1 Teoria da Informação: Codificação de Fonte

O estudo de um sistema de comunicações digitais envolve dois aspectos cruciais:

1. a eficiência da representação da informação gerada pela fonte;
2. a taxa de transmissão à qual é possível enviar a informação com fiabilidade através de um canal ruidoso.

A teoria da informação estabelece os limites fundamentais associados às questões acima referidas. A saber:

I. o número mínimo de unidades de informação binária (bit) por símbolo necessário para representar completamente a fonte;
II. o valor máximo da taxa de transmissão que garante fiabilidade da comunicação através de um canal ruidoso.

Começaremos por abordar o primeiro dos problemas acima enunciados, isto é, o da codificação de fontes discretas (ou digitais).

1.1 Modelo de uma Fonte Discreta

Consideremos, a título de exemplo, uma fonte discreta que gera símbolos binários. Observemos as duas sequências binárias seguintes:

A: 0011010110001101…
B: 1000100010001000…

Enquanto a sequência **B** parece ser constituída pela repetição periódica do padrão {1000}, a lógica de ocorrência dos símbolos binários na sequência **A** é imperceptível, tornando difícil ou mesmo impossível predizer as ocorrências futuras. No entanto, ambas as sequências poderiam ter sido geradas pela mesma fonte binária. Por outro lado, no outro extremo do sistema de comunicações o destinatário não tem conhecimento da sequência gerada pela fonte.

Estas considerações mostram que a escolha de um modelo determinístico para representar o comportamento da fonte de informação não é o mais adequado. Com efeito, para um observador externo, a saída da fonte digital num dado instante tem sempre associada alguma incerteza. Voltemos ao exemplo das sequências **A** e **B** e suponhamos que o observador não tem memória, isto é, observa a saída num dado instante e esquece-a antes que um novo símbolo seja gerado. Suponhamos ainda que o número de ocorrências de "0" e de "1" vai sendo actualizado. Após ter sido observado um número significativo de saídas da fonte, o grau de incerteza associado à ocorrência de cada um dos símbolos binários é naturalmente diferente conforme se considera a sequência **A** ou a sequência **B**. Enquanto que em **B** a incerteza associada à ocorrência de "0" é menor do que a associada à ocorrência de "1", em **A** o grau de incerteza é igual para ambos os símbolos. O conceito de incerteza associada a um acontecimento está assim intimamente ligado à probabilidade de ocorrência desse acontecimento. Em consequência deste facto, podemos ainda avançar com a seguinte ideia: se a um valor baixo da probabilidade de ocorrência de um acontecimento corresponde um valor elevado da incerteza associada, então da ocorrência desse acontecimento deve resultar um ganho de informação também ele elevado.

À luz das ideias anteriores, deve concluir-se que a fonte de informação deve ser representada usando um modelo aleatório.

1.1.1 Fonte Discreta sem Memória

Consideremos uma fonte digital que gera símbolos de um alfabeto

$$A = \{m_i, i = 1,2,\ldots,M\}$$

com probabilidade $p_i = Pr\{m_i\}$ tal que:

$$\sum_{i=1}^{M} p_i = 1. \qquad (1.1)$$

Def. 1.1: Uma fonte discreta sem memória gera ao longo do tempo símbolos estatisticamente independentes. □

De acordo com a definição anterior, a probabilidade de ocorrência de qualquer sequência gerada pela fonte é dada pelo produto das probabilidades de ocorrência dos símbolos que a constituem.

Exemplo 1.1: Consideremos a sequência temporal $S = \{m_1, m_{M-1}, m_5, m_5, m_3\}$ gerada pela fonte A. Supondo que esta fonte não tem memória, então

$$Pr\{S\} = p_1 p_{M-1} p_5^2 p_3. \qquad \square$$

1.2 Informação e Entropia

Consideremos uma sequência muito longa de K símbolos do alfabeto A gerados pela fonte discreta definida na subsecção 1.1.1. Uma maneira possível de avaliar o conteúdo informativo da fonte, isto é, a informação própria, consiste em determinar o número total de mensagens (ou sequências) de comprimento K que a fonte pode gerar. Note-se que a informação própria da fonte cresce com o número de mensagens possíveis. Portanto, é equivalente usar o número de mensagens ou o respectivo logaritmo, uma vez que a função logarítmica é monótona crescente.

O número Ω de mensagens de comprimento K, incluindo K_1 ocorrências do símbolo m_1, K_2 do símbolo m_2, etc., K_M do símbolo m_M, é dado por

$$\Omega = \frac{K!}{K_1! K_2! \cdots K_M!}, \qquad (1.2)$$

onde

$$K = \sum_{i=1}^{M} K_i. \qquad (1.3)$$

Supondo que K é tão elevado que qualquer dos K_i é também muito grande, podemos calcular uma aproximação de Ω usando a fórmula de Stirling

$$K! \cong (2\pi)^{1/2} e^{-K} K^{K+\frac{1}{2}}$$

em (1.2), obtendo-se

$$\Omega \cong \frac{(2\pi)^{1/2}\, e^{-K} K^{K+\frac{1}{2}}}{(2\pi)^{M/2} \prod_{i=1}^{M} e^{-K_i} \prod_{i=1}^{M} K_i^{K_i+\frac{1}{2}}}. \qquad (1.4)$$

Como

$$\prod_{i=1}^{M} e^{-K_i} = e^{-\sum_{i=1}^{M} K_i} = e^{-K},\ K \cong K + \frac{1}{2},\ K_i \cong K_i + \frac{1}{2},\ K^K = \prod_{i=1}^{M} K^{K_i},$$

de (1.4), vem:

$$\Omega \cong (2\pi)^{-(M-1)/2} \prod_{i=1}^{M} \left(\frac{K}{K_i} \right)^{K_i}.$$

Aplicando a função log em ambos os membros da relação anterior, e tendo em conta que a probabilidade de ocorrência do símbolo m_i é o número p_i para o qual converge a razão K_i/K quando $K_i, K \longrightarrow +\infty$, vem

$$\log \Omega \cong -\frac{1}{2}(M-1)\log(2\pi) - K \sum_{i=1}^{M} p_i \log p_i,$$

ou, tendo em conta que o 2º termo se torna dominante para K suficientemente elevado,

$$\log \Omega \cong -K \sum_{i=1}^{M} p_i \log p_i. \qquad (1.5)$$

A fórmula anterior dá o valor aproximado da informação própria de uma fonte discreta com M símbolos, ou dito de outra forma, de uma mensagem de comprimento muito longo K gerada pela mesma fonte.

Observando a fórmula (1.5), verificamos que, em média, a informação por símbolo é medida pela quantidade

$$\frac{\log \Omega}{K} \cong -\sum_{i=1}^{M} p_i \log p_i. \qquad (1.6)$$

Por outro lado, a quantidade $-\log p_i$ está associada à ocorrência do símbolo m_i, ou seja, é uma variável aleatória discreta que toma o valor real $I(m_i) = -\log p_i$ com probabilidade p_i. Note-se que o 2º membro de (1.6), não é mais do que o valor expectável (média) desta variável aleatória.

1.2.1 Medida de Informação

A discussão anterior sugere então a seguinte definição para o ganho de informação associado à ocorrência de um símbolo:

Def. 1.2: Considere-se a fonte discreta sem memória introduzida na subsecção 1.1.1. A informação associada à ocorrência de um símbolo desta fonte é definida por:

$$I(m_i) = \log(1/p_i) = -\log p_i,\ i = 1,2,\ldots M. \qquad (1.7)\ \square$$

Esta medida quantitativa da informação gerada pela ocorrência de um símbolo na saída de uma fonte discreta foi introduzida por Claude E. Shannon no seu trabalho intitulado The Mathematical Theory of Communication, publicado em 1948 no nº de Outubro do Bell System Technical Journal. É interessante notar que, sendo o conceito de informação relativamente subjectivo, a medida (1.7) dá conta de algumas das suas propriedades qualitativas:

1. $I(m_i) = 0$ se $p_i = 1$ (1.8)
2. $I(m_i) \geq 0$ (1.9)
3. $I(m_i) > I(m_j)$ se $p_i < p_j$ (1.10)

ou seja,

1. o ganho de informação resultante da ocorrência do acontecimento certo é nulo;
2. excepto no caso do acontecimento certo, a ocorrência de um qualquer acontecimento conduz a um ganho de informação;
3. quanto menor for a probabilidade de ocorrência de um acontecimento maior é o ganho de informação que lhe está associado.

Tendo em conta (1.7), verificamos que a informação associada à ocorrência simultânea de dois acontecimentos estatisticamente independentes

$$\begin{aligned} I(m_i, m_j) &= -\log\left(\Pr\{m_i, m_j\}\right) \\ &= -\log(p_i p_j) \\ &= -\log p_i - \log p_j \\ &= I(m_i) + I(m_j) \end{aligned} \qquad (1.11)$$

é a soma da informação associada a cada uma das ocorrências.

Nas expressões anteriores é usual considerar a função logarítmica definida na base 2. A unidade de medida de informação define-se como se segue.

Def. 1.3: a unidade binária de informação (bit) é a informação própria associada a cada um dos símbolos de uma fonte binária com símbolos equiprováveis:

$$I(0) = I(1) = -\log_2 \frac{1}{2} = 1 \text{ bit} \qquad (1.12)$$ □

1.2.2 Entropia de uma Fonte Discreta sem Memória

Já foi sublinhado anteriormente que a informação própria de um símbolo, ver Def. 1.2, é uma variável aleatória discreta em que cada realização $I(m_i), i = 1, \ldots, M$, ocorre com probabilidade $p_i, i = 1, \ldots, M$. Recorde-se que esta distribuição de probabilidade verifica (1.1).

Def. 1.4: A entropia de uma fonte discreta sem memória é o valor expectável da informação própria dos símbolos da fonte:

$$H(A) = E\{I(m)\} = \sum_{i=1}^{M} p_i I(m_i) = -\sum_{i=1}^{M} p_i \log_2 p_i \qquad (1.13)$$ □

Exemplo 1.2: Consideremos uma fonte binária com símbolos equiprováveis. De acordo com (1.13), a entropia desta fonte vale

$$H(A) = -\sum_{i=1}^{2} \frac{1}{2} \log_2 \frac{1}{2} = 1 \text{ bit/símbolo}$$

Consideremos agora o caso mais geral em que

$$\begin{aligned} \Pr\{0\} &= p \\ \Pr\{1\} &= 1-p \end{aligned} . \qquad (1.14)$$

Recorrendo novamente a (1.13), podemos escrever

$$H(A) = -p \log_2 p - (1-p)\log_2(1-p) \text{ bit/símbolo.} \qquad (1.15)$$

A entropia da fonte binária, expressa em (1.15), está representada na Figura 1.1 em função da probabilidade p .

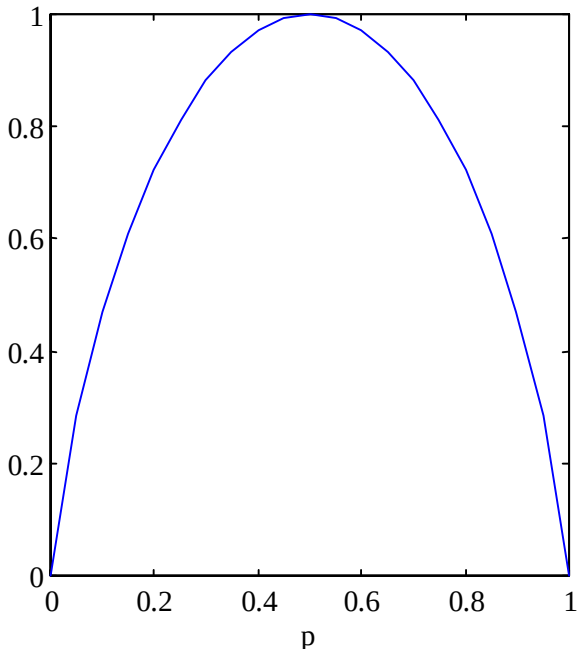

Figura 1.1: Entropia da fonte binária

É interessante notar que:

1. quando $p = 0$, a entropia $H(A) = 0$, pois $x \log x \to 0$ quando $x \to 0$;
2. a entropia $H(A) = 0$ quando $p = 1$;
3. a entropia $H(A)$ atinge o valor máximo $H(A) = 1$ bit/símbolo quando $p = 1/2$, ou seja, quando os símbolos são equiprováveis.

□

Estas propriedades, inferidas a partir do exemplo anterior são generalizáveis para qualquer fonte discreta sem memória.

1.2.2.1 Propriedades da Entropia de uma Fonte Discreta sem Memória

A entropia $H(A)$ da fonte discreta sem memória A, definida na subsecção 1.2.2, é limitada de acordo com a seguinte desigualdade:

$$0 \le H(A) \le \log_2 M, \tag{1.16}$$

onde M é o número de símbolos do alfabeto A. Podemos mostrar que:

P1. $H(A) = 0$ sse $p_i = 1$ para algum i e $p_j = 0,\ j = 1,\ldots,i-1,i+1,\ldots,M$. O limiar inferior da entropia corresponde portanto à *ausência de incerteza* sobre a saída da fonte.

P2. $H(A) = \log_2 M$ sse $p_i = 1/M, \forall i = 1,\ldots,M$, isto é, sse todos os símbolos forem equiprováveis. O limiar superior da entropia corresponde assim ao *máximo da incerteza*.

A expressão (1.13) da Def. 1.4 pode escrever-se na forma

$$H(A) = \sum_{i=1}^{M} p_i \log_2 (1/p_i). \tag{1.17}$$

Como $0 \le p_i \le 1$, conclui-se que todas as parcelas de (1.17) são não negativas. Portanto, $H(A) = 0$ sse todas as parcelas forem nulas. Como a distribuição de probabilidade verifica a restrição (1.1), conclui-se que o limiar inferior da entropia só é atingido na *ausência de incerteza*, como se diz em **P1**.

□

O problema da maximização de (1.17) pode ser formulado do seguinte modo:

Determinar a distribuição $p_i, i = 1,\ldots,M$, que maximiza

$$H(A) = \sum_{i=1}^{M} p_i \log_2 (1/p_i)$$

sujeita à restrição

$$\sum_{i=1}^{M} p_i = 1.$$

Para o resolver podemos usar o método dos multiplicadores de Lagrange. Definindo a Lagrangeana

$$L(p_1,\ldots,p_M) = -\sum_{i=1}^{M} p_i \log_2 p_i + \lambda\left(1 - \sum_{i=1}^{M} p_i\right), \tag{1.18}$$

onde λ é o multiplicador de Lagrange, verificamos que maximizar (1.18) é o mesmo que maximizar $H(A)$, pois a segunda parcela de (1.18) é sempre nula. Diferenciando $L(\cdot)$ em ordem a cada um dos p_i, e igualando a zero, obtém-se o seginte sistema de equações[1]:

[1] Recorda-se que $\dfrac{d \log_2 x}{dx} = \dfrac{1}{\ln 2}\dfrac{1}{x}$

$$-\log_2 p_i = \frac{1}{\ln 2} + \lambda, \quad i = 1, \ldots, M.$$

Concluímos assim que, mesmo sem calcular o valor de λ que garante a restrição, todos os p_i são iguais. Portanto, como se diz em **P2**, a distribuição de probabilidade que maximiza[2] a entropia é

$$p_i = \frac{1}{M}, \; i = 1, \ldots, M.$$

Deste modo, o valor máximo da entropia é $H_{max}(A) = \sum_{i=1}^{M} \frac{1}{M} \log_2 M = \log_2 M.$

□

1.2.2.2 Desigualdade Fundamental

P3. Seja $p_i, i = 1, \ldots, M,$ uma distribuição de probabilidade associada aos símbolos $m_i \in A, i = 1, \ldots, M.$ Obviamente, a restrição (1.1) é verificada. Sendo $q_i, i = 1, \ldots, M$, uma outra distribuição de probabilidade,

$$\sum_{i=1}^{M} q_i = 1, \tag{1.19}$$

então

$$\sum_{i=1}^{M} p_i \log_2 \left(\frac{q_i}{p_i} \right) \leq 0, \tag{1.20}$$

sendo atingido o valor máximo quando

$$q_i = p_i, \; i = 1, \ldots, M. \tag{1.21}$$

A demonstração deste facto resulta directamente da resolução do seguinte problema:

Sendo $p_i, i = 1, \ldots, M$, uma distribuição de probabilidade, determinar os valores de $q_i, i = 1, \ldots, M,$ que maximizam

$$\sum_{i=1}^{M} p_i \log_2 \left(\frac{q_i}{p_i} \right), \tag{1.22}$$

sujeitos à restrição

$$\sum_{i=1}^{M} q_i = 1.$$

[2] Pode verificar-se que com esta distribuição de probabilidade a segunda derivada da Lagrangeana é negativa. De facto, $\left. \frac{d^2 L}{dp_i^2} \right|_{p_i = 1/M} = -\frac{M}{\ln 2} < 0.$

Usando (1.19) e (1.22) podemos escrever a Lagrangeana

$$L(q_1,\ldots,q_M)=\sum_{i=1}^{M} p_i \log_2 q_i - \sum_{i=1}^{M} p_i \log_2 p_i + \lambda\left(1-\sum_{i=1}^{M} q_i\right)$$

Diferenciando em ordem aos $q_i, i=1,\ldots,M,$ e igualando a zero, obtém-se o sistema de equações

$$(\lambda \ln 2) q_i = p_i, i=1,\ldots,M. \qquad (1.23)$$

Tendo em conta (1.1) e (1.19) e somando membro a membro todas as equações do sistema (1.23), conclui-se que $\lambda = 1/\ln 2$. Usando este valor em (1.23), obtém-se a distribuição que maximiza[3] (1.22):

$$q_i = p_i, i=1,\ldots,M. \qquad (1.24)$$

Usando este resultado, verifica-se facilmente que o máximo de (1.22) é nulo e, portanto, a desigualdade (1.20) fica demonstrada.

□

A desigualdade (1.20) é conhecida por desigualdade fundamental e será usada mais adiante para obter outros resultados importantes da Teoria da Informação.

1.3 Codificação de Fonte

Consideremos o problema da codificação de símbolos pertencentes a um alfabeto estendido de símbolos estatisticamente independentes. Em particular, consideremos um alfabeto de 37 símbolos equiprováveis: 26 letras, o espaço em branco, e 10 dígitos. Suponhamos que para codificar estes símbolos dispomos apenas de símbolos binários (bits[4]) e, naturalmente por razões de eficiência de representação, pretendemos usar palavras de código de comprimento mínimo. Suponhamos, a título de exemplo, que codificamos individualmente cada um dos 37 símbolos. Como $2^5 < 37 < 2^6$ precisamos de usar pelo menos 6 bits por símbolo. No entanto, esta estratégia de codificação resulta em desperdício uma vez que sobram $2^6 - 37 = 27$ palavras de código às quais não corresponde qualquer símbolo do alfabeto original. Outro modo de avaliar este desperdício consiste em verificar que a entropia da fonte vale $H = \log_2 37 = 5.21$ bit/símbolo (note-se que os símbolos são equiprováveis e portanto a entropia é igual à informação própria de cada símbolo). Como se vê, a informação média por símbolo é inferior ao comprimento da palavra de código indicando que este código corresponde a uma representação redundante do alfabeto considerado. Consideremos agora um novo alfabeto (extensão de 2ª ordem) em que cada símbolo estendido corresponde a um dos 37^2 pares de símbolos do alfabeto original. Para codificar cada um dos símbolos da extensão são necessários 11 bits, enquanto que a respectiva entropia é agora $H_2 = \log_2 37^2 = 10.42$ bit/símbolo estendido. Isto corresponde de facto a usar, em média, 5.5 bits por cada símbolo original, resultando numa estratégia de codificação mais eficiente. Este exercício pode ser continuado para extensões de ordem crescente, obtendo-se os resultados que se mostram na tabela 1.1. Da consulta desta tabela conclui-se:

[3] Pode verificar-se que, com esta distribuição, a segunda derivada da Lagrangeana é negativa.

[4] Aqui "bit" designa um símbolo binário e não deve ser confundido com a unidade binária de informação.

1. à medida que aumenta a ordem da extensão, vai diminuindo o número médio de bits necessários para codificar cada símbolo do alfabeto original;
2. esta diminuição não é uniforme, embora pareça convergir para a entropia do alfabeto original.

ordem da extensão	entropia	comprimento da palavra de código	comprimento médio por símbolo
1	5.21	6	6.00
2	10.42	11	5.50
3	15.63	16	5.33
4	20.84	21	5.25
5	26.05	27	5.40
6	31.26	32	5.33
7	36.47	37	5.29
8	41.68	42	5.25
9	46.89	47	5.22
10	52.09	53	5.30

Tabela 1.1: Eficiência da codificação de alfabetos estendidos

Este exercício, embora não seja conclusivo de um ponto de vista estritamente formal, sugere que a eficiência dos códigos está associado a extensões de ordem superior do alfabeto da fonte discreta.

1.3.1 Extensão de Fonte

Def. 1.5: Consideremos a fonte discreta sem memória definida na subsecção 1.1.1. A extensão de ordem K desta fonte é ainda uma fonte discreta sem memória com alfabeto

$$A^K = \{\sigma_1, \sigma_2, \ldots, \sigma_{M^K}\}, \sigma_i = (m_{i_1}, m_{i_2}, \ldots, m_{i_K}), m_{i_k} \in A \qquad (1.25)$$

e distribuição de probabilidade

$$p_i = \Pr\{\sigma_i\} = \Pr\{m_{i_1}\}\Pr\{m_{i_2}\}\cdots\Pr\{m_{i_K}\}, i = 1,2,\ldots,M^K. \qquad (1.26)$$ □

1.3.2 Entropia da Fonte Estendida

Antes de calcularmos a entropia da fonte A^K, extensão de ordem K da fonte A, vamos verificar que a distribuição (1.26) é de facto uma distribuição de probabilidade. Em primeiro lugar, qualquer dos p_i em (1.26) é um produto de probabilidades e portanto $0 \le p_i \le 1$. Notemos ainda que a extensão de ordem K se pode obter da extensão de ordem $K-1$, isto é,

$$\sigma_i^{(K)} = (\underbrace{m_{i_1}, \ldots, m_{i_{K-1}}}_{\sigma_i^{(K-1)}}, \begin{Bmatrix} m_1 \\ m_2 \\ \vdots \\ m_M \end{Bmatrix}), \qquad (1.27)$$

cada símbolo da extensão de ordem K−1 dá origem a M símbolos da extensão de ordem K. Portanto

$$\sum_{i=1}^{M^K} \Pr\{\sigma_i^{(K)}\} = \sum_{l=1}^{M^{K-1}} \Pr\{\sigma_l^{(K-1)}\} \underbrace{\sum_{i=1}^{M} \Pr\{m_i\}}_{=1} = \sum_{l=1}^{M^{K-1}} \Pr\{\sigma_l^{(K-1)}\} \qquad (1.28)$$

Prosseguindo o mesmo raciocínio, verificamos que esta igualdade se mantém válida seja qual for a ordem da extensão considerada. Em particular,

$$\sum_{i=1}^{M^K} \Pr\{\sigma_i^{(K)}\} = \sum_{i=1}^{M} \Pr\{m_i\} = 1. \qquad (1.29) \quad \square$$

Por definição, Def. 1.4, eq. (1.13), a entropia da extensão de ordem K da fonte A é

$$H(A^K) = -\sum_{i=1}^{M^K} \Pr\{\sigma_i^{(K)}\} \log_2 \Pr\{\sigma_i^{(K)}\},$$

ou seja,

$$\begin{aligned} H(A^K) &= -\sum_{l=1}^{M^{K-1}} \sum_{i=1}^{M} \Pr\{\sigma_l^{(K-1)}\} \Pr\{m_i\} \log_2 \left(\Pr\{\sigma_l^{(K-1)}\} \Pr\{m_i\} \right) \\ &= \underbrace{-\sum_{l=1}^{M^{K-1}} \Pr\{\sigma_l^{(K-1)}\} \log_2 \Pr\{\sigma_l^{(K-1)}\}}_{H(A^{K-1})} \underbrace{\sum_{i=1}^{M} \Pr\{m_i\}}_{=1} \\ &\quad + \underbrace{\sum_{l=1}^{M^{K-1}} \Pr\{\sigma_l^{(K-1)}\}}_{=1} \left[\underbrace{-\sum_{i=1}^{M} \Pr\{m_i\} \log_2 \Pr\{m_i\}}_{H(A)} \right]. \end{aligned}$$

Finalmente, podemos escrever

$$H(A^K) = H(A^{K-1}) + H(A),$$

e, repetindo argumentos, concluímos que:

> A entropia da extensão de ordem K de uma fonte discreta sem memória é igual a K vezes a entropia da fonte original, isto é,
>
> $$H(A^K) = KH(A). \qquad (1.30) \quad \square$$

1.3.3 Comprimento Médio do Código

A codificação de fonte consiste em atribuir uma palavra de código única a cada uma das mensagens geradas pela fonte. Aqui a palavra mensagem é usada indiscriminadamente para designar um símbolo da fonte original ou um símbolo de uma qualquer extensão do alfabeto fonte. Como vimos em discussão anterior, a eficiência da codificação está associada à parsimónia que se usa na escolha do comprimento das palavras de código. Por outro lado, para uma dada taxa de geração de símbolos, quanto maior for o comprimento das palavras de código maiores serão as necessidades em termos de taxa de transmissão. Como, em geral, os símbolos do alfabeto fonte não são equiprováveis, é razoável pensar em códigos de

comprimento variável: o comprimento de cada palavra de código deverá ser tanto menor quanto maior for a probabilidade de ocorrência do símbolo correspondente. Este procedimento tenderá a minimizar o comprimento médio do código (ou das palavras do código) sendo, portanto, eficiente.

O processo de codificação consiste em atribuir rótulos ou símbolos às saídas de uma fonte de informação. Temos então que distinguir entre os símbolos da fonte e os símbolos do alfabeto do código.

Def. 1.6: Sejam

$$A = \{m_1, \ldots, m_M\} \text{ e } C = \{\alpha_1, \ldots, \alpha_r\}$$

os alfabetos fonte e do código, respectivamente. Nos códigos de blocos,

$$\forall i = 1, \ldots, M: \; m_i \in A \longleftrightarrow c_i = \left(\alpha_{i_1}, \alpha_{i_2}, \ldots, \alpha_{i_{l_i}}\right), \alpha_{i_j} \in C,$$

isto é, a cada símbolo do alfabeto fonte faz-se corresponder uma e uma só palavra de código, cujo comprimento é variável. □

Obviamente que o problema inverso da codificação tem de ter solução única, isto é, qualquer código de fonte tem de ser univocamente descodificável. Nos códigos de comprimento fixo basta que a cada palavra c_i do código se faça corresponder uma e uma só mensagem $m_i \in A$ para que se garanta a descodificação única. No entanto, tal não é suficiente quando se trata de códigos de comprimento variável, sendo necessária uma condição adicional: a estrutura do código deve permitir identificar sem ambiguidade o início e o fim de cada palavra de código. Para ilustrar esta ideia consideremos os exemplos de códigos que se mostram na Tabela 1.2.

símbolo fonte	probabilidade de ocorrência	código I	código II	código III	código IV
m_1	0.500	00	0	0	0
m_2	0.250	01	1	10	01
m_3	0.250	10	00	110	011
m_4	0.125	11	11	111	0111

Tabela 1.2: Exemplos de códigos de fonte

O código I é um código binário simples e, portanto, univocamente descodificável. Neste caso, o comprimento médio do código

$$\overline{L} = \sum_{i=1}^{M} p_i l_i \qquad (1.31)$$

é obviamente igual a 2. Os restantes códigos usam comprimentos variáveis, sendo as palavras mais longas aquelas que correspondem aos símbolos menos prováveis. Ao contrário do que acontece no código IV, nenhuma palavra do código III constitui prefixo de outra. Estes códigos são designados por códigos de prefixo. Os códigos de prefixo são sempre univocamente descodificáveis. Com efeito, estes códigos são completamente descritos por uma estrutura em árvore com um estado inicial e M estados finais, como se pode ver na Figura 1.2 para o caso do código III. Partindo do estado inicial, o descodificador vai descendo ao longo da árvore à medida que recebe cada bit e até atingir um dos quatro estados terminais. Quando isto acontece, o símbolo foi descodificado e o descodificador retorna ao estado

inicial. O comprimento médio deste código de prefixo vale $\overline{L} = 1.75$. O código III é, como seria de esperar, mais eficiente do que o código I. O código IV tem comprimento $\overline{L} = 1.875$ e, embora não sendo um código de prefixo, é também univocamente descodificável. Basta notar que nenhuma palavra de código exibe dois bits "0" consecutivos e que todas elas são iniciadas por "0". Quando é detectado um "0" o descodificador sabe que se inicia uma palavra do código, bastando contar o número de bits "1" consecutivos para identificar o símbolo fonte correspondente. Finalmente, é fácil verificar que o código II, sendo aparentemente o mais eficiente $\left(\overline{L} = 1.25\right)$, não é univocamente descodificável.

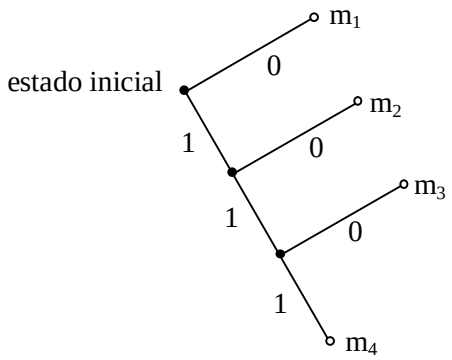

Figura 1.2: Árvore de descodificação de um código de prefixo

Os códigos de prefixo, como o código III, são códigos instantâneos pois qualquer palavra de código é descodificada assim que a totalidade dos símbolos que a constituem é recebida. Ao contrário, no código IV o símbolo "0" funciona como separador, pelo que cada palavra é descodificada com atraso de um bit.

1.3.4 Desigualdade de Kraft

Como se conclui da discussão anterior, é necessário impor restrições na estrutura de um código instantâneo de comprimento variável de modo a garantir a unicidade da descodificação.

A desigualdade de Kraft estabelece uma condição necessária e suficiente de existência de um código instantâneo formado por palavras de comprimento variável l_i:

$$\xi = \sum_{i=1}^{M} r^{-l_i} \leq 1, \qquad (1.32)$$

onde r é o número de símbolos do alfabeto do código. A soma ξ é designada por soma de Kraft.

Consideremos o exemplo da Figura 1.2, onde $M = 4$ e $r = 2$: a soma de Kraft vale $\xi = 2^{-1} + 2^{-2} + 2^{-3} + 2^{-3} = 1$, e portanto a desigualdade de Kraft (1.32) é verificada. Este facto garante que existe um código binário instantâneo, univocamente descodificável, e cuja distribuição dos comprimentos das palavras de código é o do exemplo. Sublinha-se que a verificação da desigualdade de Kraft não define o código, garantindo tão somente a sua existência.

Para provar a desigualdade de Kraft podemos usar um raciocínio simples baseado na árvore de codificação. Consideremos uma árvore r – ária onde cada nó tem r descendentes. Suponhamos ainda que cada ramo representa um símbolo da palavra de código. Por exemplo,

os r ramos que partem da raíz representam os r possíveis valores do primeiro símbolo da palavra de código. Portanto, cada palavra de código corresponde a um nó terminal da árvore. O percurso entre a raíz e um destes nós terminais identifica os símbolos que fazem parte da palavra de código. A Figura 1.3 ilustra estas ideias para o caso binário, r=2.

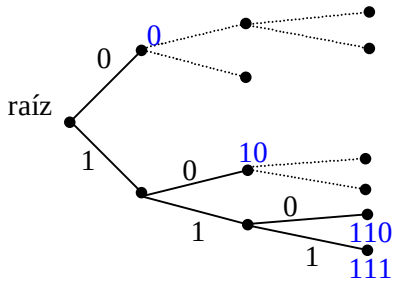

Figura 1.3: Árvore de codificação para a desigualdade de Kraft

A condição de o código ser de prefixo implica que nenhuma palavra de código seja ascendente de qualquer outra palavra de código na árvore. Assim, cada palavra de código elimina os seus descendentes como possíveis palavras do código.

Seja l_{max} o comprimento da palavra mais longa do código. Consideremos todos os nós ao nível l_{max} da árvore. Alguns são palavras de código, outros são descendentes de palavras de código e os restantes nem uma coisa nem outra. Qualquer palavra de código ao nível l_i terá $r^{l_{max}-l_i}$ descendentes no nível l_{max}. Estes conjuntos de descendentes têm de ser disjuntos e, por outro lado, o número total de nós neles incluídos deverá ser inferior ou igual a $r^{l_{max}}$. Portanto, somando para todas as palavras de código, tem-se

$$\sum_{i=1}^{M} r^{l_{max}-l_i} \le r^{l_{max}}, \tag{1.33}$$

ou seja,

$$\sum_{i=1}^{M} r^{-l_i} \le 1 \tag{1.34}$$

que é exactamente a desigualdade de Kraft (1.32).

Por outro lado, dado um conjunto de comprimentos $l_1, l_2, \ldots, l_M$ de palavras do código que satisfazem a desigualdade de Kraft, é sempre possível construir uma árvore semelhante à da Figura 1.3 de modo a obter um código de prefixo cujas palavras têm os comprimentos especificados.

1.3.5 1º Teorema de Shannon

Consideremos um código instantâneo, univocamente descodificável, para o qual se verifica necessariamente a desigualdade de Kraft (1.32). Consideremos ainda as quantidades

$$q_i = \xi^{-1} r^{-l_i}, \; i = 1, \ldots, M. \tag{1.35}$$

Note-se que

$$0 \quad \leq \quad q_i \quad = \quad \frac{r^{-l_i}}{\sum_{m=1}^{M} r^{-l_m}} \quad \leq \quad 1 \quad i = 1, \ldots, M$$

$$\sum_{i=1}^{M} q_i \quad = \quad \frac{\sum_{i=1}^{M} r^{-l_i}}{\sum_{m=1}^{M} r^{-l_m}} \quad = \quad 1$$

ou seja, as quantidades q_i formam uma distribuição de probabilidade. Então, sendo p_i a probabilidade de ocorrência de cada um dos símbolos do alfabeto fonte, podemos afirmar que as distribuições q_i e p_i, $i = 1, \ldots, M$, verificam a desigualdade fundamental (1.20). Tendo em conta a definição de entropia (1.13), aquela desigualdade pode ser escrita na forma

$$\sum_{i=1}^{M} p_i \log q_i + H(A) \leq 0. \tag{1.36}$$

Por outro lado, e usando (1.35) e (1.31), podemos escrever

$$\begin{aligned} \sum_{i=1}^{M} p_i \log q_i &= \sum p_i \left(-\log \xi - l_i \log r \right) \\ &= -\log \xi - \log r \sum p_i l_i \\ &= -\log \xi - \bar{L} \log r. \end{aligned} \tag{1.37}$$

Usando (1.37) em (1.36), obtém-se

$$H(A) \leq \bar{L} \log r. \tag{1.38}$$

Este resultado é independente da base da função logarítmica, pelo que se usarmos a função $\log_r$ (no caso de um alfabeto de código com r símbolos) concluímos que a entropia constitui o limiar inferior do comprimento médio de qualquer código instantâneo univocamente descodificável. Este facto, agora demonstrado formalmente, tinha já sido antecipado na sequência da discussão em torno do exemplo apresentado na Tabela 1.1. Da análise deste mesmo exemplo, verificou-se que, embora de forma não uniforme, o comprimento médio do código parecia convergir para a entropia da fonte à medida que se consideravam extensões do alfabeto de ordem crescente.

Naturalmente, a cada palavra c_i do código corresponde uma probabilidade de ocorrência $p_i = \Pr\{c_i\} = \Pr\{m_i\}$, onde m_i é um dos M símbolos do alfabeto da fonte A. Suponhamos que cada palavra c_i tem um comprimento que obedece às restrições

$$-\log_r p_i \leq l_i \leq -\log_r p_i + 1, \quad i = 1, \ldots, M, \tag{1.39}$$

garantindo-se que aos símbolos menos prováveis correspondem palavras de código mais longas. Note-se que (1.39) garante ainda que existe o código instantâneo univocamente descodificável, pois a desigualdade de Kraft é verificada. Com efeito, temos

$$r^{\log_r p_i} = p_i \geq r^{-l_i}, \quad i = 1, \ldots, M$$

e (1.32) resulta imediatamente somando de 1 a M ambos os membros da desigualdade anterior. Multiplicando por p_i todos os termos de (1.39), somando de 1 até M, e tendo em conta (1.13) e (1.31), obtém-se

$$H(A) \le \overline{L} \le H(A) + 1. \qquad (1.40)$$

Obviamente que, sendo esta desigualdade verificada para a fonte A e para o código que verifica (1.39), então

$$H(A^K) \le \overline{L}_K \le H(A^K) + 1, \qquad (1.41)$$

onde $\overline{L}_K$ é o comprimento médio do código usado para codificar os símbolos da fonte A^K. Recordando (1.30), de (1.41) obtém-se

$$H(A) \le \frac{\overline{L}_K}{K} \le H(A) + \frac{1}{K}. \qquad (1.42)$$

Este resultado demonstra que existe pelo menos um código instantâneo univocamente descodificável cujo comprimento médio $\overline{L}_K / K$ é arbitrariamente póximo da entropia da fonte A; basta notar que em (1.42) a parcela $1/K$ vai para zero quando K cresce, enquanto $\overline{L}_K / K$ é sempre uma quantidade finita. Portanto a codificação eficiente da fonte discreta sem memória obtém-se considerando extensões de ordem mais elevada. O custo da eficiência tem como contrapartida a crescente complexidade do código.

Estamos neste momento em condições de enunciar formalmente o 1º Teorema de Shannon para a codificação de fonte.

1º Teorema de Shannon
É possível codificar (e descodificar univocamente) uma fonte discreta sem memória com entropia H bit/símbolo usando um código instantâneo de comprimento médio $\overline{L}$ bit/símbolo tal que, para qualquer $\varepsilon > 0$, $\overline{L} = H + \varepsilon$. A codificação é impossível no caso em que $\overline{L} < H$.

Def. 1.7: A eficiência do código é definida por

$$\eta = \frac{H}{\overline{L}}. \qquad (1.43)$$

1.4 Código de Huffman

O código de Huffman é formado por palavras cujo comprimento médio se aproxima do limiar inferior especificado pela entropia de uma fonte discreta sem memória.

> O código de Huffman é óptimo no sentido em que, para uma dada fonte discreta sem memória, não existe outro conjunto de palavras de código com descodificação unívoca e que tenha um comprimento médio inferior.

A essência do algoritmo de codificação consiste na substituição da estatística da fonte discreta sem memória por uma outra mais simples. Este processo de redução é conduzido passo a passo até que se atinja um conjunto final das estatísticas correspondentes a apenas dois símbolos e para os quais os símbolos binários 0 e 1 são um código óptimo.

Mais concretamente, o algoritmo de codificação é o seguinte:

1. os símbolos fonte são ordenados por ordem decrescente das respectivas probabilidades de ocorrência, sendo atribuídos os bits 0 e 1 aos dois símbolos de menor probabilidade;
2. estes dois símbolos são associados *formando um novo símbolo* cuja probabilidade de ocorrência é a soma das probabilidades dos símbolos associados, reduzindo-se a lista de símbolos de uma unidade; a nova lista é reordenada por ordem decrescente das probabilidades de ocorrência;
3. os procedimentos anteriores são repetidos até que se atinja uma lista final com apenas dois símbolos aos quais são atribuídos os bits 0 e 1;
4. a palavra de código associada a cada símbolo original é construída seguindo da frente para trás a sequência de 0's e 1's que foram sendo atribuídos ao referido símbolo e respectivos sucessores.

Vamos socorrer-nos de um exemplo para perceber melhor o mecanismo do algoritmo de codificação que acabámos de descrever.

Exemplo 1.3: Na Tabela 1.3 estão listados os símbolos de uma fonte discreta sem memória e as respectivas probabilidades de ocorrência.

m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8	m_9
0.200	0.150	0.130	0.120	0.100	0.09	0.08	0.07	0.06

Tabela 1.3: Estatísticas dos símbolos de uma fonte discreta sem memória

De acordo com o passo 1. do algoritmo os símbolos estão já ordenados por ordem decrescente.

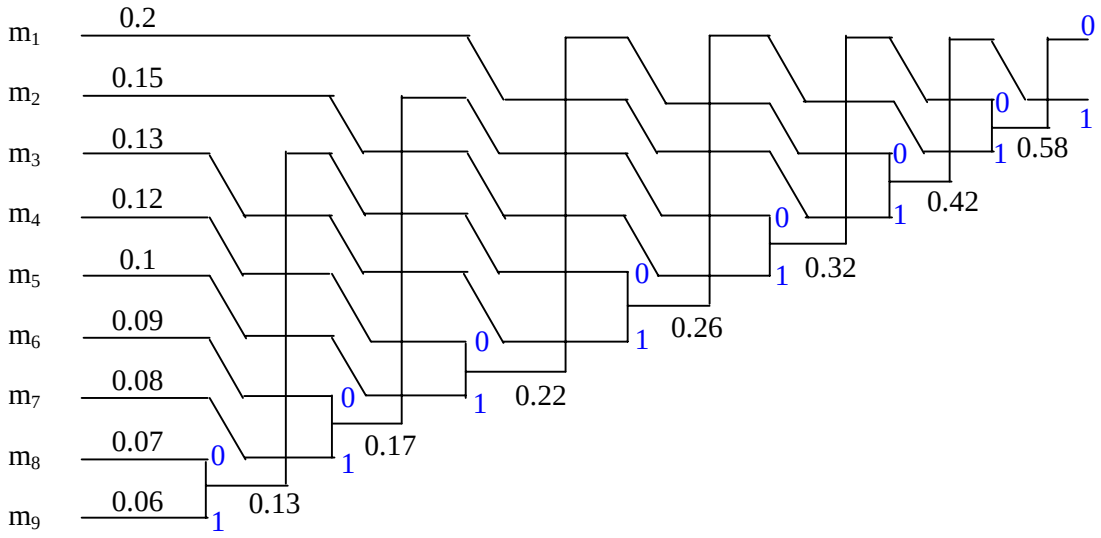

Figura 1.4: Árvore de codificação para o código de Huffman

A Figura 1.4 permite visualizar todos os passos do algoritmo de Huffman através da respectiva árvore de codificação. Note-se que por meio das sucessivas associações o número de símbolos vai-se reduzindo até se atingir o estado final com dois símbolos cujas probabilidades de ocorrência somam, como não podia deixar de ser, $0.58 + 0.42 = 1$. Os

resultados da codificação estão resumidos na Tabela 1.4. Como se pode ver, o código resultante é um código de prefixo. Por outro lado, aos símbolos menos prováveis correspondem as palavras de código mais longas. Pode também verificar-se que a entropia da fonte é $H = 3.0371\,\text{bit/símbolo}$ e que o comprimento médio do código é $\overline{L} = 3.1\,\text{bit/símbolo}$ [5].

m_i	c_i	p_i	l_i
m_1	11	0.2	2
m_2	001	0.15	3
m_3	011	0.13	3
m_4	100	0.12	3
m_5	101	0.1	3
m_6	0000	0.09	4
m_7	0001	0.08	4
m_8	0100	0.07	4
m_9	0101	0.06	4

Tabela 1.4: Resultados da codificação de Huffman

Como se vê o comprimento médio do código, embora superior, tem um valor muito próximo da entropia da fonte. Tal significa que o código obtido tem muito pouca redundância e, neste caso, constitui a representação mais eficiente da fonte original. Neste caso e de acordo com (1.43), temos $\eta = 3.0371/3.1 = 0.9797$, isto é, muito próximo dos 100%. Naturalmente, e como foi visto anteriormente, a eficiência da codificação poderia ser melhorada se considerássemos extensões da fonte de ordem superior.

Exemplo 1.4: Consideremos o alfabeto especificado na Tabela 1.5.

m_1	m_2	m_3
0.70	0.15	0.15

Tabela 1.5: Alfabeto fonte

m_i	c_i	p_i	l_i
m_1	0	0.70	1
m_2	10	0.15	2
m_3	11	0.15	2

Tabela 1.6: Resultados da codificação

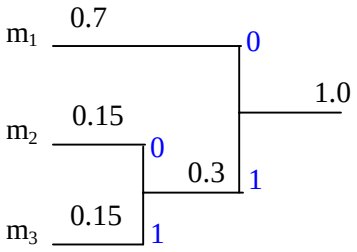

Figura 1.5: Árvore de codificação

[5] Chama-se a atenção para o facto de as unidades em que se exprimem a entropia e o comprimento médio terem significados diferentes: no primeiro caso bit significa unidade binária de informação, enquanto que no segundo a mesma designação é usada com o significado de símbolo binário.

Na Figura 1.5 está representada a árvore de codificação e na Tabela 1.6 resumem-se os resultados da codificação. Podemos calcular a entropia $H = 1.1813$, o comprimento médio do código $\overline{L} = 1.3$, e a respectiva eficiência $\eta = 0.9087$.

Consideremos agora a extensão de 2ª ordem do alfabeto original, e determinemos o correspondente código de Huffman como se ilustra na Figura 1.6. Os resultados aparecem resumidos na Tabela 1.7.

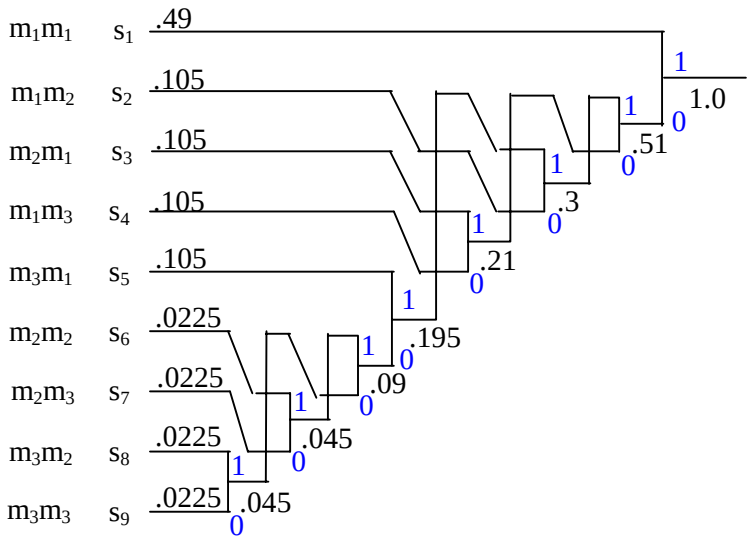

Figura 1.6: Árvore de codificação da extensão da fonte

m_i	c_i	p_i	l_i
s_1	1	0.4900	1
s_2	010	0.1050	3
s_3	001	0.1050	3
s_4	000	0.1050	3
s_5	0111	0.1050	4
s_6	011011	0.0225	6
s_7	011010	0.0225	6
s_8	011001	0.0225	6
s_9	011000	0.0225	6

Tabela 1.7: Resultados da codificação da extensão da fonte

A entropia da fonte estendida é obviamente dupla da entropia da fonte original. O comprimento médio do código anterior vale 2.395. Portanto, a eficiência deste código é $\eta = 2.3626/2.395 = 0.9865$, sendo significativamente maior do que a eficiência do código da fonte original.

Quando se usa o algoritmo de Huffman, deve na fase de reordenação ter-se o cuidado de colocar o mais acima possível o resultado da associação de dois símbolos. Consegue-se deste modo reduzir a variância dos comprimentos das palavras de código e, portanto, garantir que o

tempo gasto na descodificação das palavras de código é semelhante para todas elas. Recordemos que a variância do comprimento das palavras do código vale

$$\sigma_l^2 = \sum p_i \left(l_i - \overline{L}\right)^2. \qquad (1.44)$$

Mais uma vez vamos socorrer-nos de um exemplo para ilustrar este facto.

Exemplo 1.5: Consideremos uma fonte discreta sem memória cujo alfabeto e respectivas estatísticas se mostram na Tabela 1.8.

m_1	m_2	m_3	m_4	m_5	m_6	m_7
0.30	0.20	0.20	0.10	0.10	0.05	0.05

Tabela 1.8: Alfabeto e estatísticas da fonte

A entropia desta fonte vale $H = 2.5464$ bit/símbolo. A Tabela 1.9 apresenta os resultados obtidos quando se aplica o algoritmo de Huffman tal como nos Exemplos 1.3 e 1.4. Neste caso, o valor médio e a variância do comprimento das palavras do código valem $\overline{L} = 2.6$ bit/símbolo e $\sigma_l^2 = 0.44$.

m_i	c_i	p_i	l_i
m_1	10	0.30	2
m_2	00	0.20	2
m_3	111	0.20	3
m_4	011	0.10	3
m_5	010	0.10	3
m_6	1101	0.05	4
m_7	1100	0.05	4

Tabela 1.9: Resultados da codificação

Vejamos agora a situação em que o resultado de uma associação não é colocado o mais acima possível na tabela de probabilidades, como se mostra na Figura 1.7.

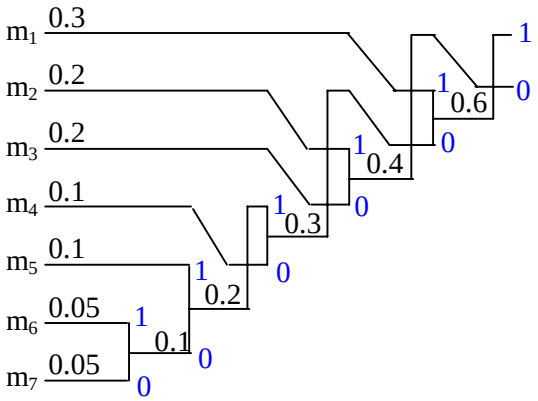

Figura 1.7: Árvore de codificação alternativa

A Tabela 1.10 resume os resultados obtidos quando o algoritmo de Huffman é aplicado do modo acima descrito.

m_i	c_i	p_i	l_i
m_1	11	0.30	2
m_2	01	0.20	2
m_3	00	0.20	2
m_4	100	0.10	3
m_5	1011	0.10	4
m_6	10101	0.05	5
m_7	10100	0.05	5

Tabela 1.10: Resultados da codificação alternativa

Verifica-se facilmente que neste caso o comprimento médio das palavras de código mantém-se, mas a variância aumenta para $\sigma_1^2 = 1.04$.

1.5 Outras Leituras Recomendadas

[1]- C.E. Shannon, "A Mathematical Theory of Communication," Collected Papers, eds. N.J.A. Sloane e Aaron D. Wyner, IEEE Press, 1993.