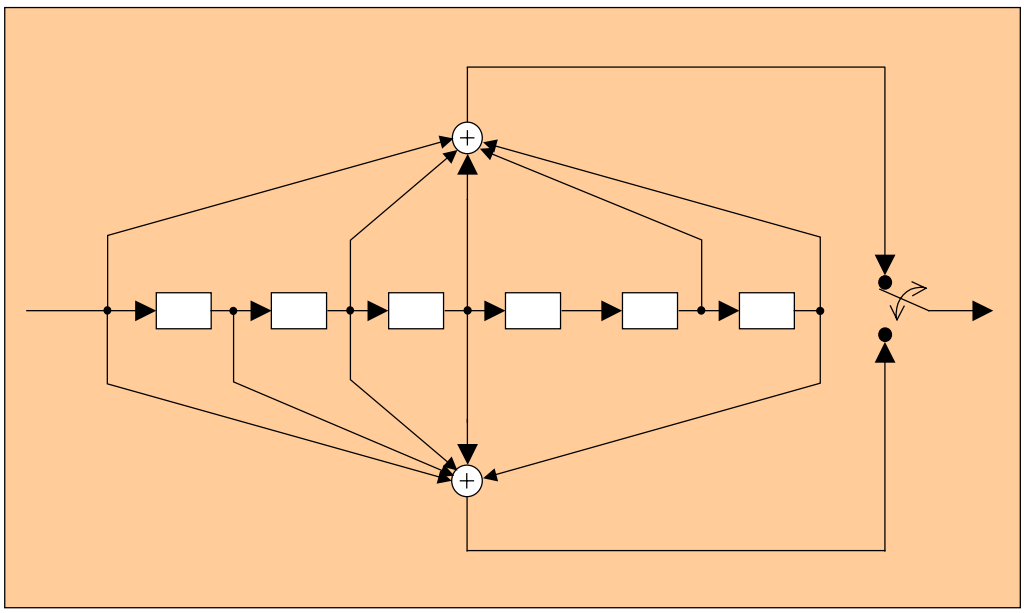
2.5. Códigos convolucionais

- Métodos de representação gráfica e não gráfica
- Distância livre e limite de Heller
- Função de transferência
- Codificadores catastróficos
- Ganho de codificação
- Códigos perfurados
- Métodos de descodificação:
  - algoritmo de Viterbi (máxima verosimilhança)
  - descodificação sequencial
  - descodificação com *feedback*

Códigos convolucionais

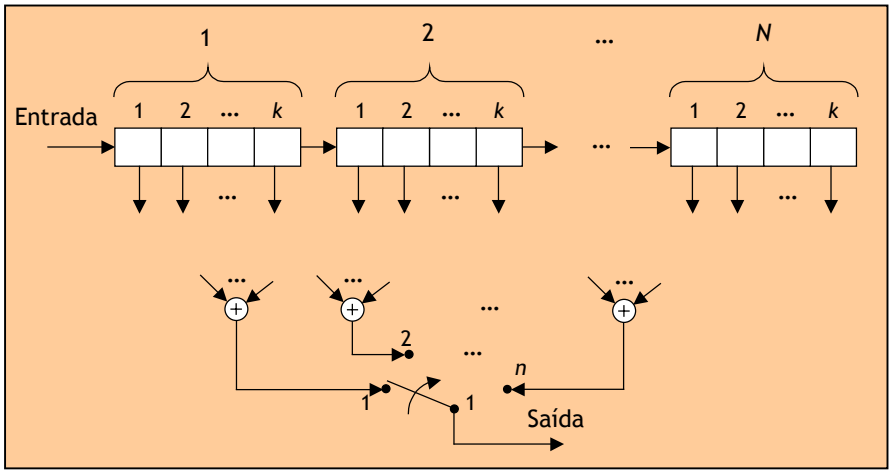
- O processamento não se faz em bloco com palavras de código, como nos códigos algébricos (de blocos).
- São códigos em árvore, representáveis por diagramas de estados.
- Usam-se registos de deslocamento na codificação.
- O "hardware" de codificação convolucional é *mais simples* que o "hardware" de codificação por blocos (por exemplo, não é preciso "buffer" de entrada).
- A estrutura convolucional é adequada a comunicações espaciais e por satélite:
  - requer codificadores simples (no "espaço");
  - a elevada "performance" do código obtém-se com métodos de descodificação sofisticados (em "Terra").

Um exemplo: o código de Odenwalder (NASA)

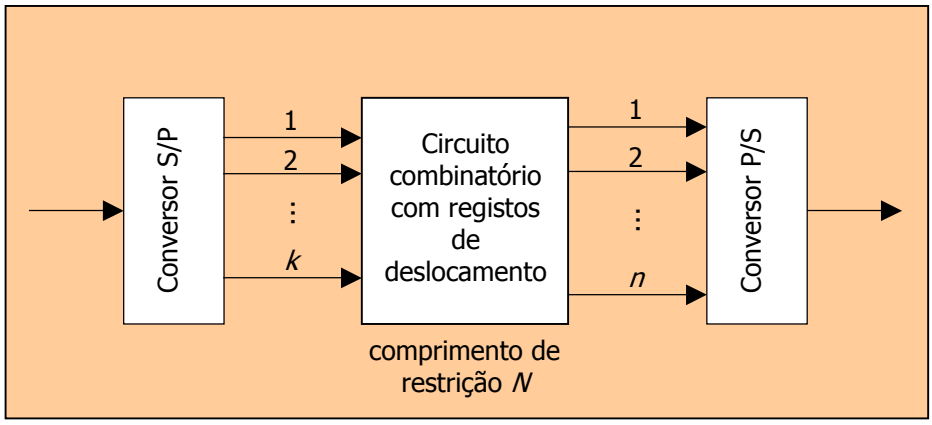


Codificadores convolucionais genéricos

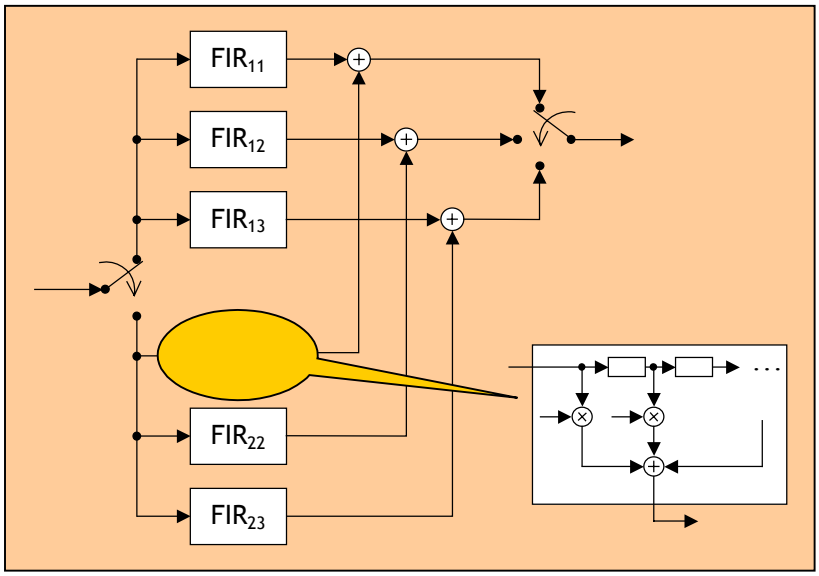
Diagrama de blocos genérico de um codificador convolucional (*n*, *k*) com comprimento de restrição *N* e entrada em série



Esquema genérico de codificador convolucional com entradas em paralelo

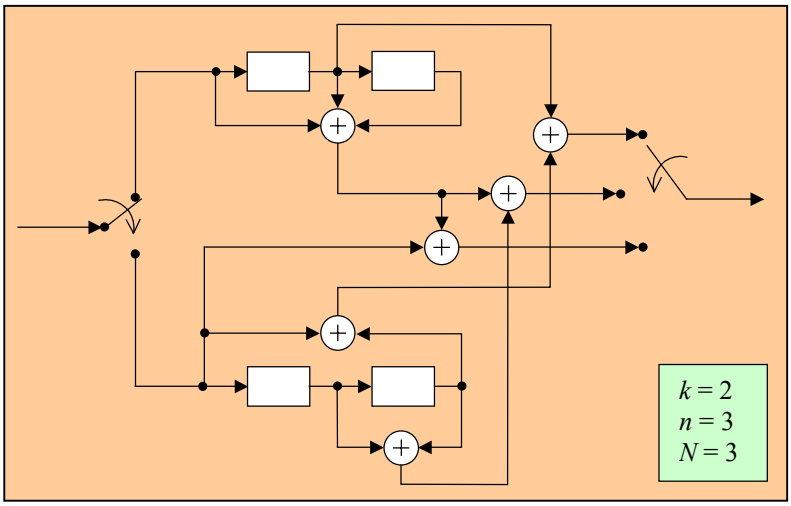


Esquema genérico de codificador convolucional com filtros FIR

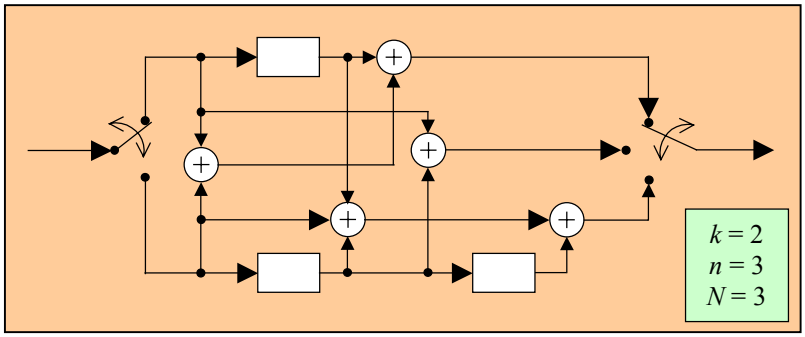


Exemplos de codificadores convolucionais (*n*, *k*, *N*)

Codificador convolucional (3,2) com comprimento de restrição 3



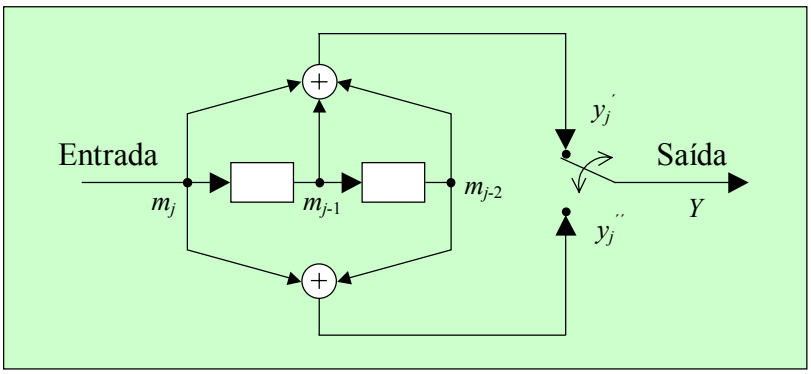
Outro codificador sistemático (3,2) com comprimento de restrição 3



- Taxa do código: $R_c = k/n$.
- Comprimento de restrição: $N = \max_{1 \le i \le k} N_i$.
- Memória do codificador: $\nu = \sum_{i=1}^{k} (N_i - 1)$ ($\nu = k(N-1)$ se N_i iguais)
- Número de estados: 2^{ν} ($2^{\nu} = 2^{k(N-1)}$ se N_i iguais)

Comprimento de restrição do código: número de deslocamentos de um bit de entrada que influenciam bits de saída.

Exemplo de um codificador convolucional (2, 1, 3)



$$\begin{cases} n = 2 \text{ bits de saída} \\ k = 1 \text{ bit de entrada em cada deslocamento} \\ \nu = N - 1 = 2 \text{ bits de estado} \end{cases}$$

- Este codificador gera $n = 2$ bits, y'_j e y''_j, por cada bit de entrada:

$$y'_j = m_j \oplus m_{j-1} \oplus m_{j-2} \qquad y''_j = m_j \oplus m_{j-2}$$

A sequência binária de saída é $Y = y'_1\ y''_1\ y'_2\ y''_2\ y'_3\ y''_3\ y'_4\ y''_4 \ldots$

- Memória do Codificador: $\nu = N - 1 = 2$
- Existem $2^{k(N-1)} = 4$ estados diferentes: 00, 01, 10 e 11.
- Saídas e Estados

Supondo que inicialmente o registo está limpo ($m_0 m_{-1} = 00$):

$$\Rightarrow \quad \begin{cases} \text{entrada } m_1 = 0 \Rightarrow \text{saída } y'_1 y''_1 = 00 \\ \text{entrada } m_1 = 1 \Rightarrow \text{saída } y'_1 y''_1 = 11 \end{cases}$$

- Taxa do Código: $R_c = k/n = 1/2$
- Um bit de entrada influencia $nN = 6$ bits sucessivos de saída.

Normalmente n e k são pequenos e o comprimento de restrição toma valores inferiores a 10.

Codificadores convolucionais

- *Métodos de representação gráfica*:
  - Árvore do código
  - Treliça ("trellis") do código
  - Diagrama de estados

  ] Métodos "condensados"

- *Métodos de representação não gráfica*:
  - vectores de ligação (ou vectores geradores)
  - polinómios de ligação (ou polinómios geradores)
  - matriz geradora
  - resposta impulsional

- *Métodos de descodificação*:
  - Máxima verosimilhança (Algoritmo de Viterbi)
  - Descodificação sequencial (Wozencraft-Fano)

  ] Métodos algorítmicos ("software")

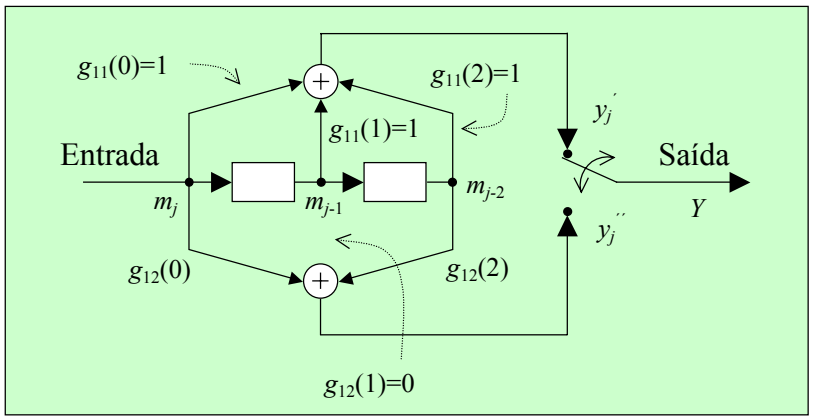
  - Descodificação com "feedback"

— O *algoritmo de Viterbi* requer "hardware" complexo para cálculo e armazenamento de informação.

— A *descodificação sequencial* é de complexidade média. A "performance" é próxima do método anterior em certos casos.

— A *descodificação com "feedback"* é o método mais simples, mas de todos o menos fiável.

Exemplos de representações não gráficas



$$\begin{cases} n = 2 \\ k = 1 \\ N = 3 \end{cases}$$

• Vectores de ligação

Ligação superior: g_1 = 1 1 1
Ligação inferior: g_2 = 1 0 1

• Resposta impulsional (resposta do codificador a um único bit "1")

Conteúdo do registo	Palavra y'_j	Palavra y''_j
1 0 0	1	1
0 1 0	1	0
0 0 1	1	1

Sequência de entrada: 1 0 0
Sequência de saída: 11 10 11 ← *Resposta impulsional do codificador*

Seja a sequência de entrada *m* = 1 0 1. A saída pode ser determinada pela *sobreposição* ou *adição linear* de *"impulsos"* de entrada deslocados no tempo (isto é, é a *convolução* da sequência de entrada com a resposta impulsional).

Entrada	Saída				
1	11	10	11		
0		00	00	00	
1			11	10	11
Soma (mód. 2)	11	10	00	10	11

⇒ Os códigos convolucionais são *lineares*.

Exemplos de representações não gráficas

• Polinómios de ligação

O codificador pode ser representado por n polinómios geradores binários de grau N-1 ou menor.

No exemplo apresentado: $g_1(D) = 1 + D + D^2$ (polinómio superior)

$g_2(D) = 1 + D^2$ (polinómio inferior)

(o termo de ordem mais baixa corresponde ao andar de entrada)

- D é um operador de atraso unitário.
- A sequência de saída, $Y(D)$, vem dada por $Y(D) = m(D)g_1(D)$ entrelaçado com $m(D)g_2(D)$.

Exemplo: Se a sequência de entrada for m = 101 então $m(D) = 1 + D^2$:

$$m(D)g_1(D) = (1+D^2)(1+D+D^2) = 1+D+D^3+D^4$$
$$m(D)g_2(D) = (1+D^2)(1+D^2) = 1+D^4$$

$$\Downarrow$$

$$m(D)g_1(D) = 1+D+0\,D^2+D^3+D^4$$
$$m(D)g_2(D) = 1+0\,D+0\,D^2+0\,D^3+D^4$$

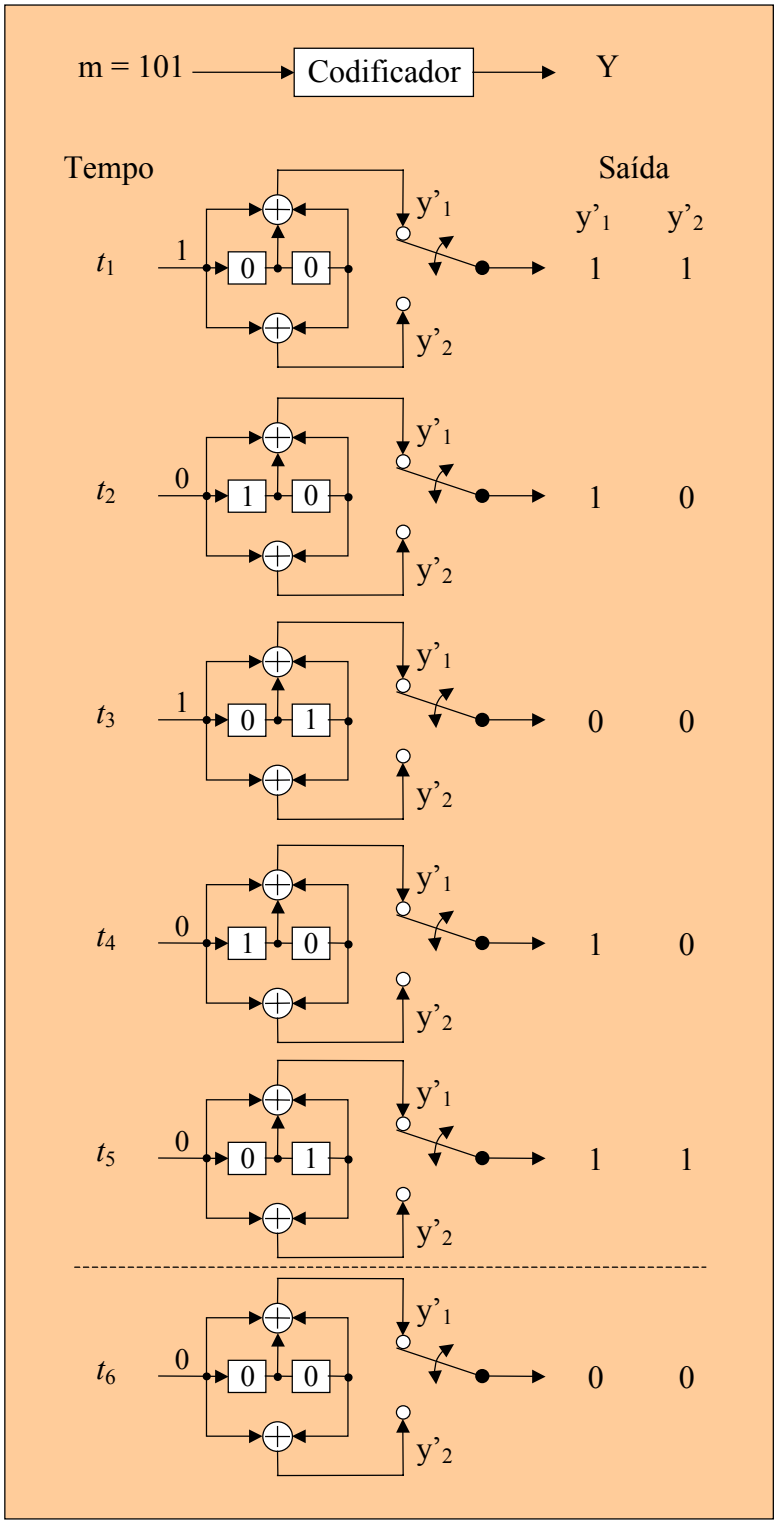
$$\Downarrow$$

$$Y(D) = (1,1)+(1,0)D+(0,0)D^2+(1,0)D^3+(1,1)D^4$$

$$\Downarrow$$

$$Y = 11\ \ 10\ \ 00\ \ 10\ \ 11$$

Exemplo de funcionamento



Sequência de saída: $Y = 11 \quad 10 \quad 00 \quad 10 \quad 11$

Codificadores convolucionais: representações não gráficas

Os vectores ou os polinómios geradores dão origem à *matriz geradora*, de dimensões $k \times n$:

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}_{11} & \mathbf{g}_{12} & \cdots & \mathbf{g}_{1,n} \\ \vdots & & \ddots & \\ \mathbf{g}_{k,1} & \mathbf{g}_{k,2} & \cdots & \mathbf{g}_{k,n} \end{bmatrix}$$

ou

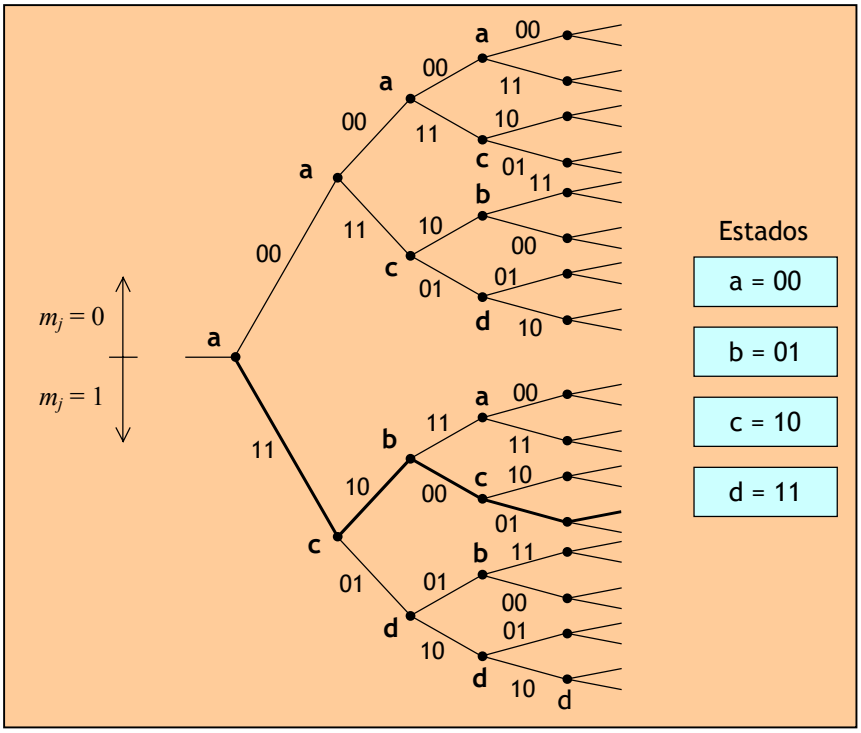
$$\mathbf{G}(D) = \begin{bmatrix} g_{11}(D) & g_{12}(D) & \cdots & g_{1,n}(D) \\ \vdots & & \ddots & \\ g_{k,1}(D) & g_{k,2}(D) & \cdots & g_{k,n}(D) \end{bmatrix}$$

Resumo de representações da matriz geradora

- vectores binários: $\mathbf{G} = \begin{bmatrix} 010 & 111 & 111 \\ 101 & 011 & 100 \end{bmatrix}$
- vectores em octal: $\mathbf{G} = \begin{bmatrix} 2 & 7 & 7 \\ 5 & 3 & 4 \end{bmatrix}$
- polinómios binários: $\mathbf{G}(D) = \begin{bmatrix} D & 1+D+D^2 & 1+D+D^2 \\ 1+D^2 & D+D^2 & 1 \end{bmatrix}$

Representações gráficas dos códigos convolucionais: a árvore

Árvore do código (2, 1, 3) apresentado antes:



- Há 2^j ramos possíveis para o j-ésimo bit de mensagem.
- O padrão de ramos começa a repetir-se em j=3 porque é $N = 3$.

Como há repetição poderemos usar duas outras formas de representação: a *treliça* e o *diagrama de estados*.

Representações gráficas dos códigos convolucionais: a treliça e o diagrama de estados

Treliça

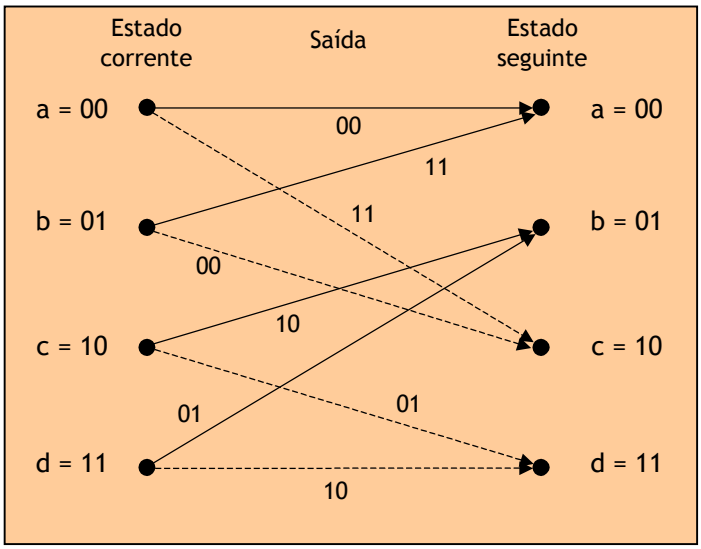
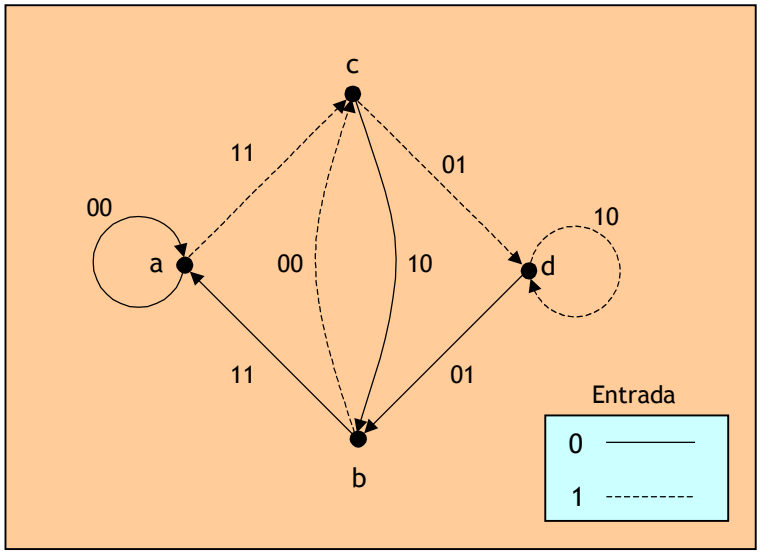


Diagrama de estados



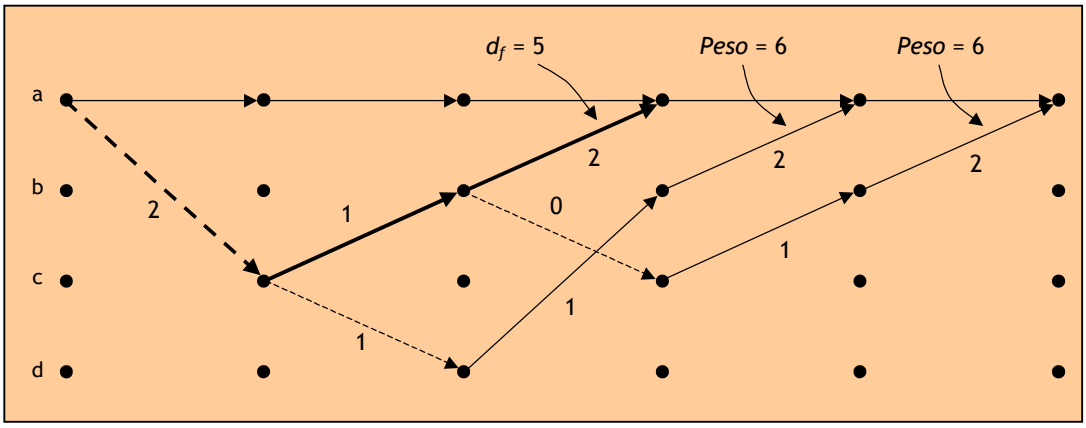
Distância livre de um código convolucional

Nos códigos de blocos a capacidade de controlo de erros depende da *distância mínima* do código, determinada a partir dos *pesos* das palavras de código. Como num código convolucional os bits não são agrupados em palavras considera-se o peso $w(Y)$ de toda uma sequência transmitida Y gerada por uma certa sequência-mensagem. Por definição temos então:

Distância livre de um código convolucional: $d_f \triangleq \left[w(Y) \right]_{\min} \qquad Y \neq 000...$

Claro que d_f não vai ser determinado consultando todas as mensagens possíveis: o que se faz é considerar mensagens que terminem no mesmo estado inicial (00, no exemplo seguido) e que tenham os pesos mais baixos (com poucos "uns"):

Consultando a treliça do código (2, 1, 3) vê-se que o estado *a* é atingido se o estado anterior for *a* ou *b*. Eis três possibilidades (em cada ramo está indicado o respectivo peso):

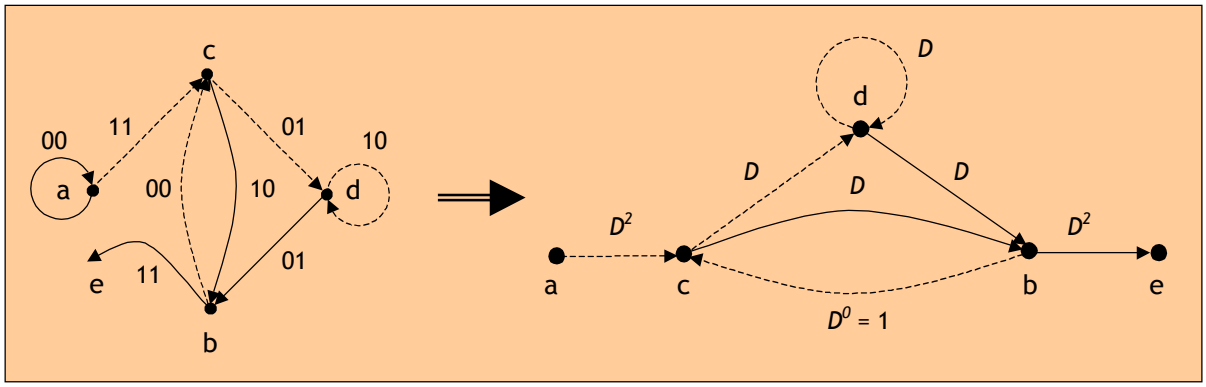


O percurso de menor peso que começa e termina em *a* é **aa ... acbaa** $\Rightarrow d_f = 5$.:

Função de transferência de códigos convolucionais

Uma forma de calcular analiticamente a distância livre é usar a chamada *função de transferência* do código convolucional. Esta função fornece ainda outras indicações úteis.

Tomemos como exemplo o código (2, 1, 3) já apresentado. No seu diagrama de estados vamos associar a cada ramo um rótulo D^i, em que o expoente i representa a distância de Hamming da palavra desse ramo em relação ao ramo só com zeros. Eliminemos a anel do nó *a* porque, tendo peso nulo, em nada contribui para as propriedades de distância das palavras de código em relação ao percurso só com zeros. Além disso, o nó *a* pode ser dividido em dois: um, chamado *a*, que representa a entrada do diagrama de estados e outro, chamado *e*, que representa a saída.



A função de transferência do percurso *acbe* é $D^2DD^2 = D^5$ (o expoente de D representa, portanto, o peso acumulado do percurso).

Como calcular a função de transferência

Podemos definir *equações de estado* e a partir delas calcular a função de transferência do código, $T(D)$:

$$\begin{cases} X_b = DX_c + DX_d \\ X_c = D^2 X_a + X_b \\ X_d = DX_c + DX_d \\ X_e = D^2 X_b \end{cases}$$

$$\Downarrow$$

Função de transferência do codificador

$$T(D) = \frac{X_e}{X_a} = \frac{D^5}{1-2D} = D^5 + 2D^6 + 4D^7 + \ldots + 2^i D^{i+5} + \ldots =$$

$$= \sum_{d=d_f}^{\infty} A_d D^d$$

há um percurso de peso 5

há 2 percursos de peso 6

há 2^i percursos de peso $i + 5$

$$\Downarrow$$

$d_f = 5$

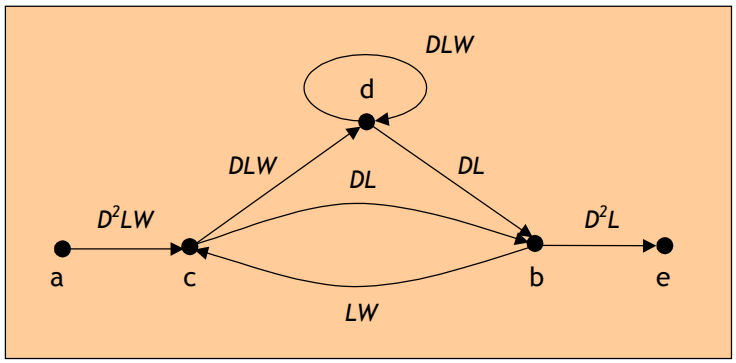
- Atendeu-se a que, através do binómio de Newton, $\frac{1}{1+x} = 1 - x + x^2 - x^3 + \ldots$
- A distância livre deste código é 5 porque o percursos de *a* a *e* com menor peso tem peso 5.
- A função de transferência também pode ser calculada recorrendo à *regra de Mason* (ver Anexo 1). Os interessados poderão consultar o livro de Lin & Costello referido na bibliografia.

Função de transferência

Vamos fazer mais algumas modificações no diagrama de estados:

1 - Introduzir um factor L em cada ramo de maneira que o expoente de L sirva de "contador" para indicar o número de ramos de um dado percurso desde *a* a *e*.

2 - Introduzir um factor W em todos os ramos que resultem de uma transição de estado devido a um bit de entrada "1" (ramo tracejado – – –)



Assim, temos:

$$\begin{cases} X_b = DLX_c + DLX_d \\ X_c = D^2LWX_a + LWX_b \\ X_d = DLWX_c + DLWX_d \\ X_e = D^2LX_b \end{cases}$$

$$T(D,L,W) = \frac{D^5L^3W}{1 - DL(1+L)W} =$$

$$= D^5L^3W + \underbrace{D^6L^4(1+L)W^2}_{D^6L^4W^2 + D^6L^5W^2} + D^7L^5(1+L)^2W^3 + \ldots + D^{l+5}L^{l+3}(1+L)^lW^{l+1} + \ldots$$

Significa que há um único percurso de peso 5, o qual tem comprimento 3 e foi originado por uma sequência de entrada com apenas um bit 1.

Há dois percursos de peso 6, um com comprimento 4 e outro com comprimento 5. As sequências de entrada que lhes deram origem têm ambas peso 2.

Codificadores convolucionais óptimos com comprimento de restrição pequeno (taxas 1/2 e 1/3)

Codificadores de taxa ½ e máxima distância livre

Comprimento de restrição, N	Polinómios geradores		d_f	$d_{f_{\max}}$
2	1	3	3	4
3	5	7	5	5
4	15	17	6	6
5	23	35	7	8
6	53	75	8	9
7	133	171	10	10
8	247	371	10	11
9	561	753	12	12
10	1131	1537	12	13

Codificadores de taxa 1/3 e máxima distância livre

Comprimento de restrição, N	Polinómios geradores			d_f	$d_{f_{\max}}$
2	1	3	3	5	6
3	5	7	7	8	8
4	13	15	17	10	10
5	25	33	37	12	12
6	47	53	75	13	13
7	117	127	155	15	15
8	225	331	367	16	17
9	575	623	727	18	18
10	1167	1375	1545	20	20

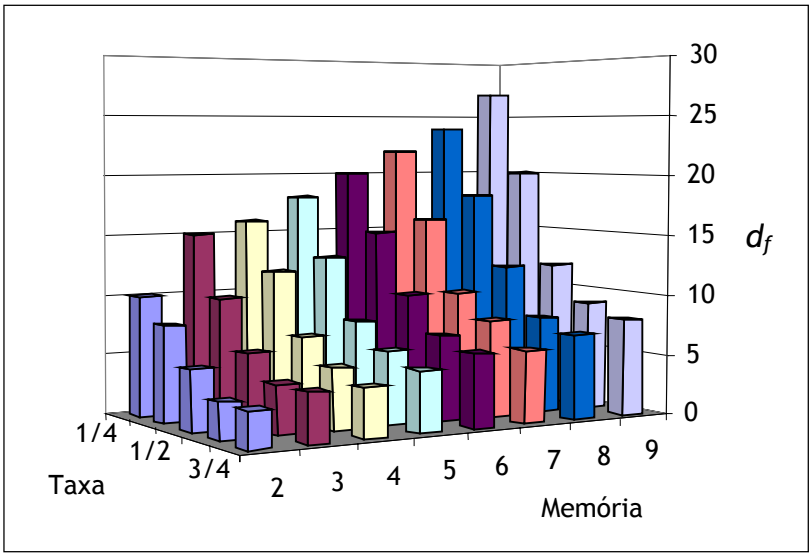
Limite, ou majorante, de Heller: $$d_f \le \min_{l \ge 1} \left\lfloor \frac{2^{l-1}}{2^l - 1}(l + N - 1)n \right\rfloor$$

Distâncias livres dos melhores codificadores convolucionais

Distâncias livres dos melhores codificadores

Memória,	Taxas					
ν	1/5	1/4	1/3	1/2	2/3	3/4
1	–	6	5	3	–	–
2	13	10	8	5	3	3
3	16	15	10	6	4	4
4	20	16	12	7	5	4
5	22	18	13	8	6	5
6	25	20	15	10	7	6
7	28	22	16	10	8	6
8	–	24	18	12	8	7
9	–	27	20	12	9	8
10	–	29	22	14	10	–

Evolução da distância livre em função da taxa de código e da memória de alguns codificadores representativos (taxas 1/4, 1/3, 1/2, 2/3 e 3/4).

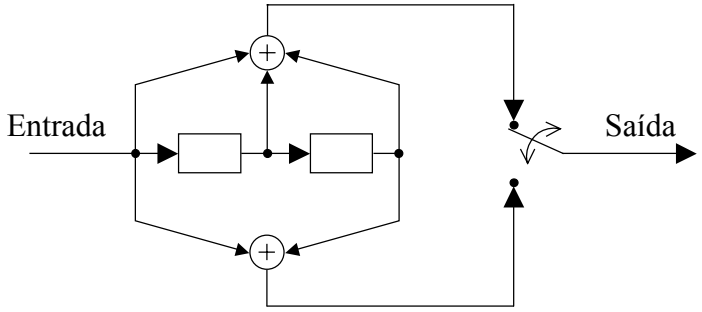


Codificadores recursivos sistemáticos

Codificadores sistemáticos têm menores distâncias livres que codificadores não sistemáticos (para os mesmos parâmetros)…

a não ser que… sejam **recursivos** (RSC)!!

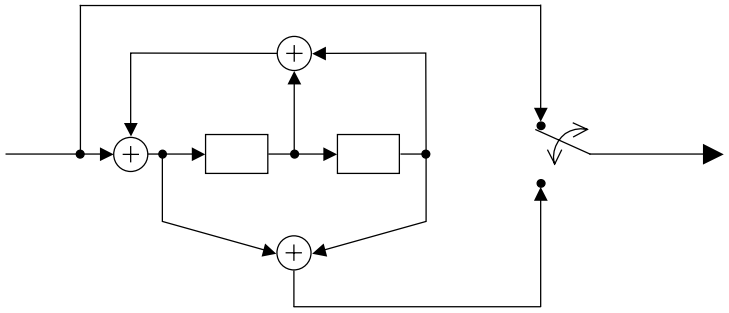
Como determinar o codificador sistemático recursivo equivalente a um codificador não sistemático e não recursivo? Vejamos com um exemplo:



$$\mathbf{G}(D) = \left[g_{11}(D) \quad g_{12}(D) \right] = \left[1 + D + D^2 \quad 1 + D^2 \right]$$

$$\Downarrow$$

$$\tilde{\mathbf{G}}(D) = \left[1 \quad g_{12}(D)/g_{11}(D) \right] = \left[1 \quad \frac{1 + D^2}{1 + D + D^2} \right]$$



Ambos os codificadores têm a mesma distância livre, $d_f = 5$.

Nota: codificador equivalente = codificador que gera o mesmo código

Codificadores recursivos sistemáticos

Outro exemplo (taxa 1/3, $N = 4$)

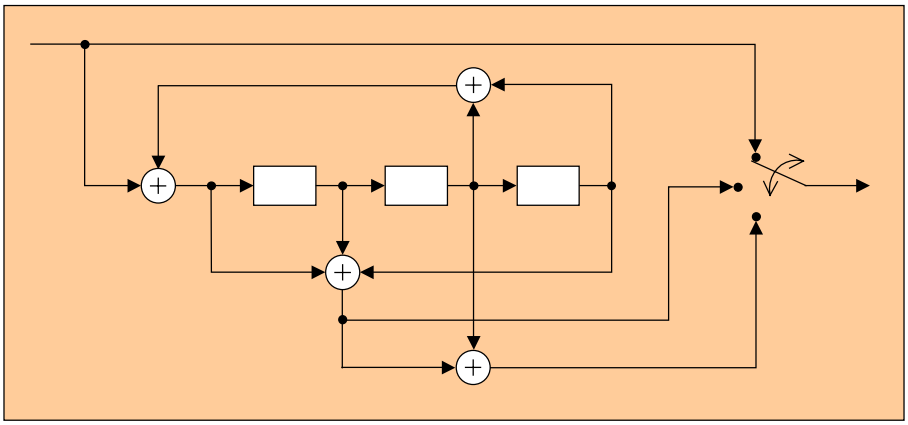
Um codificador convolucional é caracterizado pela seguinte matriz geradora:

$$G(D) = \left[1 + D^2 + D^3 \quad 1 + D + D^3 \quad 1 + D + D^2 + D^3 \right]$$

Qual é o codificador RSC equivalente?

$$\tilde{G}(D) = \left[1 \quad \frac{1 + D + D^3}{1 + D^2 + D^3} \quad \frac{1 + D + D^2 + D^3}{1 + D^2 + D^3} \right]$$

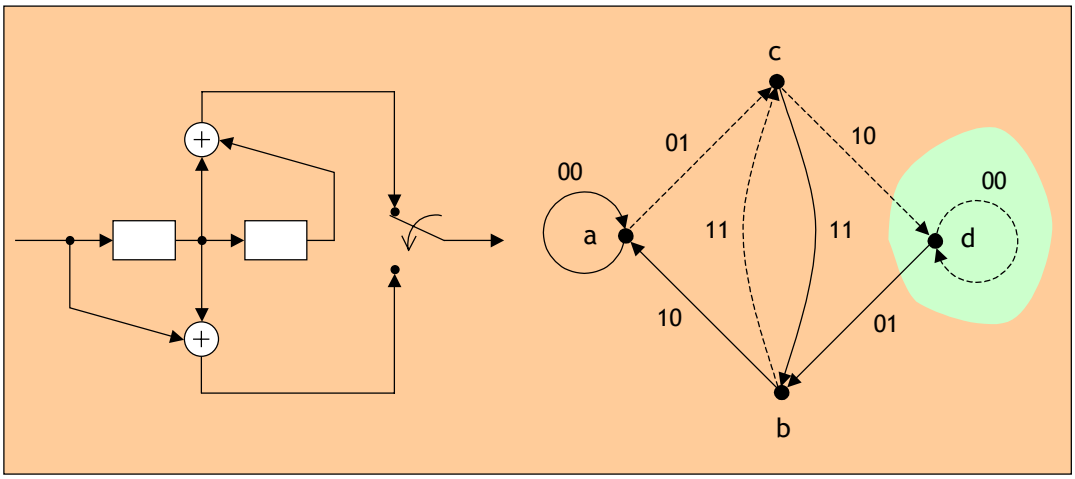
$$\Downarrow$$



Ambos os codificadores geram o mesmo código $\Rightarrow$ têm a mesma distância livre.

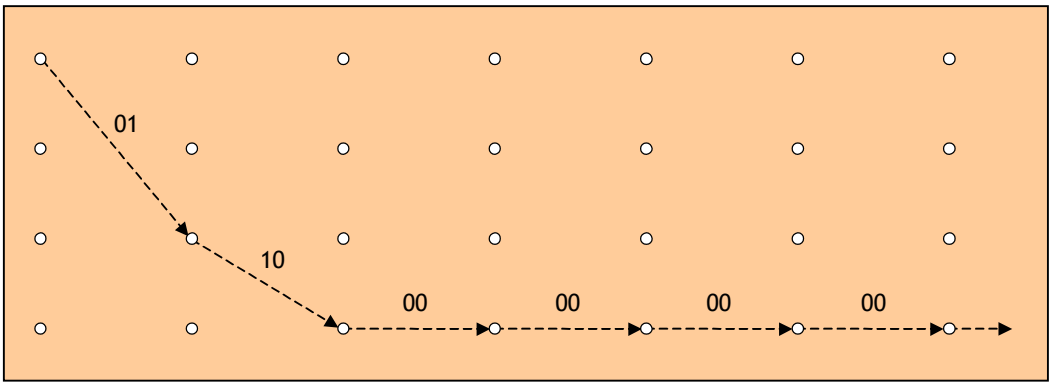
Codificadores catastróficos

Exemplo de codificador convolucional catastrófico:



Se mensagem: 111…11 ⇒ Sequência transmitida: 0110000…00….

Percurso correspondente:



Mas se for:

Mensagem: 0 0 0 0 …

Sequência transmitida: 00 00 00 …

Sequência recebida: 01 10 00 … ⇒ O percurso é o mesmo!!

A mensagem descodificada é 11…1, bem diferente da original! Dois erros no canal provocaram um número incontável de erros de descodificação!!

É uma situação… catastrófica! A evitar absolutamente!

Codificadores catastróficos: como evitá-los?

Condição necessária e suficiente:

Para que um codificador de taxa 1/n não seja catastrófico é necessário e suficiente que o máximo divisor comum dos polinómios geradores seja igual a D^m, para um certo m inteiro não-negativo:

$$m.d.c.\left[g_{11}(D), g_{12}(D), \ldots, g_{1n}(D)\right] = D^m, \quad m \geq 0$$

Os polinómios geradores não deverão, portanto, ter factores comuns que não sejam D^m. Como m pode ser nulo, basta que os polinómios sejam primos entre si,

$$m.d.c.\left[g_{11}(D), g_{12}(D), \ldots, g_{1n}(D)\right] = 1,$$

– ou seja, não tenham nenhuns factores comuns – para que o codificador 1/n não seja catastrófico.

Estas condições reflectem-se no diagrama de estados: **este não deve conter um anel de peso zero em estado não-nulo nem conter um percurso de peso zero saindo de um estado e regressando ao mesmo.**

P.: Porque é que o codificador anterior é catastrófico?

R.: Por causa do factor comum, $1 + D$, que existe nos polinómios geradores $g_{11}(D) = D + D^2$ e $g_{12}(D) = 1 + D$ ou, se quisermos, por causa do anel de peso zero que existe no estado d do diagrama de estados (anel este obrigatoriamente gerado por um bit de entrada 1).

Ganho de codificação

Para obtermos uma determinada BER num sistema de comunicações sem codificação necessitamos de uma determinada relação E_b/N_0. Com codificação adequada necessitaremos de uma menor relação E_b/N_0. A diferença, em dB, entre esses dois valores representa o *ganho de codificação* do código. O ganho de codificação máximo (*ganho de codificação assimptótico*) atinge-se quando a relação E_b/N_0 é muito elevada.

- *Decisões rígidas:* $\text{Ganho de codificação} \le 10\log\frac{R_c d_f}{2}$
- *Decisões brandas não-quantizadas:* $\text{Ganho de codificação} \le 10\log(R_c d_f)$

Ganho de codificação assimptótico, G_a, com decisões rígidas

Comprimento de restrição, N	*Taxa 1/2*		*Taxa 1/3*	
	Distância livre	Ganho de codificação máximo (dB)	Distância livre	Ganho de codificação máximo (dB)
3	5	0,97	8	1,25
4	6	1,76	10	2,22
5	7	2,43	12	3,01
6	8	3,01	13	3,36
7	10	3,98	15	3,98
8	10	3,98	16	4,26
9	12	4,77	18	4,77
10	12	4,77	20	5,23

Com decisões brandas o ganho de codificação é cerca de 3 dB mais elevado.

Códigos perfurados

Imaginemos que temos um código de taxa 1/2 e que, em cada dois pares de bits de saída, eliminamos (isto é, não transmitimos) o último bit. Deste modo dois bits de entrada vão dar origem não a quatro bits de saída, como seria normal, mas apenas a três tendo o código resultante – *código perfurado* – uma taxa de 2/3.

Os códigos perfurados são muito usados dada a sua flexibilidade pois:

- torna-se possível não só obter códigos com diversas taxas de código a partir de um único codificador-pai mas também fazer a descodificação de todos eles com o mesmo descodificador;
- a descodificação torna-se mais simples relativamente aos códigos não-perfurados com a mesma taxa.
- É possível alterar a taxa do código durante a transmissão conforme as características do canal: enquanto houver "pouco" ruído usa-se uma taxa elevada (5/6, por exemplo) mas se o canal se tornar mais ruidoso muda-se para uma taxa mais baixa (2/3 ou ½, por exemplo), pois tem uma distância livre maior.
- Em qualquer dessas situações o descodificador é sempre o mesmo, o que é uma vantagem.
- Em contrapartida as propriedades de distância são piores e há necessidade de sincronismo de trama, pois passa a ser preciso saber onde começa e acaba cada grupo de bits de saída onde se vai fazer a perfuração.

Códigos perfurados

Exemplo com os códigos perfurados das normas DVB-S e DVB-T

- Código-pai: taxa ½, comprimento de restrição 7, polinómios geradores 171 e 133.

Taxas do código	Padrão de perfuração	Sequência transmitida (após conversão paralelo-série)	Distância livre
1/2	y':1 y'':1	$y_1' y_1''$ (1-2)	10
2/3	y':10 y'':11	$y_1' y_1'' y_2''$ (1-2-X-4)	6
3/4	y':101 y'':110	$y_1' y_1'' y_2'' y_3'$ (1-2-X-4-5-X)	5
5/6	y':10101 y'':11010	$y_1' y_1'' y_2'' y_3' y_4'' y_5'$ (1-2-X-4-5-X-X-8-9-X)	4
7/8	y':1000101 y'':1111010	$y_1' y_1'' y_2'' y_3'' y_4'' y_5' y_6'' y_7'$ (1-2-X-4-X-6-X-8-9-X-X-12-13-X	3

Se a sequência de saída do código de taxa ½ for 111011001001 a sequência de saída do código perfurado de taxa ¾ será 11010000:

Saída do codificador de taxa ½: 11 10 11 00 10 01

Punção ou perfuração: 11 X0 1X 00 X0 0X

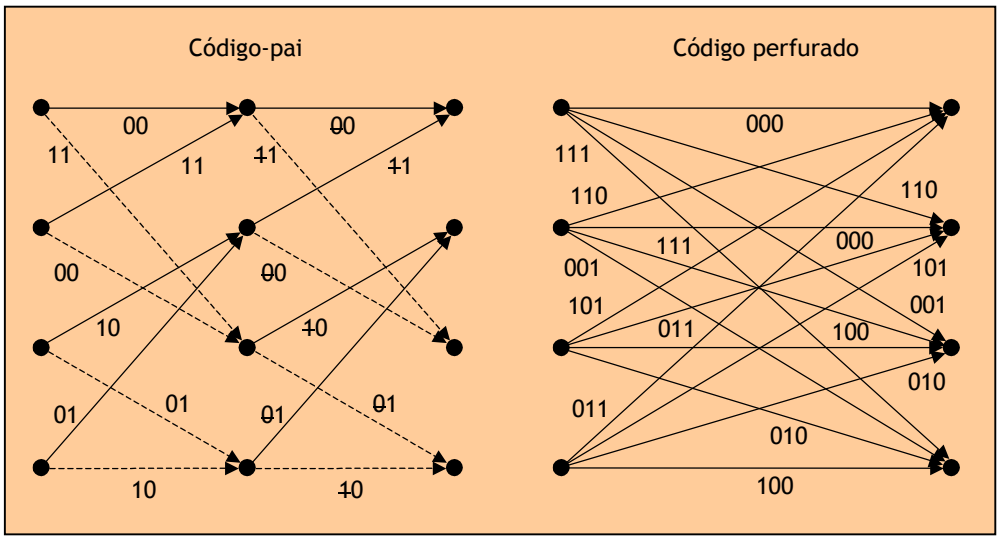
Saída do codificador de taxa ¾: 1 1 0 1 0 0 0 0

Códigos perfurados

É fácil de obter a treliça de um código perfurado. Para isso basta dispor da treliça do código-pai e atender ao padrão de perfuração, como se mostra no exemplo seguinte:

P.: Partindo do código de taxa ½ gerado por $\mathbf{G} = \begin{bmatrix} 111 & 101 \end{bmatrix}$ *qual é a treliça do código perfurado de taxa 2/3 no qual o terceiro bit em quatro é puncionado?*

R.: Começa-se pela treliça do código-pai, à esquerda na figura seguinte, e atende-se ao padrão de perfuração:



P.: Como fazer a descodificação de uma sequência perfurada?

R.: Reinserem-se bits fictícios na sequência recebida nas posições de onde foram retirados. Exemplo:

Padrão de perfuração: 1101 (o 3º bit em quatro é perfurado)
Sequência perfurada: 110110110100
Conversão: 11X0 11X0 11X0 10X0

Depois calculam-se as métricas do modo habitual, sendo as métricas relativas aos bits perfurados sempre nulas.

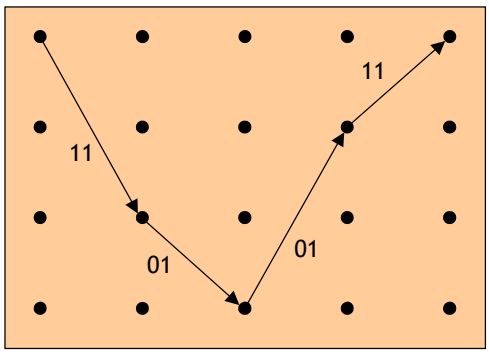
Distâncias euclidianas e distâncias de Hamming

- **Desmodulação branda**: o descodificador recebe valores reais.
- **Desmodulação rígida**: o descodificador recebe valores binários.

Seja desmodulação branda, recebendo-se a sequência de reais z_l

$$\{0{,}6 \quad -0{,}4 \quad -0{,}1 \quad 0{,}2 \quad 0{,}2 \quad 0{,}3 \quad -0{,}2 \quad -0{,}3\}.$$

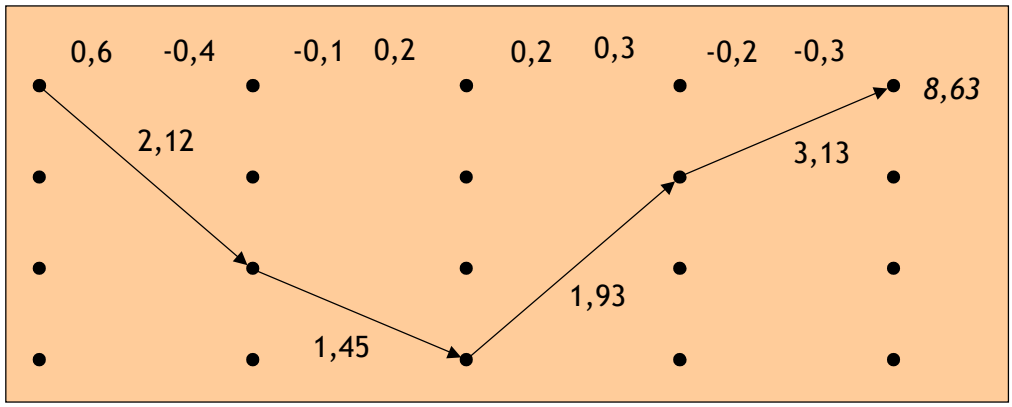
Qual é a sua distância euclidiana quadrática ao percurso da figura seguinte?



- Os bits 0 e 1 da figura deverão ser convertidos em $y_l = \pm 1$ (fazendo 0 → -1 e 1 → +1). Assim, o primeiro ramo, 11, corresponde ao ponto de coordenadas (+1, +1); o segundo ramo corresponde ao ponto de coordenadas (-1, +1)
- A distância euclidiana de cada ramo é: $\sum_{j=1}^{n}(z_{lj} - y_{lj}^{(i)})^2$ (se $n = 2$: $\left[z_{l1} - y_{l1}^{(i)}\right]^2 + \left[z_{l2} - y_{l2}^{(i)}\right]^2$)
- Exemplo de distância euclidiana: do par (–0,1; 0,2) ao ponto (-1, +1) (segundo ramo)

$$(-0{,}1+1)^2 + (0{,}2-1)^2 = 1{,}45$$

- Métrica acumulada: 2,12 + 1,45 + 1,93 + 3,13 = 8,63



Descodificador de máxima verosimilhança (algoritmo de Viterbi)

1. Suponhamos que recebemos a sequência $Z = Y + E$. Na treliça ou na árvore o percurso de Z diverge do percurso de $Y \rightarrow$ pode ou não ser um percurso válido.

2. Se não for um percurso válido um *descodificador de máxima verosimilhança* tenta encontrar o percurso válido que tenha a *menor distância de Hamming* de Z.

3. Mas… há 2^n percursos possíveis para uma sequência-mensagem arbitrária de n bits!

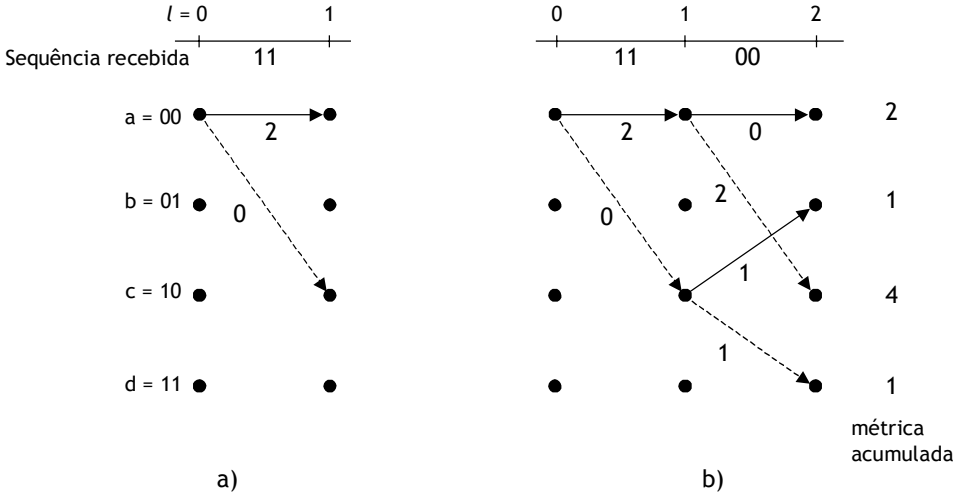
4. No algoritmo de Viterbi limita-se a comparação a $2^{k(N-1)}$ *percursos sobreviventes* (que é um número independente do tamanho da mensagem, n).

Isto torna viável a descodificação de *máxima verosimilhança*, baseada em *métricas* e *percursos sobreviventes*:

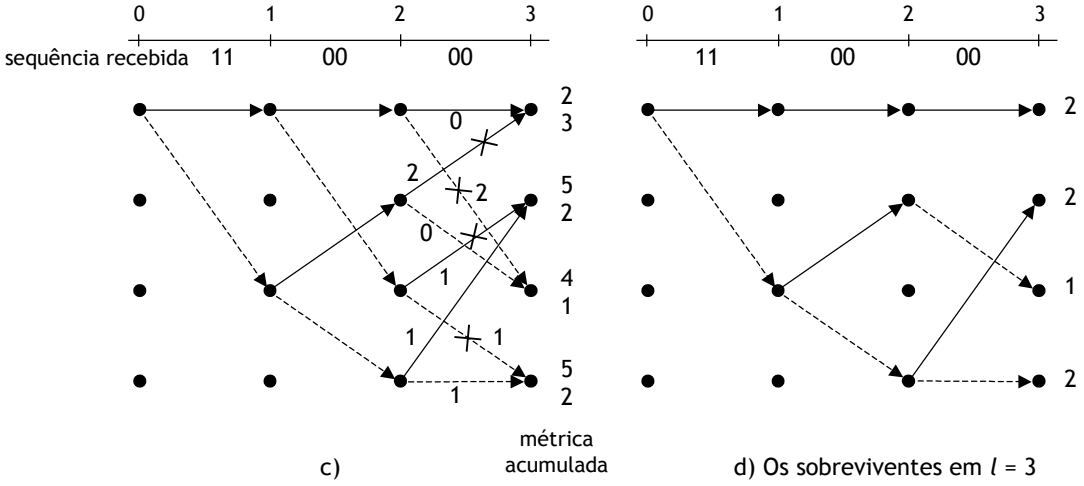
- O algoritmo de Viterbi atribui uma *métrica* a cada ramo de cada percurso sobrevivente:
    - A métrica é igual à *distância de Hamming* do ramo correspondente de Z se se tratar de desmodulação rígida (supõe-se $P_0 = P_1 = 1/2$);
    - A métrica é igual à *distância euclidiana*, se se tratar de desmodulação branda.
- A *métrica* de um percurso é igual à *soma das métricas* dos ramos constituintes.
- O descodificador deve calcular 2 métricas por cada nó e armazenar $2^{k(N-1)}$ percursos sobreviventes, cada um com n ramos.
- Sempre que dois percursos convergem num nó, sobrevive aquele que tiver menor métrica. Em caso de empate escolhe-se um à sorte.
- Z é descodificado (estimado) como o *percurso sobrevivente* com *menor métrica*.

O algoritmo de Viterbi

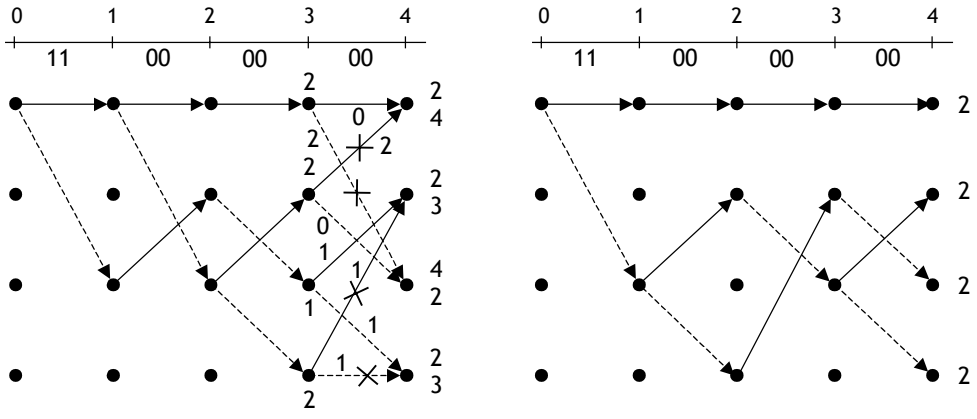
Mensagem: 1 0 1 1 1 0 11 0 0
Sequência codificada: 11 10 00 01 10 01 00 01 01 11
Sequência recebida: 11 00 00 00 10 01 00 01 01 11



a) b)

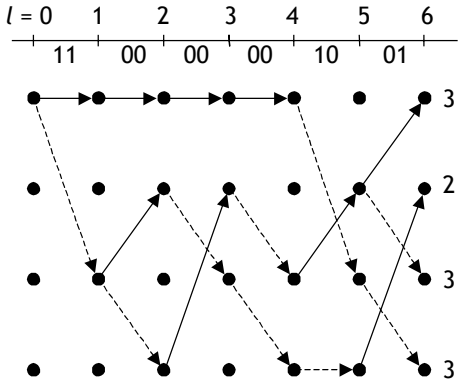


c) d) Os sobreviventes em *l* = 3

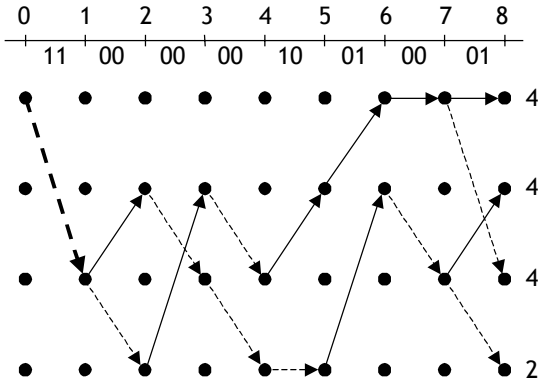


e) f) Os sobreviventes em l = 4

O algoritmo de Viterbi (cont.)

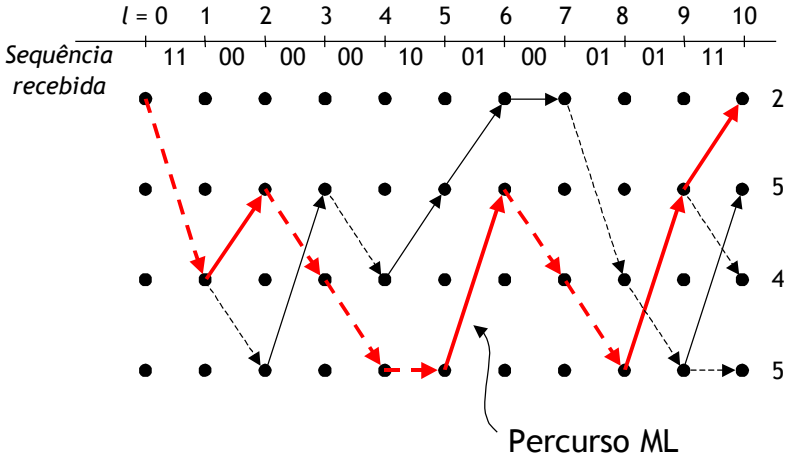


g) Os sobreviventes em $l = 6$



h) Os sobreviventes em $l = 8$

O percurso de máxima verosimilhança (percurso *ML*)



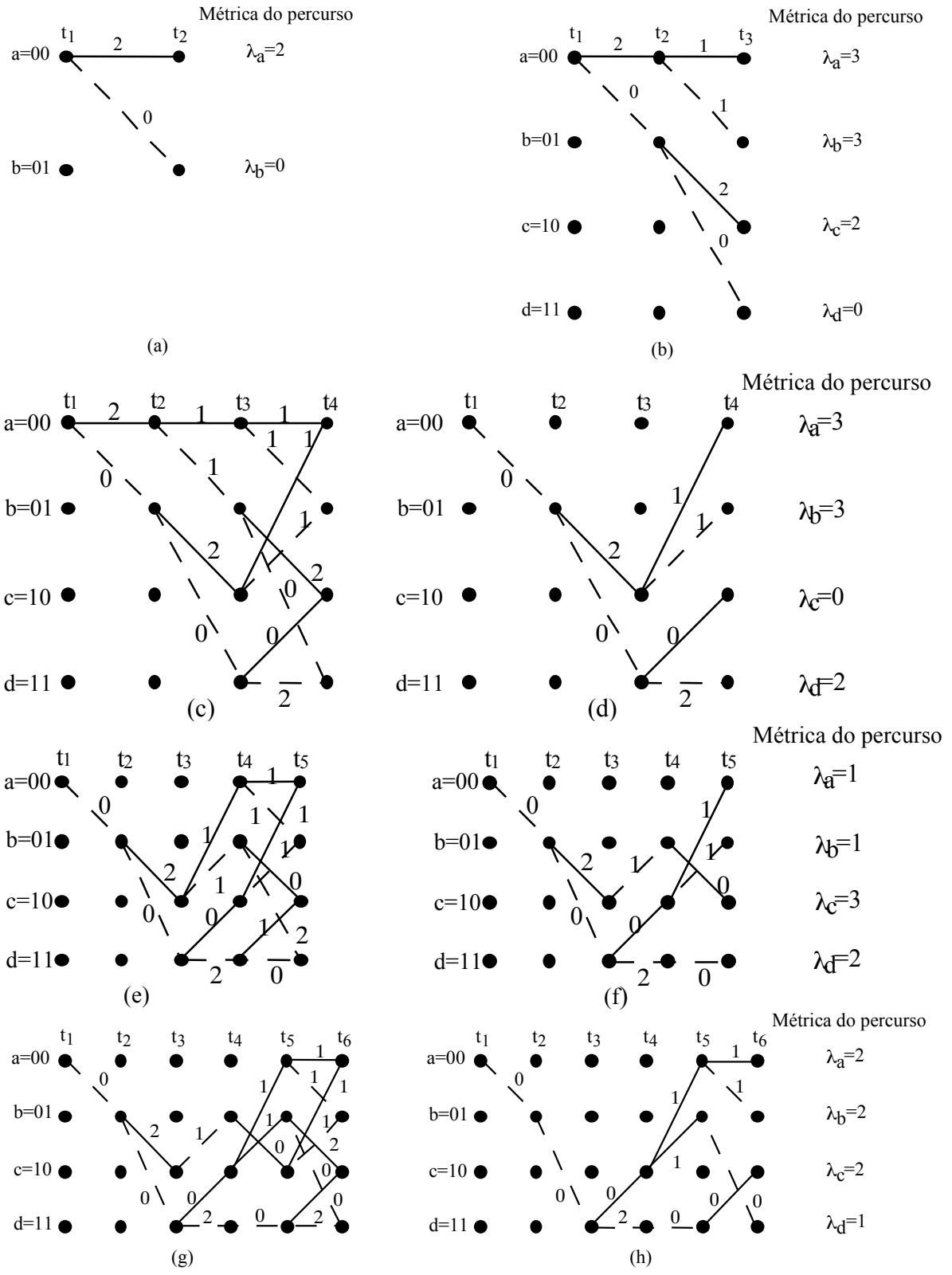
Mensagem enviada: 1 0 1 1 1 0 11 0 0

Sequência estimada: 1110000110010001 0111

Mensagem estimada: 1 0 1 1 1 0 1 1 0 0

Portou-se bem, o algoritmo: corrigiu os dois erros que sabíamos existirem na sequência recebida

Outro exemplo de descodificação (Viterbi)



Selecção de percursos sobreviventes. (a) Sobreviventes em t_2. (b) Sobreviventes em t_3. (c) Comparações de métricas em t_4. (d) Sobreviventes em t_4. (e) Comparações de métricas em t_5. (f) Sobreviventes em t_5. (g) Comparações de métricas em t_6. (h) Sobreviventes em t_6.

Exemplo de descodificação com decisões brandas

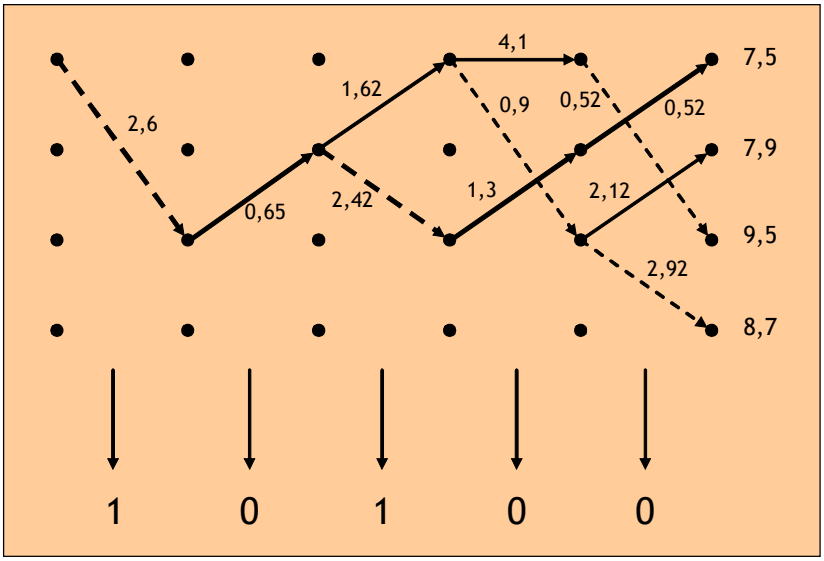
- Codificador (2, 1, 3) habitual
- Mensagem: 10100
- Sequência codificada: 11 10 00 10 11
- Sequência recebida: {-0,6; 0,8; 0,3; -0,6; 0,1; 0,1; 0,7; 0,1; 0,6; 0,40}

O percurso ML é aquele cuja distância euclidiana quadrática à sequência recebida é a menor.

- Exemplo de distância euclidiana quadrática:

$$1^{\text{o}} \text{ ramo: } (1+0,6)^2 + (1-0,8)^2 = 2,6$$

- O percurso ML tem uma métrica total de 7,5:

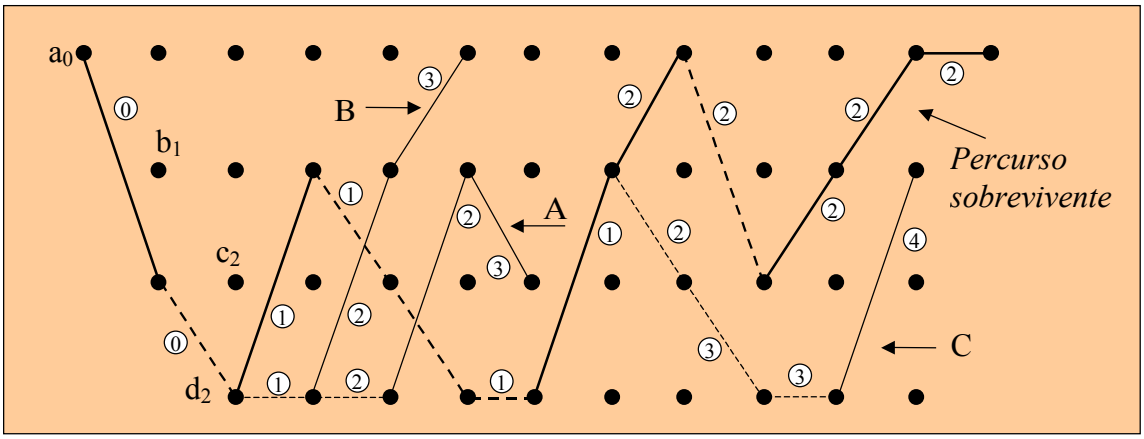


Descodificação sequencial

Mesmo código, mesma sequência recebida, mesma treliça de código e mesmas métricas:

Procedimento:

Em cada nó toma-se a direcção do ramo com menor métrica. Em caso de igualdade escolhe-se um ramo à sorte. Se neste caso a métrica aumentar muito então é preciso voltar atrás ao nó em questão e optar por outro ramo.

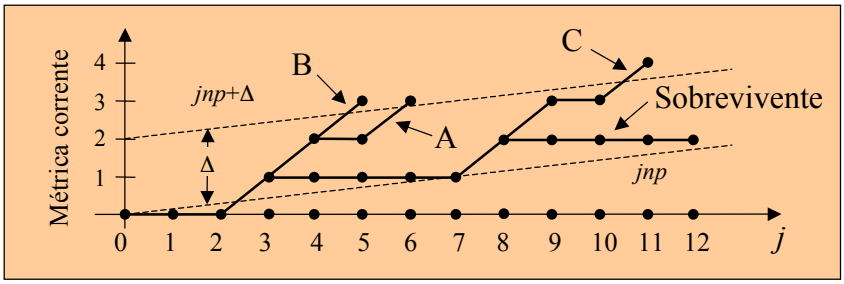


Ramos abandonados no exemplo acima: A, B e C.

Qual o critério para "voltar atrás e tentar de novo"? É o valor esperado da métrica corrente em cada nó.

Se p for a probabilidade de erro de transmissão/bit $\Rightarrow$ a métrica corrente **esperada** no nó j do percurso correcto é jnp (número esperado de erros acumulados nos bits de Z, nesse nó).

$\Rightarrow$ O descodificador sequencial abandona um percurso se a métrica corrente ultrapassar um limiar Δ acima de jnp.



$$p = \frac{1}{16}$$

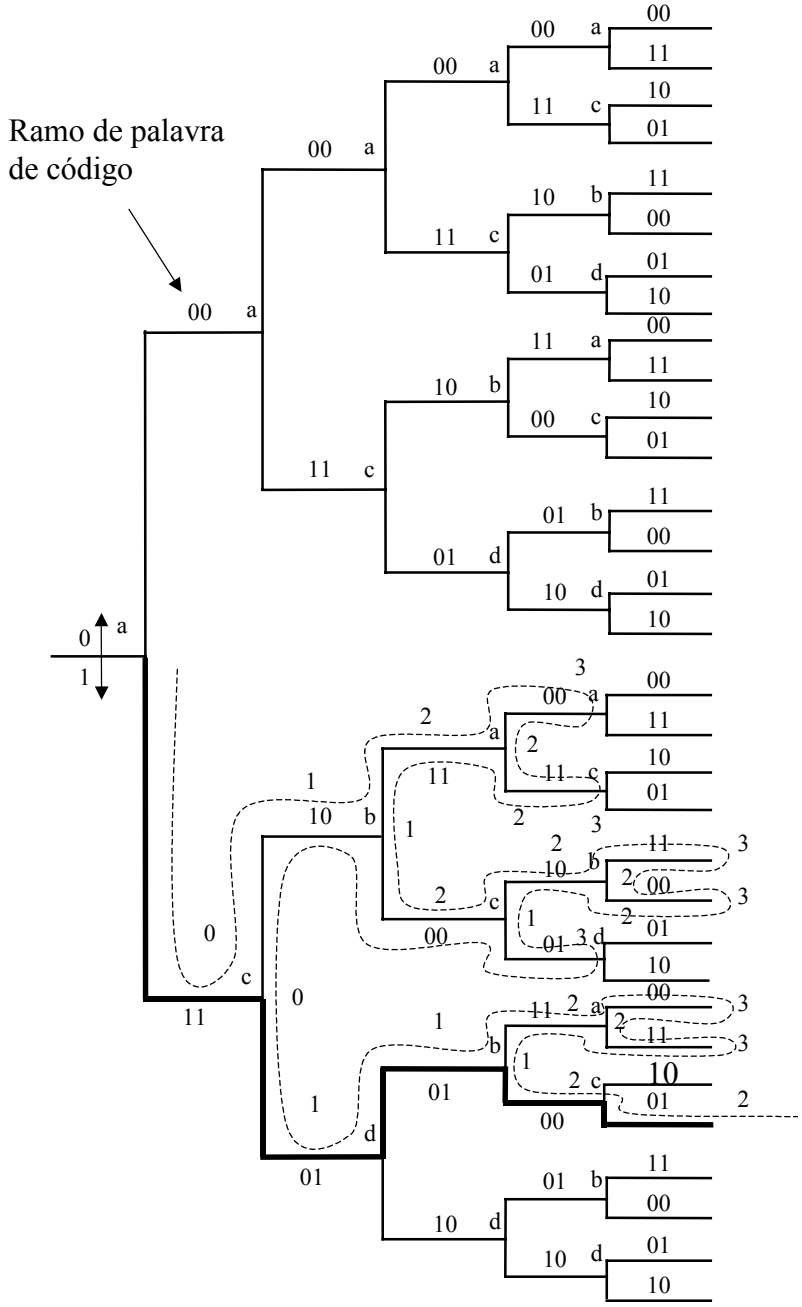
$$\Delta = 2$$

Exemplo de descodificação sequencial

Mensagem: 1 1 0 1 1…

Sequência transmitida: 11 01 01 00 01

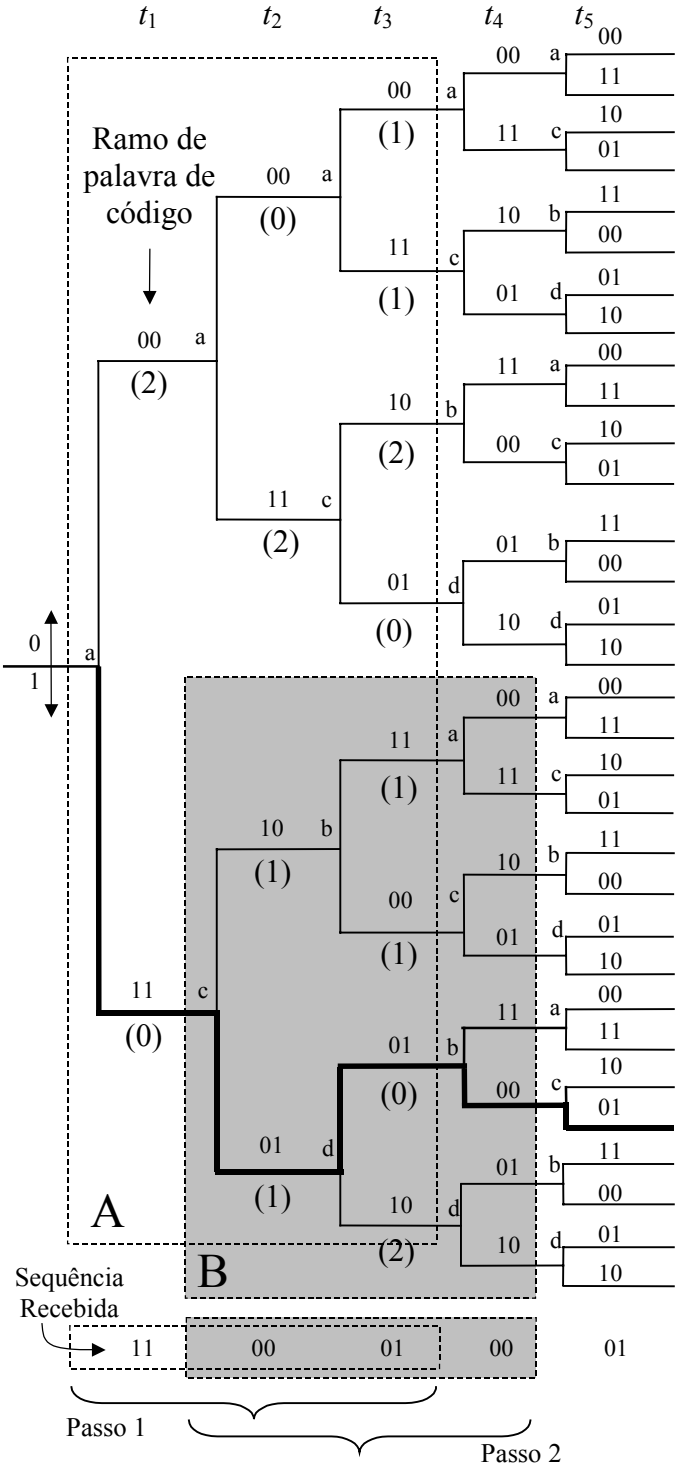
Mensagem estimada: 11 00 01 10 01



Mensagem descodificada: 1 1 0 1 1…

Exemplo de descodificação com "feedback"

O descodificador calcula 2^L métricas de percursos e decide que o bit do primeiro ramo é 0 se o percurso de distância mínima estiver situado na parte superior da árvore e é 1 se esse percurso estiver na parte inferior.



A, B "look-ahead windows"

"Look-ahead length" (L) = 3

( ) - métrica do ramo

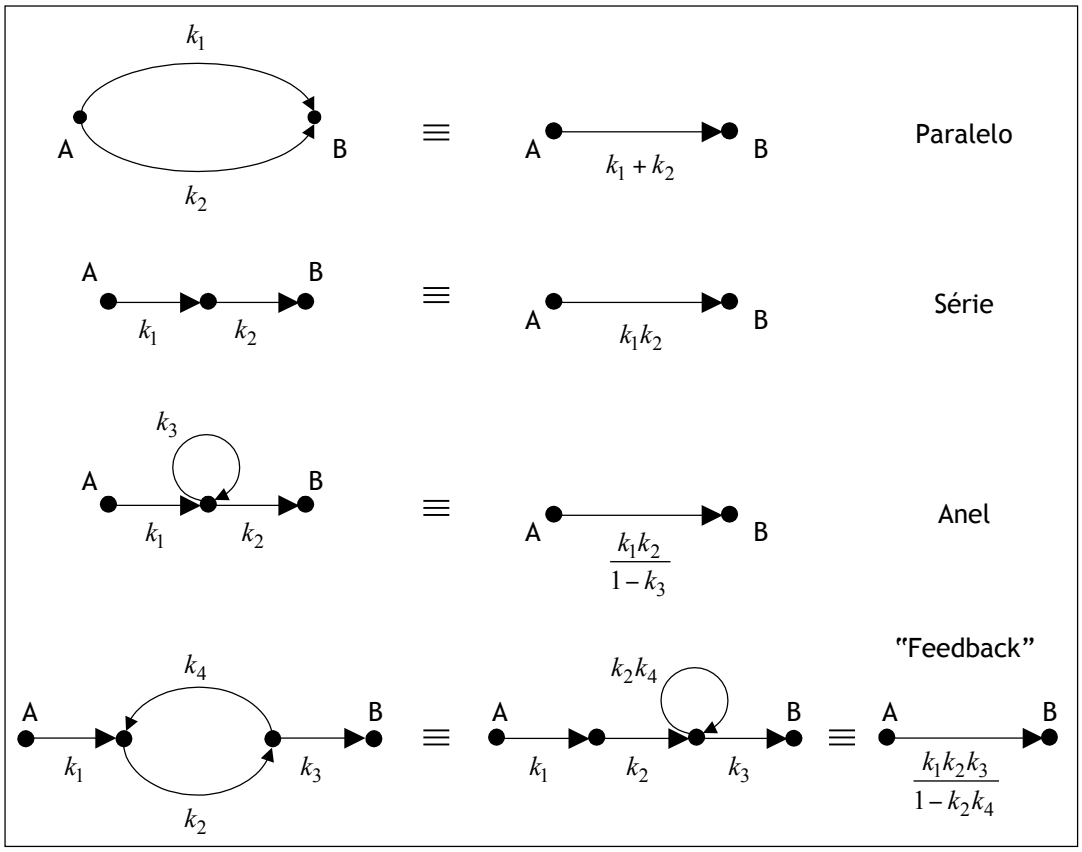
Métrica da janela A	3	3	6	4
(de cima para baixo)	2	2	1	3

O percurso mínimo (de métrica 1) está na parte inferior da árvore ⇒ Decidimos pelo bit "1" e passamos à janela B.

Anexo 1

Regra de Mason para simplificação de gráficos de fluxo

A simplificação de gráficos de fluxo pode fazer-se aplicando a regra de Mason, resumida nas equivalências seguintes:



Estas regras permitem calcular com facilidade a função de transferência e a distância livre de códigos convolucionais. Experimente!