CAPÍTULO 5

CODIFICAÇÃO DE ÍNDICE A PARTIR DE CÓDIGOS REED-SOLOMON

Data de aceite: 02/06/2023

Valéria G. P. Alencar
FEEC/UNICAMP, Campinas, SP

Max H. M. Costa
FEEC/UNICAMP, Campinas, SP

RESUMO. O problema de codificação de índice sujeito a erros de transmissão foi inicialmente considerado por Dau *et al.* [5]. Neste trabalho estabelecemos uma conexão entre codificação de índice e códigos corretores de erros, por meio da construção de árvore para códigos cíclicos aninhados proposta em [3]. Implementamos algoritmos para a construção de árvore na linguagem Matlab, que ajudam a solucionar alguns problemas de implementação encontrados em [3]. Verificamos que para códigos cíclicos nem sempre haverá aumento na capacidade de correção de erros entre os níveis da árvore, motivo pelo qual restringimos este estudo, inicialmente, aos códigos Reed-Solomon, por se tratarem de códigos MDS, o que garante um aumento da distância de Hamming a cada nível. Isso significa que, sob certas condições, o conhecimento de informação lateral será interpretado como um aumento na capacidade de correção de erro pelo decodificador.

PALAVRAS-CHAVE. Codificação de Índice, Informação Lateral, Códigos Corretores de Erros, Corpos Finitos.

1 | INTRODUÇÃO

O problema clássico de codificação de índice livre de ruído, consiste de um remetente com k mensagens independentes $w_1,...,w_k$ e um canal de broadcast com múltiplos receptores, onde cada receptor demanda um subconjunto de mensagens, enquanto conhece os valores de um subconjunto diferente de mensagens como informação lateral. Sejam $R_1,...,R_n$ os n receptores e suponha que S_i representa a informação lateral e D_i a demanda do receptor R_i, onde $S_i,D_i \subset \{1,...,k\}$. O objetivo é encontrar um esquema de codificação, chamado codificação de índice, que satisfaça a demanda de todos os receptores e use um número mínimo de transmissões.

Consideramos o caso específico de codificação de índice para canais de broadcast discretos com ruído, em que

todos os receptores demandam todas as mensagens da fonte, ou seja, $S_i, D_i \subset \{1,...,k\}$. Diante disso, surge a possibilidade de se projetar códigos corretores de erros cujo mapeamento das mensagens em palavras códigos é tal que o decodificador pode aumentar a distância de Hamming em um receptor que tem conhecimento prévio dos valores de algum subconjunto das mensagens como informações laterais.

Estamos supondo que o transmissor desconhece o subconjunto de mensagens já conhecido no receptor e executa uma codificação de modo que qualquer informação lateral possível possa ser usada de forma eficiente no decodificador. A noção de múltipla interpretação foi introduzida em [9], mostrando que quanto maior a quantidade de informação lateral disponível no receptor, maior será a capacidade de correção de erro na decodificação. Os códigos construídos também devem ser códigos de correção de erros para codificação de índice quando o receptor não possui informações laterais, ou seja, quando $D_i = \emptyset$.

A codificação de índice aqui apresentada é dada pela construção de árvore para códigos cíclicos aninhados proposta em [3]. Nos restringimos aos códigos Reed-Solomon por se tratarem de códigos MDS (máxima distância de separação), o que garante um aumento da distância de Hamming a cada nível da árvore. Isto significa que, sob certas condições, o conhecimento de informação lateral será interpretado como um aumento na capacidade de correção de erro pelo decodificador.

2 | CONCEITOS PRELIMINARES

2.1 Codificação de Índice

O objetivo da codificação de índice é realizar uma codificação conjunta de mensagens, visando atender simultaneamente a demanda de todos os receptores, enquanto transmite a mensagem resultante na taxa mais alta possível. Favor consultar [2] para uma abordagem aprofundada sobre codificação de índice.

A seguir apresentamos o modelo através de um exemplo. Considere o sistema de comunicação sem fio, mostrado na Figura 1. O receptor R_i está interessado na mensagem w_i, $i \in \{1,2,3,\}$, e possui algumas das outras mensagens como informações laterais. Em particular, o receptor 1 tem a mensagem w_3 como informação lateral, o receptor 2 tem w_1 e w_3 e o receptor 3 tem w_1 e w_2. O servidor deseja enviar as mensagens aos receptores usando o menor número possível de transmissões.

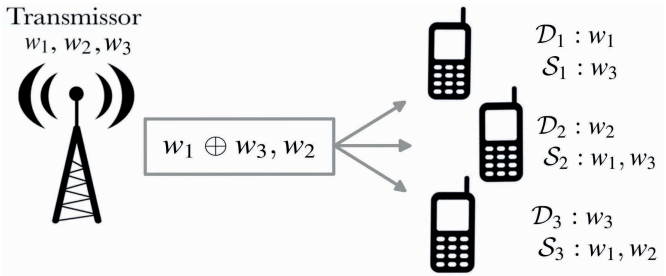

Figura 1: Codificação de índice com três receptores.

Considerando um canal livre de ruídos, o servidor poderia comunicar todas as mensagens enviando uma por vez, fazendo três transmissões. Alternativamente, se o servidor transmitir duas mensagens codificadas $w_1 \oplus w_3$ e w_2, cada receptor pode recuperar sua mensagem desejada usando as mensagens codificadas recebidas e as informações laterais disponíveis, como vemos abaixo:

$$\begin{aligned}
&\text{Receptor 1: } (w1 \oplus w3) \oplus w3 = w1\\
&\text{Receptor 2: } (w1 \oplus w3) \oplus w2 \oplus (w1 \oplus w3) = w2\\
&\text{Receptor 3: } (w1 \oplus w3) \oplus w2 \oplus (w1 \oplus w2) = w3
\end{aligned}$$

2.2 Construção de árvore de códigos cíclicos aninhados

A construção de árvore de códigos cíclicos aninhados se propõe a:

i) Codificar de maneira independente diferentes pacotes de dados, fornecendo proteção contra erros do canal;

ii) Adicionar os pacotes codificados através de operações polinomiais e enviar o pacote resultante C_0;

iii) Corrigir os erros sobre C_0 e, por fim, recuperar os dados no receptor através de operações polinomiais.

2.2.1 Códigos cíclicos aninhados

Um código aninhado pode ser caracterizado por um código global onde cada elemento é dado por uma soma de palavras-código, cada uma pertencendo a um subcódigo diferente. Isso é,

$$c = i_1 G_1 \oplus i_2 G_2 \oplus \cdots \oplus i_N G_N$$

onde $\oplus$ representa a soma módulo 2, a palavra código $i_t G_t$ pertence a um subcódigo C_t de C e $c \in C$.

$$\mathcal{C} = \mathcal{C}_1 + \mathcal{C}_2 + \cdots + \mathcal{C}_N$$

Consideramos uma classe mais restrita dos códigos aninhados, a classe dos códigos cíclicos aninhados, na qual os subcódigos são gerados por polinômios. Podemos então

definir os códigos cíclicos aninhados, de maneira mais formal, como se segue:

Seja $\mathcal{C} = \{C(x) \in F_q[x]; g(x)|C(x)\}$ um *t*-código (um código capaz de corrigir *t* erros) corretor de erro cíclico, onde *g*(*x*) é o polinômio gerador.

$\mathcal{C} = \langle g(x) \rangle$ é um ideal de R_n, mas também é um subespaço vetorial, e desse modo, podemos escrever:

$$C(x) = p_1(x)g_1(x) + p_2(x)g_2(x) + \cdots + p_N(x)g_N(x)$$

onde $C_\ell(x) = p_\ell(x)g_\ell(x), 1 \leq \ell \leq N$, é um pacote codificado pertencente ao t_ℓ-subcódigo corretor de erro

$$\mathcal{C}_\ell = \{C_\ell(x) \in F_q[x]; g_\ell(x)|C_\ell(x)\}$$

gerado por gℓ(x) e satisfazendo as condições:

1) $g_\ell(x)|g_{\ell+1}(x)$;

2) $deg[C_\ell(x)] < deg[g_{\ell+1}(x)]$.

2.2.2 O método de construção de árvore

Considere uma árvore na qual o nó raiz está associado ao subespaço vetorial de um código corretor de erro abrangente.

Defina o nó raiz da árvore como sendo o código C tal que:

$$\mathcal{C}_{i0} = \langle g_{i0}(x) \rangle = \{C_{i0}(x) \in F_q[x]; g_{i0}(x)|C_{i0}(x)\}.$$

Este subespaço corresponde a um t_0-código corretor de erro cíclico, dado por $C_{i0}(n,k_{i0})$gerado pelo polinômio $g_{i0}(X)$.

Uma árvore de códigos cíclicos aninhados é uma árvore finita que satisfaz as condições:

1) Cada nó interno pode ser subdividido em outro nó interno e um terminal;

2) O *j*-ésimo nó interno está associado com um subespaço linear $C_{ij} \subset F_q^n$ de dimensão k_{ij} e pode ser subdividido da forma:

$$\mathcal{C}_{ij} = \mathcal{C}_{i(j+1)} + \mathcal{C}_{t(j+1)}, com \ \mathcal{C}_{i(j+1)} \cap \mathcal{C}_{t(j+1)} = \{0\}$$

$$\text{Se } k_{i(j+1)} = dim\mathcal{C}_{i(j+1)} \text{ e } k_{t(j+1)} = dim\mathcal{C}_{t(j+1)}, \text{ então } k_{ij} = k_{i(j+1)} + k_{t(j+1)}.$$

3) Todos os subespaços associados com os nós internos devem ser códigos de blocos lineares cíclicos definidos por um polinômio gerador;

4) Se $\mathcal{C}_{ij} = \langle g_{ij}(x) \rangle$e $\mathcal{C}_{i(j+1)} = \langle g_{i(j+1)}(x) \rangle$. Então $g_{ij}(x)|g_{i(j+1)}(x)$. Além disso, $g_{ij}(x)|x^n - 1$ para qualquer $g_{ij}(x)$;

5) Para encerrar, o último nó interno não terá ramificações.

Seja *pj*(*x*) o pacote de dados associado ao *j*-ésimo nó terminal, para $1 \leq j \leq T$. A codificação é dada por:

$$C_j(x) = p_j(x)g_{i(j-1)}(x)$$

Os pacotes codificados são somados e a palavra código resultante é enviada.

$$C_0(x) = C_1(x) + C_2(x) + \cdots + C_T(x)$$

Favor consultar [3] para maiores detalhes sobre o processo de codificação utilizando a construção de árvore.

3 I CONSTRUÇÃO ÁRVORE: ALGORITMO E CONSIDERAÇÕES

Os programas (m- files) em *MatLab®* para a construção de árvore, que podem ser encontrados em [1], permitem efetuar os cálculos em corpos finitos fazendo as devidas transformações da representação inteira para potências de α, baseadas na Tabela 1.

Exemplo 1. *Para T=3 seja $C_{i0}(7,5)$ um código de Reed-Solomon em GF(8) e $k_{t1}=k_{t2}=2$ as dimensões dos subespaços C_{t1},C_{t2}, respectivamente. O último nó associado a C_{i2} de dimensão $k_{i2}=1$. Os pacotes $p_1(x)=x+\alpha^2$, $p_2(x)=\alpha^3x+\alpha$ estão associados aos nósterminais e ambos tem comprimento 2, o pacote $p_3(x)=\alpha^5$ está associado ao último nó e tem comprimento 1.*

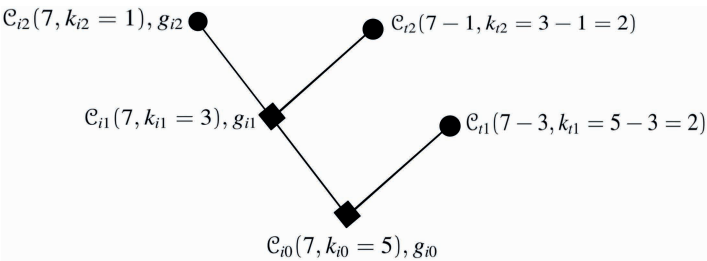

Figura 2: Construção de Árvore.

Seja α o elemento primitivo de GF(8), então os polinômios geradores são:

1) $deg(g_{i0}(x)) = n - k_{i0} = 2 \Rightarrow g_{i0}(x) = \prod_{j=1}^{2}(x - \alpha^j) = x^2 + \alpha^4 x + \alpha^3.$

2) $deg(g_{i1}(x)) = n - k_{i1} = 4 \Rightarrow g_{i1}(x) = \prod_{j=1}^{4}(x - \alpha^j) = x^4 + \alpha^3 x^3 + x^2 + \alpha x + \alpha^3.$

3) $deg(g_{i2}(x)) = n - k_{i2} = 6 \Rightarrow g_{i2}(x) = \prod_{j=1}^{6}(x - \alpha^j) = x^6 + x^5 + x^4 + x^3 + x^2 + x + 1.$

Considere os pacotes $p_1(x) = x + \alpha^2$, $p_2(x) = \alpha^3 x + \alpha$, $p_3(x) = \alpha^5$, ou seja, $p_1 = [1, 4]$, $p_2 = [3, 2]$ e $p_3 = [7]$, de acordo com a Tabela 1. Codificando os pacotes, temos:

$$\begin{aligned} C_1(x) &= p_1(x)g_{i0}(x) \\ &= x^3 + \alpha x^2 + \alpha^4 x + \alpha^5 \end{aligned}$$

$$\begin{aligned} C_2(x) &= p_2(x)g_{i1}(x) \\ &= \alpha^3 x^5 + \alpha^5 x^4 + \alpha^6 x^3 + \alpha^2 x^2 + x + \alpha^4 \end{aligned}$$

$$C_3(x) = p_3(x)g_{i2}(x)$$
$$= \alpha^5x^6 + \alpha^5x^5 + \alpha^5x^4 + \alpha^5x^3 + \alpha^5x^2 + \alpha^5x + \alpha^5$$

Então, a palavra código a ser transmitida é dada por:

$$C_0(x) = C_1(x) + C_2(x) + C_3(x)$$
$$= \alpha^5x^6 + \alpha^2x^5 + 0x^4 + \alpha^3x^3 + 1x^2 + 0x + \alpha^4$$

Potência de α	Elemento de GF(8)	Binária	Inteira
0	0	000	0
1	1	001	1
α	x	010	2
α^2	x^2	100	4
α^3	x+1	011	3
α^4	x^2+x	110	6
α^5	x^2+x+1	111	7
α^6	x^2+1	101	5

Tabela 1: Construção do Corpo GF(8).

Considerando a construção de árvore baseada em códigos Reed-Solomon e supondo que o receptor tenha informa¸c˜ao lateral disponível, quando haverá ganho na capacidade de correção?

Proposição 1. *Devido `a estrutura de aninhamento, a característica de capacidade de correção de erro variável só pode ser observada se houver uma remoção sequencial dos pacotes associados aos nós, da raiz para o topo da árvore.*

Demonstração. Supondo que $C_\ell(x), 1 \le \ell \le T$, é o primeiro pacote codificado conhecido no receptor, então

$$C_0(x) = p_1(x)g_{i0}(x) + \cdots + p_{(\ell-1)}(x)g_{i(\ell-2)}(x) + p_{(\ell+1)}(x)g_{i\ell}(x) + \cdots + p_T(x)g_{i(T-1)}(x)$$
$$= [p_1(x) + \cdots + p_{(\ell-1)}(x)q_{(\ell-1)}(x) + p_{(\ell+1)}(x)q_{(\ell+1)}(x) + \cdots + p_T(x)q_T(x)]g_{i0}(x)$$

portanto, $C_0(x) \in C_{i0}(n, k_{i0})$, cuja capacidade de correção de erro é t_0. Note que mesmo o receptor conhecendo outros pacotes $C_j(x), \ell < j \le T$, o resultado não se altera. Por outro lado, se todos os pacotes $C_j(x), 1 \le j < \ell$, são conhecidos pelo receptor, podemos escrever

$$C_0(x) = p_{(\ell+1)}(x)g_{i\ell}(x) + \cdots + p_T(x)g_{i(T-1)}(x)$$
$$= [p_{(\ell+1)}(x)\overline{q}_{(\ell+1)}(x) + \cdots + p_T(x)\overline{q}_T(x)]g_{i\ell}(x)$$

assim, $C_0(x) \in \mathcal{C}_{i\ell}(n, k_{i\ell})$, cuja capacidade de correção de erro é $t_\ell \ge t_0$, ocorrendo a igualdade somente quando $d_{min}(C_\ell) - d_{min}(C_0) < 2$.

Analisamos dois casos de construção de árvore de códigos cíclicos, com os mesmos parâmetros a cada nível. Em um deles não se observa aumento na capacidade de correção

de erro do penúltimo nó interno para o último nó da árvore. Isso se dá pela variedade de possibilidades de polinômios geradores para um código cíclico de parâmetros (n, k). Em seguida, mostramos que para códigos de Reed-Solomon essa característica de aumento da capacidade será garantida desde que

$$k_{ij} - k_{i(j+1)} \geq 2, \forall j = 0, \dots, T-1.$$

Exemplo 2. *Seja $C_{i0}(15,10)$ um código cíclico em GF(2) e $k_{t1}=4$, $k_{t2}=2$ as dimensões dos subespaços C_{t1}, C_{t2}, respectivamente. O último nó está associado com C_{i2} de dimensão $k_{i2}=4$.*

Considere a fatoração: $x^{15}-1=(1+x)(1+x^3+x^4)(1+x+x^2+x^3+x^4)(1+x+x^2)(1+x+x^4)$

Caso 1. Considere os polinômios geradores:

- $g_{i0}(x) = (1+x)(1+x^3+x^4)$. Então, $d_{min}(C_{i0}) = 4$ e $t_0 = 1$;
- $g_{i1}(x) = g_{i0}(x)(1+x+x^2+x^3+x^4), d_{min}(C_{i1}) = 6 e t_1 = 2$;
- $g_{i2}(x) = g_{i1}(x)(1+x+x^2), d_{min}(C_{i2}) = 8$ e $t_2 = 3$

Note que houve aumento da capacidade de correção de erro a cada nível da árvore, o que não ocorre no caso seguinte.

Caso 2. Considere agora os seguintes polinômios geradores:

- $g_{i0}(x) = (1+x)(1+x+x^4)$. Logo, $d_{min}(C_{i0}) = 4$ e $t_0 = 1$.
- $g_{i1}(x) = g_{i0}(x)(1+x^3+x^4), d_{min}(C_{i1}) = 6$ e $t_1 = 2$.
- $g_{i2}(x) = g_{i1}(x)(1+x+x^2), d_{min}(C_{i2}) = 6$ e $t_2 = 2$.

Proposição 2. *Dado um código de Reed-Solomon de parâmetros (n,k), o qual tem distância mínima d=n-k+1 é possível garantir um aumento da capacidade de* correção *de erro a cada nível da árvore desde que* $k_{ij} - k_{i(j+1)} \geq 2, \forall j = 0, \dots, T-1$.

Demonstração: Devemos provar que $t_{i(j+1)} \geq t_{ij} + 1, \forall j = 0, \dots, T-1$. Sem perda de generalidade, fixe *j*=0. Se $k_{i0} - k_{i1} \geq 2,$ então podemos escrever:

$$(-d_{i0} + n + 1) + d_{i1} - n - 1 \geq 2$$

$$d_{i1} - 1 \geq d_{i0} - 1 + 2$$

$$\left[\frac{d_{i1}-1}{2}\right] \geq \left[\frac{d_{i0}-1}{2}\right] + 1$$

$$t_{i1} \geq t_{i0} + 1$$

4 | CONCLUSÕES

Este trabalho considera a codificação de índice a partir da construção de códigos cíclicos aninhados. Após a etapa de correção de erros no decodificador, ocorre a etapa de recuperação de dados, que foi descrita em [3] da seguinte forma:

O *j*–´ésimo pacote $p_j(x)$, para $j < T$, é decodificado pela operação:

$$p_j(x) = \frac{[C_0(x) mod g_{ij}(x)]}{g_{i(j-1)}} \, eparaj = T, p_T(x) = \frac{C_0(x)}{g_{i(T-1)}}$$

Note que a operação de módulo elimina a influência de todas as mensagens relacionadas a polinômios de grau igual ou superior ao grau de *gij*, portanto, a informação desejada estará contida no resto de uma operação de divisão, que é o efeito da operação de módulo. O quociente da última operação de divisão fornecerá a informação desejada, pois todas as outras mensagens tem grau inferior ao grau do polinômio divisor.

A constatação de que para códigos cíclicos nem sempre haverá aumento na capacidade de correção de erro entre os níveis da árvore, como considerado em [3], nos leva a buscar respostas sobre como escolher adequadamente os polinômios geradores para um código de parâmetros (*n, k*) e seus subcódigos, de modo a garantir subcódigos com maior distância de Hamming, ao ponto de se observar aumento da capacidade de correção de erro entre os níveis da árvore. Um método para construir cadeias de alguns códigos de bloco lineares, mantendo as distâncias mínimas (dos subcódigos gerados) a maior possível é apresentado em [8] e pode ser a resposta para essa busca.

REFERÊNCIAS

1. Alencar, V. G. P, Construcão-de-Arvore em MatLab no GitHub (2021). https://github.com/valeriaurca/Construção-de-Arvore.git.

2. Arbabjolfaei, F. and Kim, Y. H. (2018), Fundamentals of Index Coding, *Foundations and Trends@ in Communications and Information Theory*, 14:163–346, 2018. DOI: 10.1561/0100000094.

3. Barbosa, F. C. and Costa, M. H. M. A tree construction method of nested cyclic codes, *IEEE Information Theory Workshop*, 302-305, 2011. DOI: 10.1109/ITW.2011.6089441.

4. Blahut, R.E. *Algebraic Codes for Data Transmission* . Cambridge University Press, New York, 2003.

5. Dau, S. H., Skachek, V. and Chee, Y. M. Error Correction for Index Coding with Side Information, *IEEE Transactions on Information Theory* , 59:1517–1531, 2013. DOI: 10.1109/TIT.2012.2227674.

6. Heegard, C. Partitioned linear block codes for computer memory with “stuck-at” defects, *IEEE Transactions on Information Theory* , 29:831–842, 1983. DOI:10.1109/TIT.1983.1056763.

7. Hefez, A., Villela, M. L. T. *Códigos Corretores de Erros, 2a. edição* . IMPA, Rio de Janeiro, 2017.

8. Vinck, A. J. H. , Luo, Y. Optimum distance profiles of linear block codes, *IEEE International Symposium on Information Theory*, 1958–1962, 2008. DOI: 10.1109/ISIT.2008.4595331.

9. Xiao, L., T. Fuja, J. Kliewer and D. J. Jr. Costello. Nested codes with multiple interpretations, *2006 40th Annual Conference on Information Sciences and Systems*, 851-856, 2006. DOI: 10.1109/CISS.2006.286586.