Procura de Codificadores BGU para utilização em Códigos com Concatenação Serial

Manish Sharma e Jaime Portugheis

***Resumo*— Este artigo apresenta um método para a procura de codificadores a serem utilizados em códigos com concatenação serial, baseado em diretrizes de projeto previamente estabelecidas. Utiliza-se do conceito de codificadores binariamente geometricamente uniformes (BGU) para se garantir que as distâncias Euclidiana mínima e efetiva entre as seqüencias geradas são equivalentes aos pesos Euclidiano mínimo e efetivo. A utilização deste tipo de codificadores também garante a propriedade de uniformidade de erro de bit (UBEP). Um exemplo da utilização do método é mostrado.**

***Palavras-Chave*— Códigos Convolucionais, Concatenação Serial, Codificadores BGU, UBEP.**

***Abstract*— This paper presents a method for the search of encoders to be used in a serially concatenated coding scheme, based on previously determined project guidelines. The concept of Bit Geometrically Uniform (BGU) encoders is used to assure that the minimum and free effective euclidian distances between sequences are equivalent to the minimum and free effective euclidian weights. By using this type of encoders we also guarantee the uniform bit error property (UBEP). One example of the method's usage is shown.**

***Keywords*— Convolutional Codes, Serial Concatenation, BGU encoders, UBEP.**

I. INTRODUÇÃO

Códigos Turbo apresentam desempenhos muito próximos dos limites teóricos com uma complexidade de implementação razoável. Esses códigos se baseiam na concatenação paralela de dois códigos mais simples, utilizando um entrelaçador entre eles. Códigos com concatenação serial (CCS) diferem dos códigos turbo por concatenar um código externo com um código interno, também através de um entrelaçador. Esses códigos apresentam, em alguns casos, desempenho melhor do que os códigos Turbo.

O projeto de um sistema com concatenação serial difere do projeto de um sistema com concatenação paralela porque existe uma dependência maior entre os códigos no primeiro caso. Assim, somos limitados na escolha de um código interno dado um código externo, considerando propriedades deste para projetar o código interno de modo a tornar o desempenho do sistema o melhor possível.

O limitante superior analítico para a probabilidade de erro de bit é dominado pela distância euclidiana de seqüências de informação que geram seqüências internas com um peso de Hamming específico. Esta distância é chamada de distância Euclidiana livre efetiva, d_{ef}. O código interno deve ser preferencialmente recursivo para que haja sempre um ganho ao se utilizar um entrelaçador.

Em [1], rotulamentos e codificadores binariamente geometricamente uniformes (BGU) foram introduzidos e utilizados para projetar um codificador interno de um CCS. Os resultados obtidos foram promissores, com um aumento significativo na d_{ef}. Condições para a existência destes rotulamentos e códigos foram apresentados. Entretanto, um método para a construção de tais códigos com bons parâmetros não foi proposto. Tentamos preencher esta lacuna através de um método que permite controlar razoavelmente o peso Euclidiano de segmentos críticos de seqüências internas que dominam o limitante analítico da probabilidade de erro de bit.

Na seção II, revisamos os rotulamentos e codificadores BGU. Na seção III, revisamos os limitantes de probabilidade de erro de bit para CCS. Na seção IV, expomos o nosso método e mostramos que ele gera codificadores BGU, exemplificando. E na seção V, analisamos os resultados obtidos.

II. ROTULAMENTOS E CODIFICADORES BGU

Seja uma constelação geometricamente uniforme(GU) S com grupo gerador G, onde as regiões de decisão são congruentes. Seja um rotulamento $\varepsilon[s_i] : S \leftrightarrow H_k$, $s_i \in S$, e H_k o espaço de Hamming com k bits. Como há uma relação um para um entre os elementos de G e os de S, podemos dizer que ε é um rotulamento $G \leftrightarrow H_k$. Se o rotulamento ε satisfaz a seguinte propriedade :

$$d_h(\varepsilon(g_i), \varepsilon(g_j)) = w_h(\varepsilon(g_j^{-1} \cdot g_i)) \ \forall g_i, g_j \in G, \qquad (1)$$

onde $d_h(a, b)$ é a distância de Hamming entre a e b, e $w_h(a)$ é o peso de Hamming de a, ele possui a propriedade de uniformidade de erro de bit (UBEP), i.e., a probabilidade de erro de bit independe da seqüencia enviada [1]. Assim, a uniformidade geométrica da constelação se extende à uniformidade de bit, e ε é binariamente geometricamente uniforme(BGU). A UBEP nos permite analisar a probabilidade de bit observando somente um dos sinais transmitidos. Vemos na figura 1 (a) um exemplo de rotulamento BGU para a constelação 8-PSK, utilizando o grupo D_4 como gerador.

Analogamente, a probabilidade de erro de bit, $P_b(e)$, em codificadores BGU independe da seqüencia transmitida. Logo, podemos analisar todo o desempenho do código observando somente os pesos das seqüencias transmitidas, já que estes pesos são as distâncias para a palavra toda nula. Códigos de treliça (do inglês: Trellis Coded Modulation - TCM) com codificadores BGU são matematicamente descritos através do seguinte teorema [1]:

Manish Sharma e Jaime Portugheis¸ Departamento de Comunicações, Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Campinas, Brasil, E-mails: sharma@decom.fee.unicamp.br, jaime@decom.fee.unicamp.br. Este trabalho foi parcialmente financiado pelo CNPq (bolsa 132154/2004-8) e FAPESP (projeto 02/07473-7.)

Teorema 1: Seja S uma constelação GU e $C \subseteq S^Z$ um TCM GU invariante no tempo sobre S e T um dos grupos de uma seção de treliça. Se um codificador binário $E[C,k]{:}C \rightarrow H_k$ satisfaz:

$$d_h(E(t_i), E(t_j)) = w_h(E(t_i \cdot t_j^{-1})) \ \forall t_i, t_j \in T, \qquad (2)$$

ele possui a UBEP. ∇

O seguinte teorema de [1] contém as condições necessárias e suficientes para que um codificador seja BGU:

Teorema 2: Seja S uma constelação GU, e $C \subseteq S^Z$ um TCM GU invariante no tempo sobre S e T um dos grupos de uma seção de treliça. Seja A um subconjunto de T que contém todos os ramos que são rotulados pela palavra toda nula. Seja S_0 um subconjunto de S com os sinais associados aos ramos que partem do estado identidade. Seja F_0 o subconjunto de T que contém todos os ramos que partem do estado identidade. Um codificador binário $E[T,k]$ é BGU se e somente se as seguintes condições são satisfeitas para pelo menos um dos possíveis T's:

- $E_0[F_0, k]$ é um rotulamenteo BGU para F_0;
- $E_j(f \cdot a) = E_0(f)$,para todo $f \in F_0$, e para todo $a_j \in A$;
- A é um subgrupo de T, i.e., $T = F_0 \bullet A$, onde $\bullet$denota o produto semi-direto;
- Para todo $a_j \in A$, $w_h(E_0(f)) = w_h(E_0(f'))$, com $f' = a_j^{-1} \cdot f \cdot a_j$. ∇

Logo, a procura de um codificador BGU consiste em procurar E_0 e A.

Todo este formalismo é necessário porque precisamos de algum tipo de estrutura para poder manipular os codificadores. Ao nos restringirmos a codificadores BGU, arriscamos não obter a solução ótima. Embora todos os codificadores BGU possuam a UBEP, é possível que codificadores não BGU a possuam também. Como motivação mostramos a figura 1 (b), que possui a UBEP mas não é BGU. Para rotulamentos BGU, é trivial mostrar que $w_H(E(g)) = w_H(E(g^{-1}))$. A constelação 8PSK admite como grupo gerador o grupo D_4 ou o grupo Z_8. Em ambos os casos, tomando o elemento 000 como identidade, temos que, para $E(g) = 111$, $E(g^{-1}) = 010$, o que prova que o rotulamento não é BGU. Entretanto, a probabilidade de erro de bit independe do sinal transmitido.

Uma questão em aberto é como manipular rotulamentos com UBEP sem recorrer aos rotulamentos BGU, e estender esta propriedade para abranger também constelações com dimensões maiores, possibilitando assim gerar códigos UBEP não BGU. A complexidade de uma procura para um rotulamento de uma constelação multidimensional por um rótulo com muitos bits torna a procura exaustiva proibitiva.

III. Concatenação Serial de Códigos e Limitantes de Probabilidade de Erro

A concatenação serial de códigos consiste na utilização de mais de um código utilizados em cascata com um entrelaçador de comprimento N entre eles que embaralha os bits de saída do código externo C_e antes que estes bits alimentem o código interno C_i, como mostra a figura 2. A taxa do sistema é k/p.

Consideramos a aproximação por um código de bloco equivalente de um CCS que possui um codificador interno

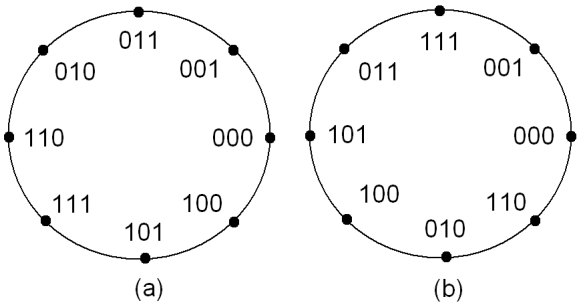

Fig. 1. 8-PSK (a) Rotulamento BGU. (b) Rotulamento com UBEP, não BGU

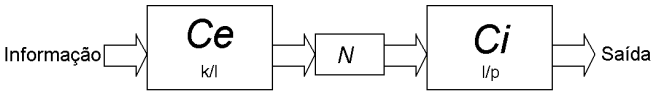

Fig. 2. Esquema de concatenação serial

TCM BGU terminado (zero-tail) . Utilizando a função de distribuição de pesos dos códigos, podemos escrever o limitante de união para $P_b(e)$ como:

$$P_b(e) \leq \sum_{w=1}^{NR_o} \frac{w}{NR_o} \sum_{l} \frac{A^o(w,l)A^i(l,D)}{\begin{pmatrix} N \\ l \end{pmatrix}} \Bigg|_{D=e^{-\frac{R_c E_s}{4N_0}}} \qquad (3)$$

Por utilizarmos como código interno um codificador BGU, não precisamos percorrer todas as seqüencias de saída possíveis, pois utilizamos como referência somente a palavra toda nula.

Nesta equação, temos os seguintes elementos:

- w = Peso de Hamming da palavra de entrada
- R_o =Taxa do código externo
- N = Tamanho em bits do entrelaçador
- l = Peso de Hamming da palavra intermediária
- $A^o(w,l)$ = Número de palavras com peso de entrada w e peso de saída l que o codificador externo gera quando o bloco de entrada tem tamanho R_oN.
- $A^i(l,D)$ = IOWEF (do inglês: Input-Output Weight Enumerating Function) do código interno, para palavras com peso de entrada igual a l
- E_s = energia do sinal, N_0 = densidade unilateral do ruído.

Para os cálculos utilizamos o conceito de entrelaçador uniforme como apresentado em [3]. Devido à sua presença, cada seqüência do código interno tem a probabilidade de $\begin{pmatrix} N \\ l \end{pmatrix}^{-1}$ de ser permutada para uma seqüência específica de entrada para o código interno, pois uma seqüência de entrada com peso de Hamming l é distribuída uniformemente entre todas as possibilidades. A equação 3 pode ser re-escrita da seguinte forma:

$$P_b(e) \leq \sum_{d} e^{-\frac{E_s}{N_0}d} \sum_{w} \frac{w}{k} \sum_{l} \frac{A^o(w,l)A^i(l,d)}{\begin{pmatrix} N \\ l \end{pmatrix}} \qquad (4)$$

onde $A^i(l,d)$ é o número de seqüências com peso de Hamming de entrada l que geram seqüências de saída com peso Euclidiano quadrático igual a d.

Desta forma percebemos que, para grandes N's, palavras que tem grandes l's intermediários não são relevantes para o desempenho do código. Assim, palavras com baixos valores de d e baixos valores de l são as mais importantes. Para altos valores da relação sinal ruído (RSR), a distância mínima do código ainda domina o desempenho, já que a exponencial decai mais devagar quanto menor for o valor de d. Palavras com altos valores de k normalmente geram seqüências com altos valores de l. Palavras com l próximos a N normalmente geram seqüências com altos pesos de saída, cuja influência desaparece rapidamente com um aumento da RSR. A escolha apropriada do código externo influencia no valor do l_{min}, que é a distância livre do código externo. Tanto o código externo como o interno devem ser escolhidos de forma a minimizar a multiplicidade das seqüências, principalmente aquelas com peso de Hamming intermediário baixo. Podemos perceber que o entrelaçador é uma grande causa de ganho, distribuindo bem as seqüências intermediárias com peso de Hamming baixo. Na prática não existem entrelaçadores uniformes, mas é possível que haja alguns entrelaçadores com desempenho igual ou melhor que o entrelaçador ideal.

Assim, vamos seguir as seguintes diretrizes:

- Para seqüências com baixos valores de l, $l > d_f^0$, onde d_f^0 é a distância livre do código externo, aumentar ao máximo a distância Euclidiana de saída, dando prioridade às palavras com menor l. O termo a ser maximizado, caso (d_f^0) seja par é:

$$d_f^o d_{f,ef}^i \tag{5}$$

  e caso (d_f^0) seja ímpar é:

$$\frac{(d_f^o - 3)d_{f,ef}^i}{2} + h_m^{(3)} \tag{6}$$

  onde $h_m^{(3)}$ é peso mínimo das seqüências do código interno geradas por uma entrada com peso de Hamming 3, e $d_{f,ef}^i$ é o peso Euclidiano mínimo das seqüências do código interno geradas por uma entrada com peso de Hamming igual a 2. O que se deseja através disso é aumentar o peso mínimo de um primeiro evento de erro do código interno. Para códigos externos com distância livre ímpar, é muito vantajoso fazer com que $h_m^{(3)} = \infty$, pois assim só há eventos de erro com peso par, tornando mais fácil o projeto do codificador interno.
- Fazer com que seqüências com peso Euclidiano baixo e menor do que a distância livre efetiva tenham o peso de Hamming intermediário o maior possível. Esta condição é ainda mais importante para o caso das palavras que geram seqüências com distância mínima do código interno. Assim, não somente minimizamos o número de seqüências de entrada que geram seqüências de saída com peso mínimo, mas também aproveitamos o ganho do entrelaçador para minimizar o multiplicador do expoente da distância mínima. Desta forma, a distância mínima domina o limitante para valores ainda maiores da RSR.
- Para que haja sempre um ganho do entrelaçador, o código interno deve ser recursivo, i.e, nenhuma seqüência de peso de entrada 1 deve gerar uma seqüência de saída com peso Euclidiano finito.

Uma análise mais detalhada foi feita em [5].

IV. Método para gerar codificadores BGU

Nesta seção vamos mostrar como construir um codificador interno BGU para ser utilizado num sistema de concatenação serial. Como parâmetros de entrada temos a constelação S a ser utilizada, a distância de Hamming livre do código externo e o tamanho em bits da entrada. Utilizaremos a seguinte nomenclatura: S uma constelação, Σ_m o conjunto dos estados de um codificador com m memórias, T um dos grupos que representa uma seção de treliça de um código TCM, composto pelo estado inicial, sinal e estado final, F_0 o subgrupo de T com os ramos que saem do estado identidade, E_0 o mapeamento entre F_0 e o espaço de Hamming correspondente e H_k um grupo de entrada. Não é necessário que o grupo de entrada seja todo o espaço de Hamming de tamanho k.

Utilizaremos uma constelação $2\times 8PSK$ para um código de taxa 5/6. Representaremos os sinais desta constelação através do rótulo dado na figura 3 . Utilizaremos dois codificadores externos, um com d_f^o =2 e outro com d_f^o =3.

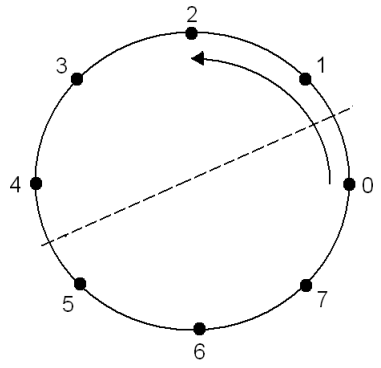

Fig. 3. Constelação $8PSK$ e grupo associado

Seguindo os seguintes passos, obtemos C_i.

1^o *passo:* Escolher quantas memórias o código terá. O espaço de memórias será Σ_m, isomorfo a Z_2^m. ◇

Vamos construir um código com 2 memórias. O espaço de estados será então isomorfo a Z_2^2.

2^o *passo:* Escolher equações de paridade Q que definam o estado final a partir do estado inicial e da palavra de entrada, resultando num mapeamento do par (H_k,Σ_m) em Σ_m. Deve ser preferencialmente par, i.e, somente seqüências de entrada com peso de Hamming par geram seqüências que saem do estado identidade e retornam a ele mesmo. Isto faz com que $h_{min}^{(3)} = \infty$, o que é desejável. As equações devem ser escolhidas de tal forma que minimizem a multiplicidade de caminhos com peso de Hamming de entrada 2 e 4. Isso pode ser feito observando inicialmente as palavras que causam transições paralelas. Minimizar estas multiplicidades também leva a códigos com melhor desempenho. ◇

O mapeamento do par (H_k,Σ_m) em Σ_m é um dos dois fatores que influencia a distância mínima do código, pois define as partições a serem utilizadas. Estas partições podem ser diferentes das partições normalmente utilizadas na construção de bons códigos GU, pois estas não levam em conta as condições de otimização de um sistema com concatenação serial. Por exemplo, em [4], a partição ótima com 4 elementos nos subconjuntos da constelação $2x8PSK$ foi gerada pelo subgrupo: {00,44,04,40}. Se tivéssemos que associar o grupo binário {0000,1100, 1111,1100} a este subgrupo, as palavras com peso 2 teriam peso Euclidiano 4, e a palavra com peso 4 teria peso 8. Se quiséssemos dar o maior peso possível às palavras com peso 2, e ao mesmo tempo ter um rótulo BGU, poderíamos utilizar o subgrupo: {00,24,40,64}. Associando os elementos na mesma ordem em que aparecem, teríamos que as palavras com peso 2 teriam peso 6, e a palavra com peso 4 teria peso 4. Note que, enquanto no primeiro caso o subgrupo é gerado por dois geradores de ordem 2, {44 e 40} (e o subgrupo é isomorfo a Z_2^2), o segundo é gerado por um único gerador de ordem 4 {24} (e o isomorfismo é com Z_4).

Sendo a palavra de entrada do codificador interno representada pelos 5 bits $\{b_1, b_2, b_3, b_4, b_5\}$, escolhemos as seguintes equações:

$$\begin{aligned} p_1 &= b_1 + b_2 + b_3 + b_4 + b_5 \\ p_2 &= b_2 + b_3 + b_4 \end{aligned} \tag{7}$$

Para fazer com que o código seja par, utilizamos as equações 8 abaixo para a definição das transições da treliça:

$$\begin{aligned} E_{1,k} &= p_1 + E_{1,k-1} \\ E_{2,k} &= p_2 + E_{1,k-1} + E_{2,k-1} \end{aligned} \tag{8}$$

onde E_i é o i-ésimo bit da memória, e $k, k-1$ são índices temporais. As palavras que causam uma transição paralela são:

$$\{00000, 10001, 01100, 01010, 00110, 11101, 11011, 10110\}.$$

3^o *passo:* Escolher grupo gerador C de S. Há mais de uma possibilidade. ◇

Para a constelação $2{\times}8PSK$ utilizaremos o grupo D_4^2. Outras possibilidades seriam $D_4 \times Z_8$ e Z_8^2.

4^o *passo:* Escolher um grupo de ligação L, que seja tanto um grupo de simetria de H_k como de um subgrupo $G \subseteq C$. Não faremos mais distinção entre o conjunto de elementos de H_k e o grupo que gera este conjunto. Escolher geradores de H_k cuja relação entre si seja a mesma da relação dos geradores de L. Os geradores de H_k podem ser descritos através de combinações de permutações e somas. Este conjunto será chamado de C_H. Escolher geradores de S que satisfaçam à mesma condição. Este conjunto será chamado de C_G, pois seus elementos geram o subgrupo G.apeamento do par (H_k,Σ_m) em Σ_m ◇

O grupo L faz com que o grupo H_k e C sejam estruturalmente os mesmos, e o homomorfismo existente entre eles seja um automorfismo trivial, com apenas uma troca de nomes. No nosso caso, C será um dos grupos que representa uma seção de treliça, contendo somente os elementos que saem do estado identidade. Futuros isomorfismos de C definirão o subgrupo F_0 da treliça. Dado um mapeamento dos elementos de C_H em C_G, existe no máximo um homomorfismo de H_k em G [6]. Sejam $g_1, g_2(h_1, h_2)$ geradores de $G(H)$. Teremos este isomorfismo se:

$$f : G \to H,\ f(g_1 \cdot g_2) = h_1 \cdot h_2$$

É fácil perceber que o homomorfismo é um rotulamento BGU. Assim sendo, já garantimos esta propriedade. Basta agora procurar entre os possíveis codificadores BGU, aquele que tem as melhores características.

Como o grupo de entrada tem 32 elementos (H_5), o grupo que utilizaremos para representa-lo será o $D(Z_4 \times Z_4)$. Há outras possibilidades, mas escolhemos esta por ter poucos geradores com ordem baixa. As relações entre os geradores deste grupo são:

$$r_1^4 = r_2^4 = s^2 = (r_1 s)^2 = (r_2 s)^2 = (r_1 r_2 s)^2 = e r_1 r_2 = r_2 r_1$$

5^o *passo:* Procurar mapeamentos do conjunto C_H que gerem automorfismos em H_k, dando preferência à geradores com baixo peso de Hamming. Deve haver um mapeamento entre os geradores do espaço de Hamming e elementos do espaço Z_2^m. Isto é:

$$Q(e_i, p) = e_f, \quad Q(e_i, p * g) = g'(e_f) \tag{9}$$

para todo $e_i, e_f \in \Sigma_m$, g gerador de L pertencente a C_H e p uma palavra de entrada. A função $g'()$ é uma operação de soma sobre Z_2^m. Através deste automorfismo obtemos um novo automorfismo $I_H : H_k \to H_k$, que é completamente determinado pelo mapeamento dos geradores. ◇

A tabela VI mostra três possibilidades de se gerar H_5 através das relações entre os geradores definidas anteriormente. Os primeiros 5 termos indicam uma permutação, os últimos 5 termos indicam o que deve ser somado, módulo 2, ao elemento inicial. Parte-se da identidade (00000) para gerar H_5.

TABELA I
GERADORES DE H_5

r_1	r_2	s
(54321)(11000)	(14325)(01100)	(12345)(00111)
(21345)(10000)	(12354)(00001)	(12345)(01110)
(14325)(01100)	(54321)(10000)	(12345)(00011)

A primeira linha fornece um bom controle sobre a posição das palavras que causam uma transição paralela, além de permitir um controle sobre o estado final. De acordo com as equações de paridade, a aplicação de r_1, r_2 e s sobre o espaço de estados leva às equações 10 abaixo, utilizando a representação binária, onde E_i representa o estado inicial:

$$\begin{aligned} r_1(E_i) &= E_i, \\ r_2(E_i) &= E_i \oplus (10), \\ s(E_i) &= E_i \oplus (11), \end{aligned} \tag{10}$$

Assim temos um isomorfismo de H_k ou G em Σ_m.

6^o *passo:* Conhecemos a ordem dos geradores que geram segmentos críticos (i.e., todas a transições paralelas, sendo mais importantes aquelas com peso baixo, ou, ramos que saem do estado identidade com peso 1). Procurar automorfismos

do subgrupo gerado por C_G de modo a fornecer altos pesos Euclidianos para estes ramos críticos. Obtemos assim uma função $I_G : G \rightarrow G$. Este automorfismo é completamente determinado pelo mapeamento dos geradores. Este ponto é extremamente importante na otimização do código.

Como critério para a escolha do automorfismo, vamos tentar maximizar o peso Euclidiano das palavras com peso 2 que causam uma transição paralela. Isto nos fornece um limite superior para $d^i_{f,ef}$. Vamos também fazer com que as palavras com peso de entrada 1 tenham no mínimo a metade deste valor. Vamos também tentar fazer com que os sinais com baixo peso Euclidiano tenham altos pesos de Hamming de entrada, de modo que a influência destes seja minimizada devido ao ganho do entrelaçador. ◇

A tabela II mostra três possibilidades de se gerar subgrupos de 32 elementos de $2 \times 8PSK$ utilizando-se a representação da figura 3 e geradores que respeitam a lei acima, sendo assim isomorfos ao grupo $D(Z_4 \times Z_4)$.

TABELA II

GERADORES DE 32 ELEMENTOS EM $2 \times 8PSK$

r_1	r_2	s
(2 0)	(0 2)	(1 1)
(2 4)	(0 2)	(3 5)
(2 4)	(2 2)	(1 5)

A ultima linha fornece bons valores para palavras com peso 2 que causam transições paralelas e também fornece bons pesos Euclidianos para as palavras com peso 1 e 2 que saem do estado identidade .

7^o *passo:* Associar os grupos até agora definidos da seguinte forma:

$$I_H(H) \leftarrow H_k \leftarrow L \rightarrow G_S \rightarrow I_G(G) \tag{11}$$

Como um homomorfismo de um homomorfismo ainda é um homomorfismo, ainda temos um homomorfismo de H_k em G, o que faz com que o código seja BGU. Através deste mapeamento obtemos E_0. Este mapeamento (que é um isomorfismo) é completamente determinado pelo mapeamento dos geradores. O estado inicial de cada ramo é o estado identidade, e o estado final é obtido através das equações 10. Assim, temos o subgrupo F_0 de T. ◇

A associação final obtida é definida pela tabela III

TABELA III

ASSOCIAÇÃO DE GERADORES

Gerador de H_5	Gerador de G
(54321)(11000)	(2 2)
(14325)(01100)	(2 4)
(12345)(00111)	(1 5)

8^o *passo:* Encontrar candidatos para o grupo A. Dentre todos os elementos do grupo S, escolher aqueles que satisfazem à condição:

$$wh(E_0(f)) = wh(E_0(a \cdot f \cdot a^{-1})) \tag{12}$$

para todo $f \in F$. Estes candidatos formam o conjunto A'. ◇

Para o código que estamos procurando, os candidatos encontrados foram :{00, 03, 04, 07, 30, 33, 34, 37, 40, 43, 44, 47, 70, 73, 74, 77}.

9^o *passo:* Para cada estado distinto da identidade, definir qual a palavra de entrada do codificador interno que causa uma transição que faz com que o codificador retorne ao estado identidade. Estas palavras estão associadas, através de E_0, a sinais. Esta é a partição da constelação que, ao ser multiplicada por um elemento a apropriado, gera os sinais que fazem o codificador retornar ao estado identidade. ◇

10^o *passo:* Para cada estado e para candidato de A', obter o vetor $d_{ij}(\cdot)$, multiplicando a partição obtida no nono passo para o i-ésimo estado pelo j-ésimo elemento de A'. O n-ésimo termo de $d_{ij}(\cdot)$ possui a distância Euclidiana mínima que resulta da multiplicação para o ramo com rótulo cujo peso de Hamming é igual a $n, n \leq k$. ◇

11^0 *passo:* Organizar os vetores $d_{ij}(\cdot)$, em relação a j, para cada estado i. Os melhores são aqueles que resultam no maior valor para a primeira dimensão do vetor. Entre estes os melhores são aqueles que resultam no maior valor para a segunda dimensão, e assim sucessivamente, até a ultima dimensão. Se, para alguma dimensão a distância Euclidiana for zero, o vetor geraria codificadores ruins, e deve ser colocado no fim da classificação. ◇

Os passos anteriores resultam na tabela IV

TABELA IV

CANDIDATOS À FORMAREM O GRUPO A

00	01	10	11
Por definição,	74	70	77
00	70	04	73
	34	07	07
	30	03	03
	47	74	74
	43	34	70
	77	40	04
	73	44	47
	37	30	73
	33	47	37
	03	43	33
	07	00	44
	44	77	40
	40	73	34
	04	37	30
	00	77	00

12^o *passo:* Através da classificação obtida, tentar formar um grupo com o melhor candidato possível para cada estado. Devemos evitar escolhas de candidatos que resultem, na treliça, em auto-laços com peso Euclidiano nulo (com peso de Hamming não nulo), pois isso faria com que palavras com alto peso de Hamming teriam baixo peso Euclidiano de saída. Também devemos escolher os candidatos de modo a utilizar todos os sinais da constelação. A escolha dos candidatos deve ser de tal forma que eles formem um grupo isomorfo ao grupo de estados Σ_m.

Não é útil fazer com que um ramo com peso de entrada 1 tenha peso Euclidiano muito maior do que $d^i_{f,ef}$, pois isso não resultaria em nenhum grande ganho. Os menores pesos Euclidianos devem ficar com sinais com os maiores pesos de Hamming. Devemos também tentar tornar a distância mínima

do código alta, pois para altos valores da RSR, é esta distância que determina o desempenho do código. ◇

Como não conseguimos montar o grupo desejado com o melhor para cada estado, escolhemos sacrificar um pouco das distâncias das sequencias que saem do estado 11. Assim, chegamos à seguinte associação:

TABELA V

GRUPO A FINAL.

Estado	Elemento correspondente
00	00
01	74
10	70
11	04

V. RESULTADOS

A figura 4 apresenta os limitantes de união para o código encontrado aqui (código A) e para o código encontrado em [1] (código B). A linha com pontos indica o desempenho do código A e a linha cheia representa o desempenho para o código B. As linhas mais grossas são para o sistema com d_f^0 igual a 2, e as linhas mais finas são para o caso com d_f^0 igual a 3. Comparado com o desempenho do código B, cujo código interno tinha apenas uma memória, podemos esperar uma melhora significativa no desempenho.

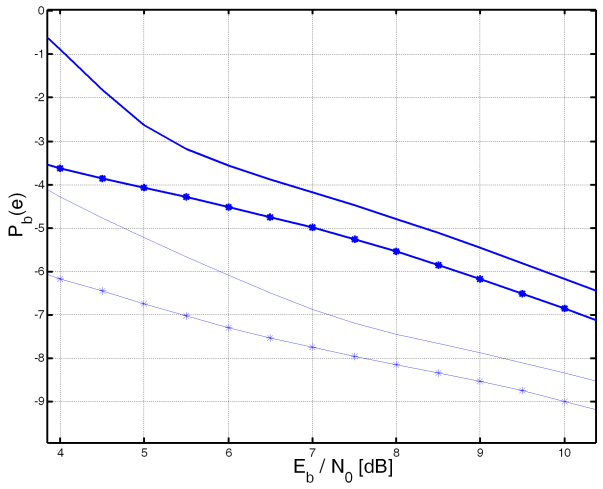

Fig. 4. Desempenho de códigos

O código encontrado, embora não possa ser representado através de somadores e registradores de deslocamento, pode ser descrito através da tabela VI. O estado final de cada ramo pode ser obtido utilizando as equações 7 e 8.

VI. CONCLUSÃO

Embora não seja uma solução ótima para o problema, o método apresentado aqui fornece bons resultados. Uma única busca exaustiva foi substituída por três buscas guiadas, simples o suficiente para poderem ser feitas a mão. O código encontrado foi utilizado num esquema de concatenação serial obtendo resultados melhores do que os anteriores, com complexidade não muito maior.

TABELA VI

CÓDIGO DE 2 MEMÓRIAS, 5/6, MAPEADO EM 2X8PSK.

Palavra de entrada	Estado inicial			
	00	01	11	10
00000	00	74	70	04
01100	24	50	54	20
01010	40	34	30	44
00110	64	10	14	60
11000	22	56	52	26
11011	44	30	34	40
00011	66	12	16	62
11110	46	32	36	42
10111	60	14	10	64
00101	02	76	72	06
10010	62	16	12	66
10001	04	70	74	00
01001	26	52	56	22
10100	06	72	76	02
11101	20	54	50	24
01111	42	36	32	46
00111	15	61	65	11
01011	71	05	01	75
01101	55	21	25	51
00001	31	45	41	35
11111	73	07	03	77
11100	51	25	21	55
00100	37	43	47	33
11001	57	23	27	53
10000	35	41	45	31
00010	13	67	63	17
10101	33	47	43	37
10110	11	65	61	15
01110	77	03	07	73
10011	17	63	67	13
11010	75	01	05	71
01000	53	27	23	57

REFERÊNCIAS

[1] R. Garello , G. Montorsi , S. Benedetto , D. Divsalar e F. Pollara, *Labelings and Encoders With the Uniform Bit Error Property With Applications to Serially Concatenated Trellis Codes*,IEEE Transactions on Information Theory, v. 48, n. 1, p. 123-136, Janeiro de 2002.

[2] A. G. i Amat, G. Montorsi, S. Benedetto, *Design and Decoding of Optimal High-Rate Convolutional Codes*, IEEE Transactions on Information Theory, v. 50, n. 5, p. 867-881, Maio de 2004.

[3] S. Benedetto, G. Montorsi, *Unveiling Turbo Codes: Some Results on Paralle Concatenated Coding Schemes*, IEEE Transactions on Information Theory, v. 42, n. 2, p. 409-428, Março de 1996..

[4] S. Benedetto, R. Garello, M. Mondin, G. Montorsi, *Geometrically Uniform Partitions of* L *x* MPSK *Constellations and Related Binary Trellis Codes*, IEEE Transactions on Information Theory, v. 39, n. 6, p. 1773-1798, Novembro de 1993.

[5] S. Benedetto, D. Divsalar, G. Montorsi, F. Pollara, *Serial Concatenation of Interleaved Codes: Performance Analysis, Design, and Iterative Decoding*, IEEE Transactions on Information Theory, v. 44, n. 3, p. 909-926, Maio de 1998.

[6] S. Lang, *Algebra*,Addison-Wesley,1965.