3
Códigos de Permutação Escalar

Neste capítulo, discutimos um quantizador vetorial estruturado conhecido como código de permutação escalar (SPC)[1]. Este código é capaz de atingir um desempenho equivalente ao do quantizador escalar com restrição de entropia (ECSQ), e possui a vantagem adicional de ser de taxa fixa.

Códigos de permutação escalar foram definidos por Slepian em 1965 [34] no contexto de modulação e, independentemente, por Dunn [12] para compressão de fontes. Subseqüentes desenvolvimentos neste contexto foram obtidos por Berger *et al.* [2], Berger [3,4], Townes *et al.* [35] e, recentemente, Goyal *et al.* (2001) [17]. O objetivo deste capítulo é revisar essa teoria.

Este capítulo está organizado da seguinte forma. A Seção 3.1 apresenta a definição de código de permutação escalar, e a Seção 3.2 apresenta um método eficiente para sua codificação ótima. O projeto de códigos de permutação escalar é abordado na Seção 3.3. Na Seção 3.4, o desempenho do SPC é discutido e comparado com o desempenho do ECSQ. A equivalência assintótica entre SPC e ECSQ é discutida na Seção 3.5.

3.1
Definição

Na literatura, os códigos de permutação escalar são definidos em duas versões: Variante I e Variante II [2]. Este trabalho lida apenas com a primeira variante desses códigos.

Definição 3.1 (Código de Permutação Escalar) Seja $\{\mu_1, \mu_2, \ldots, \mu_K\}$ um conjunto de números reais distintos tais que $\mu_1 < \mu_2 < \ldots < \mu_K$ e seja $\{n_1, \ldots, n_K\}$ um conjunto de inteiros positivos tais que

$$n = \sum_{k=1}^{K} n_k. \tag{3-1}$$

[1] De *scalar permutation code.*

Considere a seguinte *palavra de referência*, de comprimento n, formada por n_k repetições de cada símbolo μ_k em ordem crescente de índice:

$$\boldsymbol{y}_1 = (\underbrace{\mu_1, \ldots, \mu_1}_{n_1}, \underbrace{\mu_2, \ldots, \mu_2}_{n_2}, \ldots, \underbrace{\mu_K, \ldots, \mu_K}_{n_K}). \tag{3-2}$$

Um *código de permutação escalar* (SPC) é um quantizador vetorial estruturado, de taxa fixa e de dimensão n, cujo dicionário $\mathcal{C} = \{\boldsymbol{y}_i\}_{i=1}^{M}$ é formado por todas as palavras que podem ser obtidas rearranjando-se os componentes de $\boldsymbol{y}_1 = (y_{11}, \ldots, y_{1n})$ de todas as maneiras possíveis. Os conjuntos $\{\mu_k\}_{k=1}^{K}$ e $\{n_k\}_{k=1}^{K}$ são chamados, respectivamente, de *alfabeto* e *vetor de composição*.[2]

A taxa do código, em bits por amostra escalar, é dada por

$$R = \frac{1}{n} \log M \tag{3-3}$$

onde o tamanho do dicionário M é dado por

$$M = \frac{n!}{\prod_{k=1}^{K} n_k!}. \tag{3-4}$$

3.2 Codificação Ótima

Utilizando o esquema ótimo de codificação, uma palavra da fonte $\boldsymbol{X} = (X_1, \ldots, X_n)$, onde cada X_j é uma variável aleatória real, será reproduzida pela palavra de $\mathcal{C}$ que produz a menor distorção em relação a $\boldsymbol{X}$, isto é, pela palavra

$$\hat{\boldsymbol{X}} = \arg\min_{\boldsymbol{y} \in \mathcal{C}} d(\boldsymbol{X}, \boldsymbol{y}) \tag{3-5}$$

A distorção média associada ao código é portanto

$$D = E[d(\boldsymbol{X}, \hat{\boldsymbol{X}})] \tag{3-6}$$

e se a medida de distorção for de erro quadrático, teremos

$$D = \frac{1}{n} E[\|\boldsymbol{X} - \hat{\boldsymbol{X}}\|^2]. \tag{3-7}$$

[2] Pode-se também dizer que um código de permutação é formado por todas as palavras de comprimento n que possuem n_k de seus componentes iguais a μ_k, $k = 1, \ldots, K$.

Realizar a codificação descrita acima através de uma busca exaustiva entre as palavras do dicionário é, em geral, uma tarefa extremamente complexa e proibitiva quando M é muito grande (como é o caso da maioria dos códigos de permutação de interesse). Entretanto, Berger mostra em [2] que esta codificação ótima dos códigos de permutação escalar pode ser realizada através de um procedimento bastante simples. Este procedimento é válido para uma ampla classe de medidas de distorção (em particular para a distorção de erro quadrático), e é descrito a seguir.

Algoritmo 3.1 (Codificação Ótima de SPC)

1) Substitua os n_1 menores componentes de $\boldsymbol{x}$ por μ_1.

2) Substitua os subseqüentes n_2 menores componentes de $\boldsymbol{x}$ por μ_2.

$\vdots$

K) Substitua os n_K maiores componentes de $\boldsymbol{x}$ por μ_K.

Utilize a permutação de $\boldsymbol{y}_1$ resultante dessas substituições para reproduzir $\boldsymbol{x}$. ■

A identificação dos componentes de $\boldsymbol{x}$ requerida pelo algoritmo pode ser realizada através de uma única operação de ordenação que retorne os índices do vetor original. A complexidade do algoritmo é, portanto, igual a da operação de ordenação, $O(n \log n)$. Por conveniência, uma função que realiza a ordenação de números reais é definida a seguir.

Definição 3.2 (Ordenação de Escalares) A função que ordena o vetor de escalares $\boldsymbol{x}$ é definida como

$$\text{ord}(\boldsymbol{x}) = (x_{j_1}, \ldots, x_{j_n}) \tag{3-8}$$

onde os índices $j_1, \ldots, j_n \in \{1, \ldots, n\}$ são inteiros distintos tais que $x_{j_1} \leq x_{j_2} \leq \cdots \leq x_{j_n}$.

3.3 Projeto de SPC

Dados o comprimento n e o tamanho do alfabeto K, um código de permutação é completamente caracterizado por seu alfabeto $\{\mu_k\}_{k=1}^{K}$ e seu respectivo vetor de composição $\{n_k\}_{i=k}^{K}$. Nosso objetivo é escolher esses parâmetros de modo a obter um compromisso ótimo entre taxa e distorção,

o que corresponde a minimizar o lagrangiano

$$J_\lambda(\{\mu_k\},\{n_k\}) = D(\{\mu_k\},\{n_k\}) + \lambda R(\{n_k\}). \tag{3-9}$$

É importante notar que a escolha do alfabeto $\{\mu_k\}_{k=1}^K$ não afeta a taxa, a qual é completamente determinada pelo vetor de composição $\{n_k\}_{i=k}^K$, enquanto a escolha desse último afeta tanto a taxa quanto a distorção. Assim, inicialmente encontramos o alfabeto $\{\mu_k\}_{k=1}^K$ que minimiza a distorção para um vetor de composição fixo, e em seguida projetamos o vetor de composição $\{n_k\}_{i=k}^K$ de modo a obter um bom compromisso entre taxa e distorção, usando para isso o multiplicador de Lagrange λ.

3.3.1 Projeto de alfabeto

Seja $\boldsymbol{X}$ o vetor aleatório de comprimento n produzido pela fonte, e defina $(\xi_1,\ldots,\xi_n) = \text{ord}(\boldsymbol{X})$, isto é, ξ_j é a j-ésima menor componente de $\boldsymbol{X}$. As variáveis aleatórias ξ_j, $j = 1,\ldots,n$ são chamadas de *estatística ordinal* de $\boldsymbol{X}$. Usando a notação

$$S_0 = 0, \quad S_k = n_1 + n_2 + \cdots + n_k, \qquad k = 1,\ldots,K \tag{3-10}$$

e considerando a medida de distorção de erro quadrático, podemos dizer que a distorção associada a um código de permutação escalar com codificação ótima é dada por

$$D = \frac{1}{n}E[\sum_{j=1}^{n}(\xi_j - y_{1j})^2] \tag{3-11}$$

$$= \frac{1}{n}E[\sum_{k=1}^{K}\sum_{j=S_{k-1}+1}^{S_k}(\xi_j - \mu_k)^2]. \tag{3-12}$$

Desenvolvendo a expressão quadrática e notando que

$$\sum_{j=1}^{n}\xi_j^2 = \sum_{j=1}^{n}X_j^2 = \|\boldsymbol{X}\|^2 \tag{3-13}$$

podemos reescrever (3-12) na forma

$$nD = E[\|\boldsymbol{X}\|^2] - 2\sum_{k=1}^{K}\mu_k\sum_{j=S_{k-1}+1}^{S_k}E[\xi_j] + \sum_{k=1}^{K}n_k\mu_k^2. \tag{3-14}$$

Diferenciando em relação a μ_k, igualando a zero e resolvendo para μ_k, obtemos que a melhor escolha dos parâmetros $\mu_1, \ldots, \mu_K$, considerando $n_1, \ldots, n_K$ fixos, é

$$\mu_k = \frac{1}{n_k} \sum_{j=S_{k-1}+1}^{S_k} E[\xi_j], \qquad k = 1, \ldots, K. \tag{3-15}$$

Quando se utiliza os valores ótimos para μ_k dados acima, a distorção resultante é

$$D = \frac{1}{n} \left(E[\|\boldsymbol{X}\|^2] - \sum_{k=1}^{K} n_k \mu_k^2 \right). \tag{3-16}$$

3.3.2 Projeto de vetor de composição

Para simplificar a análise, assumimos que n é grande e que os parâmetros

$$p_k = \frac{n_k}{n}, \qquad k = 1, \ldots, K \tag{3-17}$$

podem variar continuamente. Assim, usando a fórmula de Stirling $\log n! \approx n \log n$, podemos aproximar a taxa R de acordo com

$$R = \frac{1}{n} \log \left(\frac{n!}{\prod_{k=1}^{K} n_k!} \right) \approx - \sum_{k=1}^{K} p_k \log p_k \tag{3-18}$$

enquanto a distorção é dada por

$$D = \frac{1}{n} E[\|\boldsymbol{X}\|^2] - \sum_{k=1}^{K} p_k \mu_k^2. \tag{3-19}$$

Adicionando em (3-9) a restrição $\sum_k p_k = 1$ através do multiplicador β, temos

$$J_{\lambda\beta} = \frac{1}{n} E[\|\boldsymbol{X}\|^2] - \sum_{k=1}^{K} p_k \mu_k^2 - \lambda \sum_{k=1}^{K} p_k \log p_k + \beta \left(\sum_{k=1}^{K} p_k - 1 \right). \tag{3-20}$$

Minimizando e resolvendo para p_k, obtemos

$$p_k = 2^{\frac{\beta - \lambda - \mu_k^2}{\lambda}} = \gamma 2^{-\frac{1}{\lambda}\mu_k^2} \qquad k = 1, \ldots, K. \tag{3-21}$$

Escolhendo o valor de β (ou, equivalentemente, o valor de γ) que minimiza (3-20), obtemos por fim

$$n_k/n = p_k = \frac{2^{-\frac{1}{\lambda}\mu_k^2}}{\sum_{\ell=1}^{K} 2^{-\frac{1}{\lambda}\mu_\ell^2}} \qquad k = 1, \ldots, K. \tag{3-22}$$

É importante mencionar que (3-22) não fornece os valores de n_k ótimos por dois motivos: $p_k n$ em geral não é inteiro, e os valores de p_k dados em (3-22) dependem dos valores de μ_k, que por sua vez dependem dos valores de n_k através de (3-15). Quando $p_k n$ é inteiro, no entanto, (3-22) é uma condição necessária para otimalidade, a qual pode ser combinada com (3-15) para produzir um algoritmo iterativo semelhante ao descrito para o projeto de quantizadores (Seção 2.3). Este algoritmo está sumarizado a seguir.

Algoritmo 3.2 (Projeto de SPC)

1) Especifique os valores de n, K e λ.
2) Inicialize $n_k \approx n/K$, $k = 1 \ldots K$.
3) Calcule $\{\mu_k\}$ através de (3-15).
4) Calcule $\{p_k\}$ através de (3-22).
5) Faça $n_k \approx p_k n$, $k = 1 \ldots K$.
6) Volte para o passo 3 até que $\{n_k\}$ permaneça inalterado em duas iterações consecutivas.
7) Fim. ■

O projeto de códigos de permutação escalar descrito nesta seção foi proposto por Berger *et al.* em 1972 [2] e permite encontrar códigos com bom desempenho de maneira eficiente. Ressaltamos que esta, porém, não é a única forma de projetar códigos de permutação escalar. Para a fonte laplaciana, por exemplo, Goyal *et al.* [17] encontram condições necessárias para otimalidade do vetor de composição que independem do alfabeto, e procedem gerando todos os códigos de permutação que satisfazem estas condições. A análise apresentada por tais autores, no entanto, limita-se a comprimentos pequenos ($n \leq 50$).

3.4 Desempenho

Nesta seção, apresentamos algumas curvas de desempenho de códigos de permutação escalar aplicados às fontes descritas na Seção 2.2 conside-

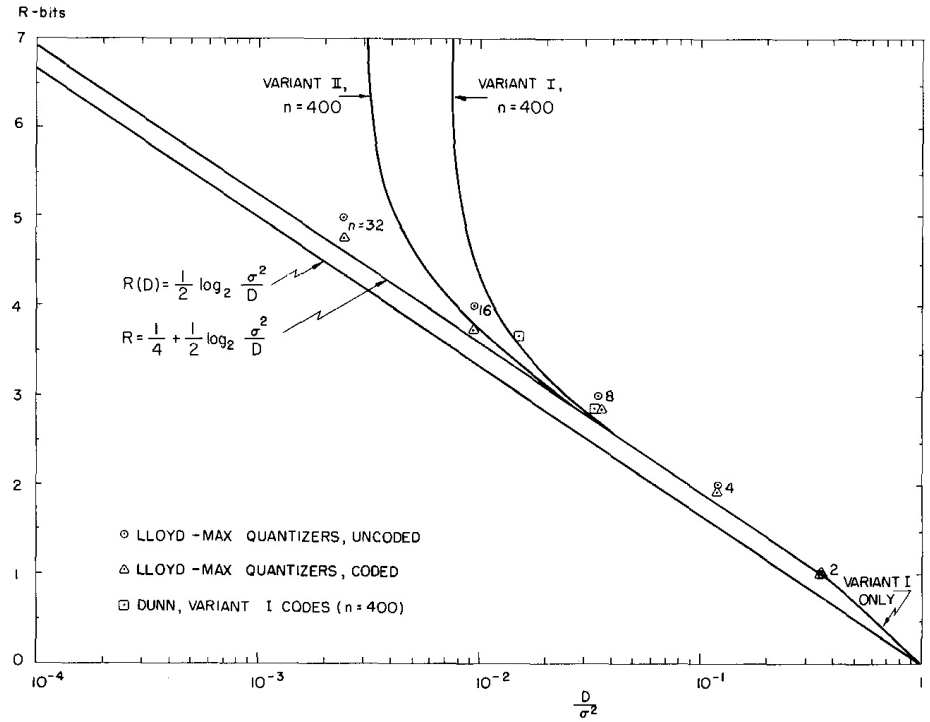

Figura 3.1: Comparação entre SPC's e o ECSQ ótimo para uma fonte gaussiana. O desempenho de outros quantizadores e a função taxa-distorção são também mostrados.

rando a medida de distorção de erro quadrático.

A Fig. 3.1 (extraída de [2]) apresenta o desempenho de SPC's com $n = 400$ projetados através do Algoritmo 3.2 e aplicados a uma fonte gaussiana sem memória. (A figura também inclui o desempenho de SPC's do tipo Variante II). São também mostradas na figura a função $R(D)$ para essa fonte e a curva de desempenho do ECSQ ótimo — sabe-se que, neste caso, tal curva situa-se exatamente um quarto de bit acima da função $R(D)$ para $R \geq 1$. Observamos que o desempenho do SPC mostra-se extremamente próximo ao do ECSQ, especialmente para taxas baixas, mas nunca superior.

A Fig. 3.2 (extraída de [17]) mostra uma comparação entre os desempenhos de SPC's Variante I e Variante II e o ECSQ ótimo para uma fonte laplaciana. Os SPC's mostrados (obtidos *cf.* discutido na seção anterior) são *ótimos* para seus respectivos comprimentos. Também nesse caso notamos que o desempenho do SPC não ultrapassa do o ECSQ.

As figuras 3.3 e 3.4 (extraídas de [17]), no entanto, mostram uma situação diferente: para uma fonte uniforme, o desempenho de um SPC é capaz de superar o do ECSQ ótimo. Estes resultados foram apresentados em [17] em 2001 e contrariam o que previamente se acreditava — que o desempenho do ECSQ ótimo não poderia ser superado pelo desempenho de um SPC.

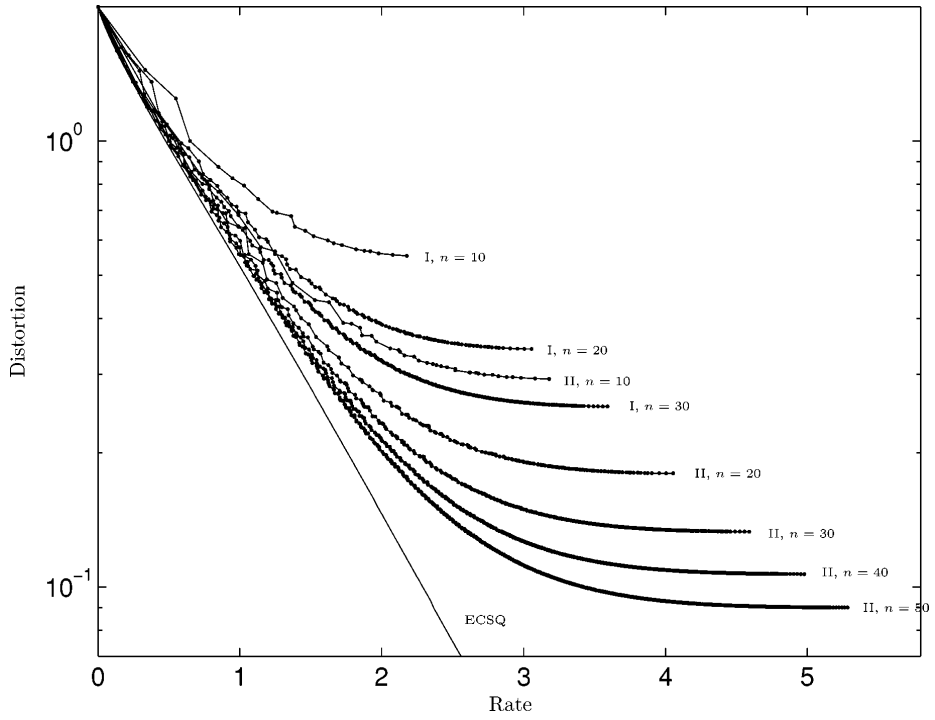

Figura 3.2: Comparação entre SPC's e o ECSQ ótimo para uma fonte laplaciana.

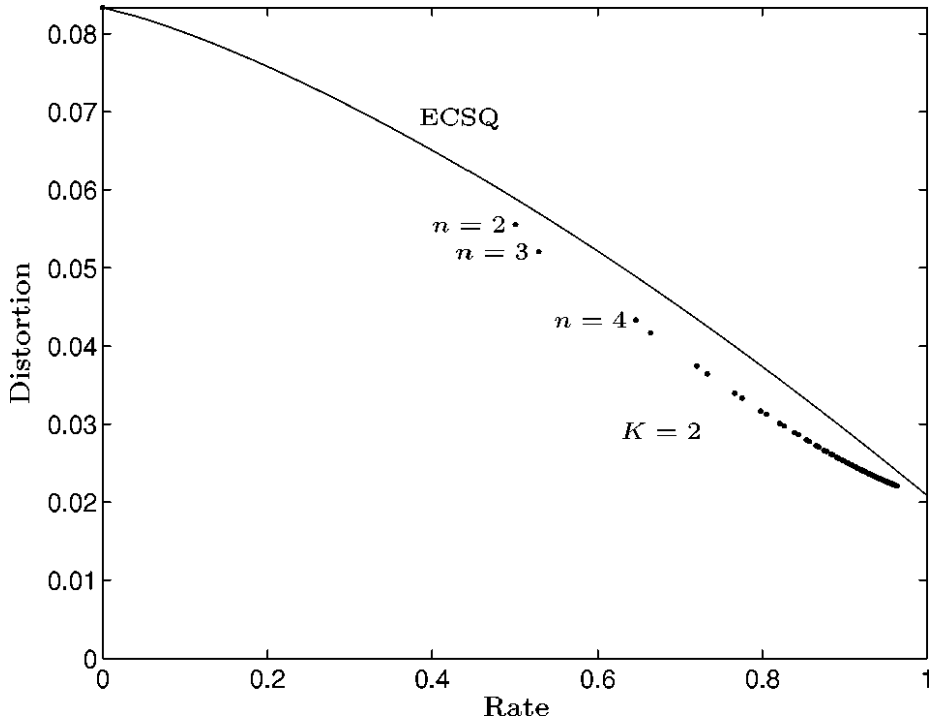

Figura 3.3: Comparação entre SPC's e o ECSQ ótimo para uma fonte uniforme.

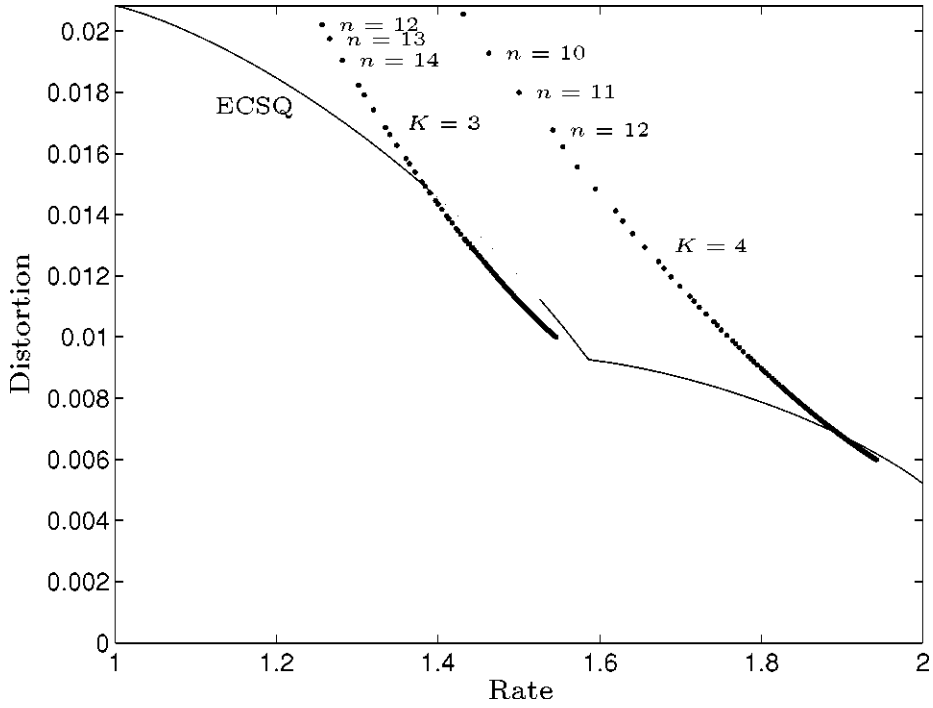

Figura 3.4: Comparação entre SPC's e o ECSQ ótimo para uma fonte uniforme.

3.5 Equivalência entre SPC e ECSQ

Ainda que os resultados para fonte uniforme nos mostrem o contrário, as figuras 3.1 e 3.2 sugerem a existência de alguma relação entre o ECSQ ótimo e um SPC com comprimento n grande. De fato, Berger provou em [3] que, quando $n \to \infty$, o SPC e o ECSQ ótimo se tornam equivalentes no sentido de que possuem exatamente o mesmo desempenho.

Para o desenvolvimento a seguir, considere um quantizador escalar caracterizado por um dicionário de reprodução $\{\mu_k\}_{k=1}^{K}$ e regiões de decisão $\mathcal{S}_k = \{x : a_{k-1} < x \leq a_k\}$, $k = 1, \ldots, K$, onde os números reais a_k são os *limiares de quantização*.[3]

Teorema 3.1 (Berger) Seja $\{X_j\}_{j=1}^{\infty}$ uma seqüência de variáveis aleatórias i.i.d. e seja X uma variável genérica desse conjunto. Dado qualquer quantizador escalar $(\{a_k\}_{k=0}^{K}, \{\mu_k\}_{k=1}^{K})$ que codifica X com taxa R e distorção D finitas, existe uma seqüência de códigos de permutação escalar $\mathcal{C}_n$ de comprimento n, $n = 1, 2, \ldots$, que codifica $(X_1, \ldots, X_n)$ com correspondentes taxas R_n distorções D_n que satisfazem ambos $\lim_{n\to\infty} R_n = R$ e $\lim_{n\to\infty} D_n = D$.

Prova. Será mostrado apenas um esboço da prova. Para todo n, faz-se o alfabeto do quantizador igual ao dicionário $\{\mu_k\}_{k=1}^{K}$ do SPC, e escolhe-se

[3]Note que geralmente se atribui $a_0 = -\infty$ e $a_K = +\infty$.

cada elemento n_k do vetor de composição $\{n_k\}_{k=1}^{K}$ arredondando np_k de forma que $\lim_{n\to\infty}\frac{n_k}{n} = p_k$, onde $p_k = P(X \in \mathcal{S}_k)$. Usando a fórmula de Stirling, mostra-se que

$$\lim_{n\to\infty} R_n = -\sum_{i=1}^{K} p_i \log p_i = R. \tag{3-23}$$

Em seguida, divide-se a distorção D_n em duas parcelas, por exemplo,

$$D_n = D + D_n^* \tag{3-24}$$

onde D_n^* é parcela adicional de distorção correspondente aos casos em que o símbolo da fonte é mapeado para valores de reconstrução diferentes pelo quantizador e pelo SPC. O desenvolvimento prossegue mostrando que, pela lei dos grandes números, $D_n^* \to 0$ quando $n \to \infty$. □

Este teorema mostra que o melhor código de permutação escalar é no mínimo tão bom quanto o melhor quantizador escalar em termos do compromisso taxa-distorção. Aliado ao fato de que os SPC's considerados na prova do Teorema 3.1 utilizam codificação ótima [3], conclui-se que é sempre possível obter um ECSQ com desempenho tão bom quanto o do melhor SPC de comprimento infinito. Combinados, estes dois fatos implicam que, quando n é infinitamente grande, o SPC ótimo e o ECSQ ótimo possuem desempenho idêntico, podendo nesse caso ser considerados *equivalentes*.

Por outro lado, ainda não se conhece um resultado geral relacionando os desempenhos do ECSQ e do SPC para um comprimento n finito. Em [3], conjecturou-se que o desempenho do SPC deve sempre melhorar com o aumento de n, o que significaria que o desempenho de um SPC com n finito seria sempre pior que o do ECSQ ótimo. Como discutido na seção anterior, esta afirmação foi refutada em [17] ao serem apresentados SPC's com desempenho melhor que o do ECSQ ótimo para uma fonte uniforme.

Finalmente, é importante mencionar que, ao contrário do ECSQ, o SPC é um quantizador de taxa fixa, não possuindo portanto as desvantagens da taxa variável discutidas na Seção 2.5. Além disso, o desempenho do ECSQ exibido nas figuras assume a utilização de codificação de entropia ideal; implementações práticas de ECSQ's apresentam sempre desempenhos piores. Por estes motivos, o SPC pode ser considerado uma alternativa promissora ao ECSQ em certas aplicações [3].

No próximo capítulo, é apresentado um novo tipo de código de permutação capaz de atingir desempenhos superiores ao do SPC e do ECSQ.