-11 2001.

[22] T. S. Group, "Multiplexing and channel coding (tdd)," Tech. Rep. TS 125.212 V6.7.0, European Telecommunications Standards Institute, December 2005.

[23] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inf. Theory*, vol. IT13, pp. 260–269, April 1967.

[24] L. Bahl, J. Jelinek, J. Raviv, and F. Raviv, "Optimal decoding of linear codes for minimising symbol error rate," *IEEE Transactions on Information Theory*, vol. IT-20, pp. 284–287, March 1974.

[25] J. Hagenauer and P. Hoeher, "A viterbi algorithm with soft-decision outputs and its applications," in *IEEE GLOBECOM*, vol. 3, pp. 1680–1686, 1989.

[26] J. Erfanian, S. Pasupathy, and G. Gulak, "Reduced complexity symbol detectors with parallel structures for isi channels," *IEEE Trans. Commun.*, vol. 42, pp. 1661–1671, Feb/March/April 1994.

[27] J. B. Cain, G. C. Clark, and J. M. Geist, "Punctured convolutional codes of rate (n-1)/n and simplified maximum likelihood decoding," *IEEE Transactions on Information Theory*, vol. IT-25, pp. 97–100, Jan 1979.

[28] J. Hagenauer, "Rate-compatible punctured convolutional codes (rcpc codes) and their applications," *IEEE Transactions on Communications*, vol. 36, pp. 389–400, April 1988.

[29] D. Rowitch and L. B. Milstein, "On the performance of hybrid fec/arq systems using rate compatible punctured turbo (rcpt) codes," *IEEE Trans. on Comm.*, vol. 48, pp. 948–959, June 2000.

[30] . Açikel and W. E. Ryan, "Punctured turbo codes for bpsk/qpsk channels," *IEEE Transactions on Communications*, vol. 47, pp. 1315–1323, Sept 1999.

[31] F. Babich, G. Montorsi, and F. Vatta, "Design of rate-compatible punctured turbo (rcpt) codes," in *IEEE Int. Conf. on Comm*, vol. 3, pp. 1701–1705, 2002.

[32] S. Benedetto and G. Montorsi, "Unveiling turbo codes: Some results on parallel concatenated coding schemes," *IEEE Trans. on Inf. Theory*, vol. 42, pp. 409–428, March 1996.

[33] E. Rosnes and O. Ytrehus, "On the construction of good families of rate-compatible punctured turbo codes," in *Int. Sym. on Inf. Theory, ISIT 2005*, pp. 602–606, Sep 4-9 2005.

[34] S. Lazcano-Salas and F. García-Ugalde, "An improved methodology to design rate compatible punctured turbo codes," *Springer Signal, Image and Video Processing*, vol. 4, pp. 405–408, November 2010.

[35] R. Garello, P. Pierleoni, and P. Benedetto, "An algorithm to compute the free distance of turbo codes," *IEEE Journal on Selected Areas in Communications*, vol. 19, pp. 800–812, May 2001.

[36] S. Lazcano-Salas and F. García-Ugalde, "Desempeño de un turbo código en condiciones de snr altas utilizando tramas pequeñas," *Ingeniería, Investigación y Tecnología*, vol. XI, pp. 461–469, Octubre 2010.

[37] E. Rosnes and O. Y. Takeshita, "Optimum distance quadratic permutation polynomial-based interleavers for turbo codes," in *IEEE Int. Symposium on Information Theory, ISIT 2006*, pp. 1988–1992, July 9-14 2006.

[38] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. on Inf. Theory*, vol. IT-42, pp. 429–445, March 1996.

[39] I. Niven and H. S. Zuckerman, *Introducción a la teoría de los números.* John Wiley & Sons, 1966.