Relatório 2 - ELE-32
Códigos de Bloco Cíclicos e BCH

Aloysio Galvão Lopes
Departamento de
Engenharia da Computação
Instituto Tecnológico de Aeronáutica
São José dos Campos, São Paulo
Email: aloysiogl@gmail.com

Vitor Pimenta dos Reis Arruda
Departamento de
Engenharia da Computação
Instituto Tecnológico de Aeronáutica
São José dos Campos, São Paulo
Email: vitor_pimenta97@hotmail.com

***Resumo*—Este relatório avalia o desempenho, sobre um canal binário simétrico, de códigos cíclicos com bloco pequeno e de códigos BCH primitivos.**

Para tanto, implementaram-se um canal BSC e codificadores/decodificadores para os códigos mencionados. A taxa de erro de bit de informação foi avaliada estatisticamente para eles, submetendo-os a várias palavras aleatórias para estimar seus desempenhos.

I. Introdução sobre códigos cíclicos

O códigos cíclicos são um caso especial dos códigos de bloco lineares. Tais códigos têm como propriedade fundamental que uma rotação de uma palavra código gera uma outra palavra código. Essa restrição adiciona informação à estrutura do código, permitindo, assim, que se decodifique a informação transmitida com menor necessidade de se armazenar informações.

É conveniente utilizar a representação polinomial de códigos, a qual é descrita com mais detalhes em [2]. Será adotada a convenção seguinte: serão enviados n bits de código para cada k bits de informação.

Para codificar uma palavra código basta multiplicar esta palavra por um polinômio gerador g, o qual é um múltiplo de $1 + D^n$ conforme [2]. A decodificação é o processo inverso: a divisão. O resto da divisão é chamado síndrome e será diferente de zero caso haja erro de transmissão. A característica de ciclicidade será utilizada na decodificação. Uma vez que o código é cíclico, pode-se obter todas as síndromes associadas às rotações de um erro a partir da síndrome de uma única rotação. Será possível, dessa maneira, decodificar palavras sem associar erros a síndromes, reduzindo a quantidade de informação armazenada.

II. Introdução sobre BCH

Códigos cíclicos BCH primitivos são completamente caracterizados por dois parâmetros: m e t, ambos naturais, de modo que se usará a notação $BCH(m, t)$ para os individualizar. m é tal que $n = 2^m - 1$ é o tamanho de bloco do código e t é a distância mínima planejada (necessariamente menor ou igual à distância mínima real).

O polinômio gerador de $BCH(m, t)$ é obtido pelo mínimo múltiplo comum entre os polinômios mínimos de $\alpha^1, ..., \alpha^{2t}$, em que α é um elemento primitivo de $GF(2^m)$.

A decodificação desses códigos, diferentemente de outros cíclicos e não-cíclicos, não exige que se memorizem associações síndrome-erro e é computacionalmente eficiente, apresentando complexidade $\mathcal{O}(n)$ confirmada em IV-B1.

O algoritmo utilizado foi retirado de [1] e é conhecido como "Berlekamp-Massey".

III. Algoritmos para códigos cíclicos

A algoritmo desenvolvido pode ser dividido nas seguintes etapas: obtenção dos polinômios geradores, codificação, e decodificação. Cada uma dessas etapas esta descrita nas subseções seguintes.

A. Obtenção dos polinômios geradores

Foram escolhidos, inicialmente, cinco tipos de códigos deferentes com duplas (k, n) mostradas a seguir contidas em $S = \{(6, 10), (7, 12), (8, 14), (9, 15), (9, 16)\}$. Para cada elemento de S foram gerados todos os polinômios geradores g_{ij} em que i é o índice em S e j o índice de um polinômio qualquer que gere um código S_i. Com este fim, foi utilizada a função *cyclpoly* do MATLAB.

Para cada i dentre todos os j escolhe-se o polinômio g_{ij} que gera o código de maior distância mínima. Para isso, geraram-se todas as palavras informação e calculou-se o peso de cada palavra código a fim de achar a distância mínima. A complexidade desta etapa é $\mathcal{O}(2^k n^{3.37})$. Considerou-se o pior caso da multiplicação de matriz $\mathcal{O}(n^{2.37})$, uma vez que para o cálculo da palavra código pela palavra informação utilizou-se a multiplicação matricial. O algoritmo descrito foi implementado na linguagem *Python*.

A principal dificuldade desta parte do código foi achar a distância mínima, no entanto, muito se pôde aproveitar da atividade anterior. Dessa forma a dificuldade desta etapa não foi tão grande.

B. Codificação

A codificação consistiu simplesmente de uma multiplicação de polinômios, a qual foi feita em *Python* utilizando a função *polymul* da biblioteca *numpy*. Dessa forma a complexidade de codificação do sistema ficou $\mathcal{O}(n^2)$ para uma palavra código de tamanho n.

Vale ressaltar que o canal utilizado foi exatamente o mesmo do experimento anterior, por isso, sua implementação não será detalhada neste documento.

C. Decodificação

Esta foi a parte mais desafiadora do experimento por envolver a correção de erros. A decodificação consiste em uma divisão polinomial da palavra código pelo polinômio gerador, considerando que, se houver resto (síndrome), deve ser feita correção de erros.

Vale ressaltar que o código foi implementado em *Python* e que a divisão de polinômios foi implementada seguindo o algoritmo de divisão manual com soma resto 2. A complexidade final do algoritmo de divisão polinomial ficou $\mathcal{O}(n^2)$.

Para corrigir os erros foi implementado o algoritmo sugerido no roteiro do laboratório, que realiza rotações para ser capaz de utilizar unicamente síndromes associadas a erros na primeira posição da palavra código. O pseudocódigo para tal algoritmo é mostrado em Algoritmo 1.

Algoritmo 1 Decodificação

```
procedure decode(message)
    sind ← calc_sindrome(message)
    while sind ≠ all_zeros do
        if sind ∈ end_one_sinds then
            change_first(message)
            sind ← calc_sindrome(message)
        else
            rotate(message, −1)
            rotate(sind, −1)
            if sind.first = 1 then
                sind.first ← 0
                sind ← sind + g
            end if
        end if
    end while
    message ← unrotate(message)
return divide(message, g)
end procedure
```

A parte mais desafiadora da atividade foi encontrar o conjunto de síndromes, representado no algoritmo por end_one_sinds. Consta em Algoritmo 2 o código utilizado para encontrar um conjunto de síndromes associadas a erros que começam em 1 de maneira a garantir que o código de Algoritmo 1 termine em ao menos n ciclos do laço mais externo.

A complexidade de decodificação do sistema é $\mathcal{O}(n^2h)$, em que h é o tamanho do conjunto de síndromes.

IV. Resultados

A. Códigos cíclicos não-BCH

1) Polinômios geradores e distâncias mínimas: Os polinômios geradores utilizados e suas respectivas distâncias mínimas de código associado são mostradas em Listagem 1. Os índices são os mesmos do conjunto S apresentado junto

Algoritmo 2 Decodificação

```
procedure update_sindromes()
    sind_map ← ∅
    for i ∈ {1, len(codeword)} do
        errs ← list(errors, weigth = i)
        for err ∈ errs do
            err_key ← calc_sindrome(err)
            if err_key = all_zeros then
                continue
            end if
            if err_key ∉ sind_map ∪
weigth(sind_map[err_key]) = i ∩
error_ends_with_zero(sind_map[err_key]) then
                sind_map[err_key] ← (err.first, i)
            end if
        end for
    end for
    for key ∈ sind_map do
        if  error_ends_with_one(sind_map[err_key])
then
            end_one_sinds.add(key)
        end if
    end for
end procedure
```

ao algoritmo. As distâncias mínimas, com mesma indexação de S, foram $\{2, 4, 3, 4, 2\}$.

$$
\begin{aligned}
p_1 &= D^4 + D^3 + D^2 + D + 1 \\
p_2 &= D^5 + D^3 + D^2 + 1 \\
p_3 &= D^6 + D^2 + 1 \\
p_4 &= D^6 + D^4 + D^3 + D^2 + 1 \\
p_5 &= D^7 + D^6 + D^5 + D^4 + D^3 + D^2 + D + 1 \qquad (1)
\end{aligned}
$$

2) Desempenho dos códigos cíclicos: Um gráfico comparativo entre os códigos aqui desenvolvidos e o código de Hamming é mostrado na Figura 1

A título de comparação, a Figura 2 traz os resultados do experimento anterior, o qual empregou códigos não necessariamente cíclicos obtidos pela tentativa de maximizar a distância mínima por meio de incrementos no tamanho de bloco.

B. Códigos BCH

1) Complexidade de decodificação BCH: Segundo [1], a decodificação de códigos BCH se dá em $\mathcal{O}(n)$, n sendo o tamanho de bloco empregado. Para confirmar a validade dessa afirmação, construíram-se códigos $BCH(m, 3)$, com $m \in \{4, 5, ..., 20\}$, e mediram-se os tempos de decodificação submetendo a mesma palavra a cada um deles, com resultados ilustrados pela Figura 3.

2) Desempenho do BCH: Para comparar o código BCH aos outros cíclicos, procurou-se um BCH de taxa próxima a 4/7 (usada como referência nos outros códigos deste laboratório), elegendo-se $BCH(5, 3)$ para ser submetido a testes

sob canal BSC. A escolha se deu com base no polinômio gerador calculado, $g(X) = X^{15} + X^{11} + X^{10} + X^9 + X^8 + X^7 + X^5 + X^3 + X^2 + X + 1$, que rendeu ao código taxa $\frac{31-15}{31} \simeq 0.52 \simeq \frac{4}{7} \simeq 0.57$.

Somente pela superior distância mínima projetada (3), era de se esperar que $BCH(5,3)$ superasse todos os outros códigos descritos neste texto. Isso é corroborado pela Figura 4, na qual se vê que ele supera $Hamming(4,7)$ (que, por sua vez, tinha o mesmo desempenho aproximado dos outros códigos cíclicos).

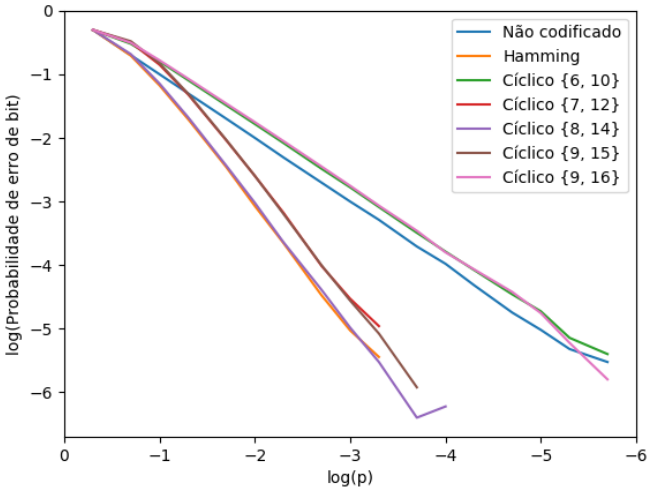

Figura 1: Comparação da eficácia dos códigos cíclicos para 5004720 bits de informação enviados.

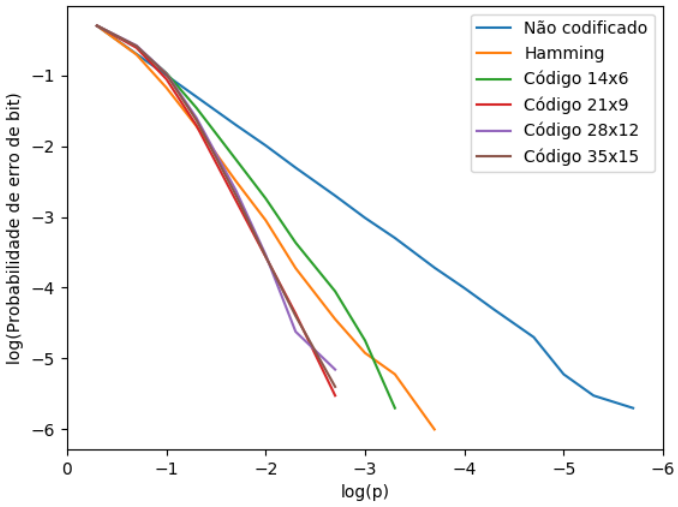

Figura 2: Comparação da eficácia dos códigos não necessariamente cíclicos para 1000080 bits de informação enviados.

V. Conclusão

O código cíclico apresenta a vantagem de ser necessário armazenar uma quantidade menor de síndromes e de não ser necessário associar síndrome a erro na decodificação. Isso introduz maior complexidade no passo de decodificação pois não há mapeamento direto entre síndrome e erro. No entanto, no código desenvolvido pelas equipes, foi necessário criar um método para geração de códigos. Dessa forma a complexidade do desenvolvimento do código cíclico é maior no tocante ao entendimento dos conceitos envolvidos e no código do experimento anterior no tocante à criação de um procedimento para a geração da matriz de paridade.

Pôde-se notar do gráfico presente em Figura 1 que o código de Hamming é mais eficiente que quase todos os códigos implementados, com exceção do BCH. Isso era esperado uma vez que nenhum dos outros códigos cíclicos corrige mais bits que Hamming e Hamming possui bloco menor que os

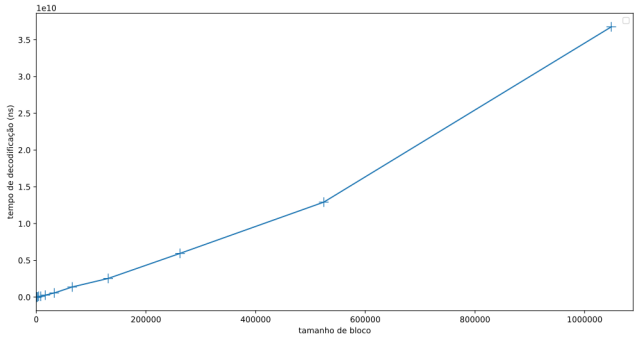

Figura 3: Tempo de decodificação para BCH com variados tamanhos de bloco.

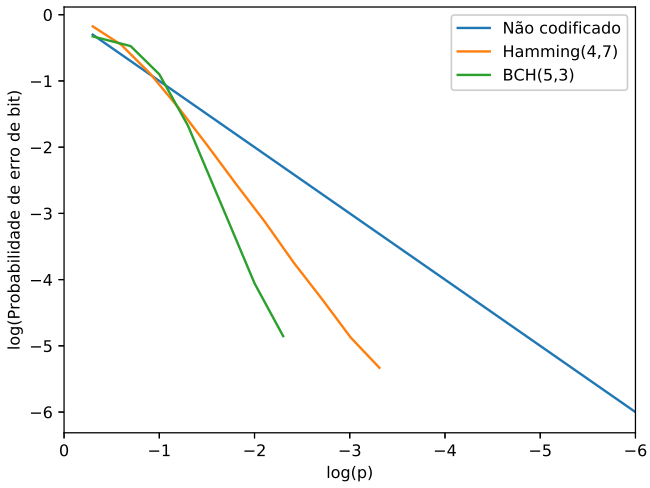

Figura 4: Chances de erro de bit de informação para $BCH(5,3)$ e $Hamming(4,7)$ sujeitos a canais BSC de variadas taxas de erro de bit.

demais. Nota-se que os casos de distância mínima dois são ligeiramente piores que o canal não codificado e que o caso de distância mínima três é muito próximo de Hamming.

Pode-se depreender que os códigos cíclicos, da maneira como foram desenvolvidos, são ineficazes na diminuição da distância mínima com o aumento do tamanho do bloco. Em contraste a isso, os códigos do experimento anterior apresentados na Figura 2 são, em sua maioria, mais eficazes que Hamming.

Os tamanhos de códigos cíclicos utilizados variaram de 10 ao maior tamanho requisitado (16) e o método utilizado para gerar o conjunto de síndromes é extensível para tamanhos maiores de códigos. Vale notar que a etapa crítica de geração do conjunto de síndromes só precisa ser realizada uma única vez no preprocessamento, por isso, seu tempo de execução total não é crítico.

A medida entre tamanho do bloco e desempenho é dada pela complexidade da multiplicação e divisão polinomial, nesse caso foram consideradas ambas $\mathcal{O}(n^2)$. Além disso, foram discutidos mais detalhes sobre as complexidades das implementações na explicação do algoritmo.

Por outro lado, o código BCH possui rica estrutura matemática subjacente (corpos), possibilitando completa flexibilidade para gerar tamanhos de bloco arbitrariamente grandes e distâncias mínimas desejadas. Além de ser robusto por construção, ainda se caracteriza por complexidade linear de decodificação, como foi empiricamente validado.

REFERÊNCIAS

[1] Berlekamp, Elwyn R. *Algebraic Coding Theory*. Edição revisada. Singapura: WSPC, 2015.

[2] Sharma, Manish *Aula2 - Códigos Cíclicos*. ELE32-Introdução a Comunicações. 2018.