Código de Blocos

Carlos Renato de Andrade Figueiredo
Divisão de Ciências da Computação
Instituto Tecnólogico de Aeronáutica
São José dos Campos, SP 12228–900
Email: carlos.figueiredo@ga.ita.br

Samara Ribeiro Silva
Divisão de Ciências da Computação
Instituto Tecnólogico de Aeronáutica
São José dos Campos, SP 12228–900
Email: samara.silva@ga.ita.br

***Resumo* - Este relatório descreve os resultados da simulação e comparação entre o código de Hamming(7,4) e um código proposto pelos alunos com taxa semelhante, mas com palavras código maiores. Comparativamente com uma simulação realizada, foi possível observar no gráfico que o código proposto pelos alunos obteve desempenho semelhante, porém levemente inferior. Foi observada a relação entre o desempenho e o tamanho da palavra código e da percentagem p. O motivo dessa relação se dá devido a capacidade de detecção e correção de erros dos códigos implementados.**

***Palavras-chave* - Hamming, código de correção de erros, código de bloco**

I. Introdução

A transmissão de mensagens compostas por uma sequência de bits pode ser comprometida por imprecisões no canal de transmissão. Esses erros, em um canal simétrico (um canal onde a probabilidade de troca (erro) de um bit de informação independe do bit transmitido), podem ser minimizados com a utilização de codificação da mensagem.

Essa codificação pode ocorrer de diversas maneiras e podem identificar e corrigir um número determinado de erros de acordo com o método utilizado. Por exemplo, o código de Hamming possui a capacidade de detectar e corrigir um erro simples, ou seja, não consegue identificar erros em dois bits distintos em uma palavra-código transmitida.

Neste relatório serão abordados a codificação de Hamming e um código proposto pelos alunos autores deste trabalho.

II. Código de Hamming

O código de Hamming foi implementado conforme a teoria apresentada em aula. Foi utilizado **u** como a palavra de informação, **v** como a palavra código gerada através de **v = uG**, $\mathbf{G_H}$ como a matriz geradora para o código Hamming, $\mathbf{H_H}^\mathbf{T}$ como a matriz verificadora de paridade, **r** como o vetor recebido **r = v + e**, **e** como o vetor de erros e **s** como a sindrome identificada $\mathbf{s = rH_H}^\mathbf{T}$.

Ao desenvolver a expressão $\mathbf{s = rH_H}^\mathbf{T}$ pode-se observar que a síndrome depende apenas do erro:

$$\mathbf{s = rH_H}^\mathbf{T}$$

$$\mathbf{s = (v + e)H_H}^\mathbf{T}$$

$$\mathbf{s = 0 + eH_H}^\mathbf{T}$$

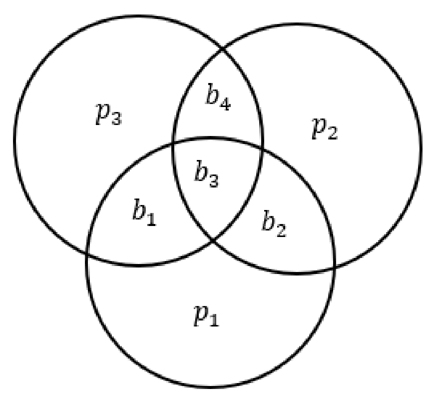
$$\mathbf{s = eH_H}^\mathbf{T}$$

Como diferentes combinações de erros podem acarretar a mesma síndrome assume-se que é mais provável que ocorra apenas um erro de transmissão. Portanto, identificando a síndrome é possível determinar um erro **e'** e executar a correção do bit indicado. Logo, se a síndrome identificada for a primeira linha de $\mathbf{H_H}^\mathbf{T}$ $\mathbf{s = H_H}^\mathbf{T}[1]$ o erro **e'** indicará que a correção necessária é no bit $\mathbf{r}[1]$.

Foi implementada a codificação para uma palavra de informação de tamanho 4 e palavra código de tamanho 7, obtendo assim uma taxa de $t = \frac{4}{7} \cong 0,57$.

Para $\mathbf{u} = \begin{bmatrix} b_1 & b_2 & b_3 & b_4 \end{bmatrix}$ a codificação foi realizada utilizando o esquema da figura 1 e matriz geradora correspondente.

Fig. 1. Código de Hamming



$$G_\mathrm{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

E a para decodificar foi utilizada a matrix verificadora de paridade $\mathbf{H_H}^\mathbf{T}$.

$$\mathbf{H_H}^\mathbf{T} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

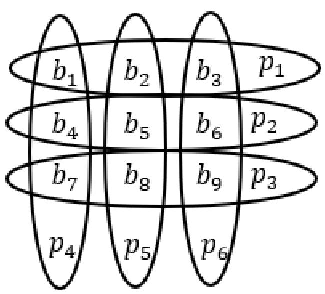
III. Código Proposto pelos Alunos

O objetivo deste laboratório é propor um código com taxa semelhante ao código de Hamming do item anterior e que tivesse um bom desempenho para palavras de informação maiores que 4.

O código proposto pelos alunos segue o mesmo princípio de codificação através da matriz geradora e decodificação e identificação de sindrome pela matriz verificadora de paridade.

Para isso observou-se que era necessário que cada bit de informação $\mathrm{b_n}$ fosse processado por pelo menos 2 bits de verificação $\mathrm{p_k}$. Na figura 2 pode-se observar como foi calculado cada bit de verificação $\mathrm{p_k}$, com palavras de informação de tamanho 9 ($\mathrm{Alunos_9}$).

Fig. 2. Código de Proposto para info = 9



Observe que cada bit de informação participa de dois grupo de bits distintos e isso é essencial para o bom funcionamento do código.

A matriz geradora para esse código é:

$$G_9 = \begin{bmatrix} I_9 & P_9 \end{bmatrix}$$

Onde $\mathrm{I_9}$ é a matrix identidade e $\mathrm{P_9}$ é:

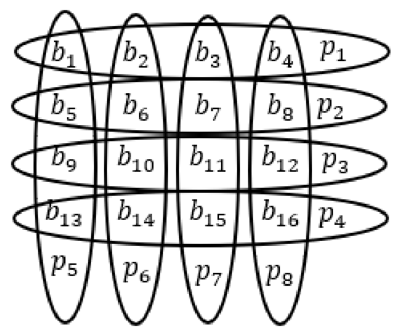
$$P_9 = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

E a para decodificar foi utilizada a matrix verificadora de paridade $\mathbf{H_9}^\mathbf{T}$.

$$\mathbf{H_9}^\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Foi implementado o codificador acima e utilizando a mesma lógica também foi implementado para palavra de informação de tamanho 15 ($\mathrm{Alunos_{15}}$) e 16 ($\mathrm{Alunos_{16}}$). Para $\mathrm{Alunos_{15}}$ os bits $\mathrm{p_k}$ foram calculados de maneira semelhante a figura 2 formando 5 grupos somando 3 bits e 3 grupos somando 5 bits, já para $\mathrm{Alunos_{16}}$ optou-se por realizar a codificação conforme a figura 3.

Fig. 3. Código de Proposto para info = 16



Vale ressaltar que com os códigos apresentados acima obteve-se taxas semelhante ao do código de Hamming $t_9 = \frac{9}{15} \cong 0,60$, $t_{15} = \frac{15}{23} \cong 0,65$ e $t_{16} = \frac{16}{24} \cong 0,67$.

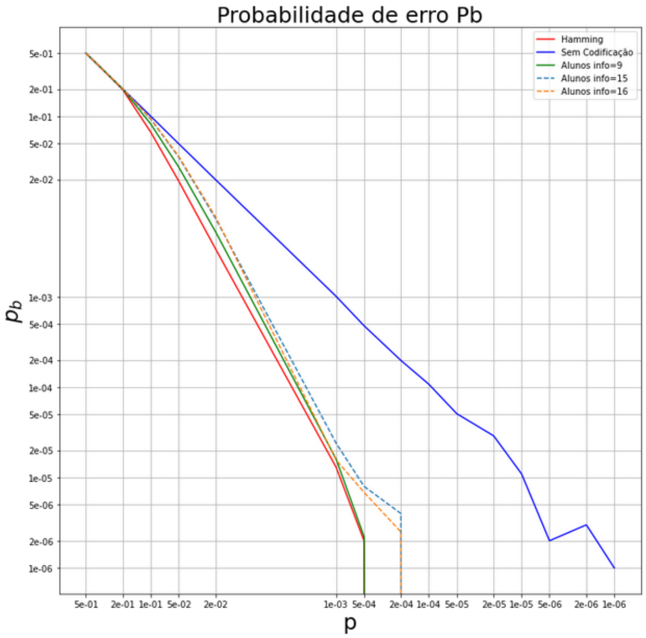
IV. Resultados

Foi realizada uma bateria de testes para verificar o funcionamento dos codificadores e decodificadores. Inicialmente, foram realizadas 250.000 transmissões de palavras código de tamanho correspondente ao codificador utilizado para três casos: código de Hamming, a transmissão sem código e $\mathrm{Alunos_9}$. Após isso, realizaram-se 100.000 transmissões para mais dois casos: $\mathrm{Alunos_{15}}$ e $\mathrm{Alunos_{16}}$.

A implementação do código e as simulações foram feitas através da plataforma Google Colab e estão contidos no link: https://bit.ly/2PMxa9N.

Os Resultados obtidos podem ser observados na figura 4. Analisando o gráfico dessa figura foi possível observar que

Fig. 4. Gráfico da Probabilidade de erro de bit de informação



a curva azul corresponde a transmissão sem codificação e possui perfil semelhante a uma reta diferindo apenas para probabilidades p muito pequenas. Esse resultado está em conformidade com a teoria pois não houve nenhuma tentativa de correção de possíveis erros.

Vale ressaltar que o código proposto pelos alunos mesmo com palavra de informação até 4 vezes maior obteve resultado semelhante ao do código Hamming, porém levemente inferior. Note, também, que o código Alunos_9 obteve um melhor resultado entre os testes propostos pelos alunos.

V. Questionamentos

1) Qual foi a maior dificuldade de implementar o decodificador para o código de Hamming?

Como o algoritmo do código de Hamming já estava pronto e foi amplamente discutido em aula, não houve grandes dificuldades em sua implementação. O ponto mais delicado foi a correção do erro após a identificação da síndrome.

2) Qual foi o método utilizado para encontrar o código maior? Este método extensível para qualquer tamanho de bloco?

O método utilizado é dividir os bits em grupos de maneira que cada bit participe de pelo menos dois grupos com integrantes distintos. Conforme as figuras 2 e 3. Na figura 2 o bit b_1 participa dos grupos (b_1, b_2, b_3) e (b_1, b_4, b_7) note que a intersecção desses dois grupo é apenas o b_1. Assim será possível observar e corrigir um possível erro do b_1.

Esse método pode ser extendido para palavras de informação maiores, com alteração apenas na quantidade de participantes dos grupos conforme foi possível observar nos códigos Alunos_{15} e Alunos_{16}

3) Qual é a relação medida entre o tamanho do bloco e o desempenho?

Foi possível observar que o resultado dos testes foram melhores para palavras de informação menores isso ocorre devido os códigos implementados permitirem a detecção e correção de apenas um bit.

4) Qual é a complexidade de sua codificação e decodificação do seu sistema?

A complexidade do codigo implementado é dada pela complexidade da multiplicação das matrizes que é $O(n^{2.37})$ [2] e [3]. Vale ressaltar que foi utilizada a biblioteca Numpy para realizar as operações de multiplicação de matrizes.

VI. Conclusão

Este trabalho serviu para que os alunos pudessem observar a importância do uso de um metódo de codificação para a transmissão de palavras de informação com maior precisão.

Foi observada que a medida que a probabilidade p diminui o desempenho dos codificadores aumentam. A relação entre o desempenho e o tamanho da palavra de informação a ser transmitida também foi constatada e isso ocorre devido a capacidade de correção de erros limitada dos metódos implementados. Portanto, faz-se necessário o estudo de metódos com maior capacidade de detecção e correção de erros.

References

[1] E1 - Códigos de Blocos. ELE-32 - Introdução a Comunicações. Divisão de Enfenharia Eletrônica. Instituto Tecnológico de Aeronáutica. São José dos Campos.

[2] NUMPY. numpy.matmul - NumPy v1.20 Manual. Disponível em: https://numpy.org/doc/stable/reference/generated/numpy.matmul.html. Acesso em 24 de março de 2021.

[3] COPPERSMITH, D.; WINOGRAD, S. Matrix multiplication via arithmetic progressions. Journal of Symbolic Computation, 1990.