Códigos de Bloco

Felipe Mourad Pereira
femp1999@gmail.com

Rodrigo Tanaka Aki
krodrigo71008@gmail.com

I. Resumo

O presente relatório tem a finalidade de reportar as atividades realizadas pelos autores durante o primeiro laboratório de ELE-32. A atividade consistiu de compreender, reproduzir e analisar a codificação e decodificação de sinais binários, com a presença de ruído no canal de comunicação.

II. Introdução

Em diversas aplicações, tem-se que a transmissão de informação é feita por uma sequência de bits. Mas, devido a imperfeições do canal, a recepção dessa informação pode possuir erros, com a troca de alguns bits. Caso a probabilidade de troca desses bits independa de seu valor, trata-se de um canal simétrico binário (*BSC*) [1].

Assim, com a intenção de corrigir possíveis erros do sinal recebido, pode-se dividir o sinal em blocos de bits e adicionar bits auxiliares que servirão para conferir a validade dos bits enviados.

Definimos a taxa do código como a razão (quantidade de bits de informação)/(quantidade de bits enviados), sendo que os bits enviados abrangem tanto os bits de informação como os bits extras para conferência.

Por exemplo, na codificação de Hamming, a cada 4 bits de informação, enviam-se 3 bits extras, chamados bits de paridade, que servirão para identificar eventuais erros nos bits enviados. Assim, sua taxa é de 4/7 $\approx 57\%$ [2].

Um bit de paridade consiste em um bit que garante que o número de 1s em certas regiões do código seja par ou ímpar. Neste laboratório, foi usado o bit de paridade par, ou seja, a soma módulo 2 de alguns bits da mensagem a ser enviada [3]. Por exemplo, para o código de Hamming, temos o seguinte bloco de bits a ser enviado a cada 4 bits de mensagem: $d_1d_2d_3d_4p_1p_2p_3$, sendo $d_{(1-4)}$ a mensagem, e $p_{(1-3)}$ os bits de paridade. Esses bits de paridade podem ser definidos pelas seguintes operações:

$$p_1 = d_1 \oplus d_2 \oplus d_3 \tag{1}$$

$$p_2 = d_1 \oplus d_3 \oplus d_4 \tag{2}$$

$$p_3 = d_1 \oplus d_2 \oplus d_4 \tag{3}$$

Podemos obter a mensagem a ser enviada a partir da mensagem original e da seguinte matriz geradora (G) [4]:

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} \tag{4}$$

Seja u uma mensagem a ser enviada. A mensagem transmitida v é então dada por:

$$\mathbf{v} = \mathbf{u} \cdot G \tag{5}$$

E podemos verificar a validade da mensagem recebida $\mathbf{r}$ por meio da matriz de verificação de paridade H, que para Hamming é dada por [5]:

$$H^{\mathrm{T}} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

Por definição, temos que:

$$\mathbf{v} \cdot H^{\mathrm{T}} = 0 \tag{7}$$

Assim, dada uma mensagem recebida $\mathbf{r}$, se $\mathbf{r} \cdot H^{\mathrm{T}} \neq 0$, podemos perceber que houve um erro na transmissão do sinal.

Com o código de Hamming, conseguimos identificar até uma troca em cada bloco de bits. Por exemplo, se a mensagem recebida for $r = 0010000$, temos que $r \cdot H^{\mathrm{T}} = [110]$. Chamaremos tal resultado obtido de síndrome. Há diversas trocas de bits possíveis que originariam tal resultado, entratanto a mais provável é uma no 3º bit (d_3), pois a probabilidade de apenas uma mudança ocorrer é menor que duas (assumindo que a probabilidade de erro seja menor do que 0.5), logo é a melhor escolha para o erro e adotado. Pode-se corrigir a mensagem recebida realizando a operação $r \oplus e = v \oplus e \oplus e = v$.

Assim, pode-se mapear as síndromes obtidas para uma tabela de possíveis erros de 1 bit que gerariam tal síndrome. Existe a probabilidade de tal erro não ser de somente um bit, fazendo a mensagem recebida ser corrigida de modo errado e podendo até aumentar o número de bits trocados numa tentativa de corrigi-laa, contudo percebe-se que para baixas probabilidades de troca de bit, o código de Hamming possui uma performance muito satisfatória na correção das mensagens recebidas.

III. Metodologia

Inicialmente, definimos qual seria a quantidade e a taxa a ser implementada. Por sugestão do professor, utilizamos 8 bits de mensagem e 6 bits de paridade, valores dobrados em relação ao código de Hamming. Assim, mantemos a taxa idêntica, e a mensagem codificada será $d_1d_2d_3d_4d_5d_6d_7d_8p_1p_2p_3p_4p_5p_6$.

Em seguida, definiu-se quem seria a matriz G, definindo os bits de paridade. Com inspiração no código de Hamming (diagrama de Venn), definimos os bits de paridade da seguinta maneira:

- O bit d_1 estará presente em todos os bits de paridade.
- Em cada bit de paridade, não constará um dado bit d_i, $2 \leq i \leq 7$. Por exemplo, o bit d_2 não constará apenas no bit p_6.
- O bit d_8 não constará nos dois últimos bits de paridade.

Assim, chegamos nas equações:

$$p_1 = d_1 \oplus d_2 \oplus d_3 \oplus d_4 \oplus d_5 \oplus d_6 \oplus d_8 \tag{8}$$

$$p_2 = d_1 \oplus d_2 \oplus d_3 \oplus d_4 \oplus d_5 \oplus d_7 \oplus d_8 \tag{9}$$

$$p_3 = d_1 \oplus d_2 \oplus d_3 \oplus d_4 \oplus d_6 \oplus d_7 \oplus d_8 \tag{10}$$

$$p_4 = d_1 \oplus d_2 \oplus d_3 \oplus d_5 \oplus d_6 \oplus d_7 \oplus d_8 \tag{11}$$

$$p_5 = d_1 \oplus d_2 \oplus d_4 \oplus d_5 \oplus d_6 \oplus d_7 \tag{12}$$

$$p_6 = d_1 \oplus d_3 \oplus d_4 \oplus d_5 \oplus d_6 \oplus d_7 \tag{13}$$

E na matriz G:

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

A matriz H correspondente é a seguinte:

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

Com o código de Hamming, é trivial identificar quais padrões de erros, dentre os que geram uma determinada síndrome, têm maior probabilidade de ocorrer. Porém, na nossa codificação, há alguns problemas. Para algumas síndromes, há mais de um padrão de erro gerador com mesma probabilidade de ocorrer e há 6 bits de síndrome, ou seja, 64 síndromes diferentes, que exigiriam um tempo muito grande para se analisar individualmente.

Para resolver esse problema, foi desenvolvido um programa que calcula o padrão de erro que tem maior probabilidade de gerar uma dada síndrome. Por questão de eficiência, para cada síndrome, foi pré-computado o padrão correspondente. Isso foi feito por meio de tentativa e erro, começando dos com menor quantidade de 1s, ou seja, em ordem decrescente de probabilidade de ocorrência. Assim, o primeiro associado a uma síndrome sempre será um dos mais prováveis. Nos casos em que há mais de um padrão com a mesma probabilidade, o primeiro é escolhido, pois é impossível saber qual é o correto.

Desse modo, foi gerada uma tabela associando cada síndrome a seu padrão de erro mais provável. Assim, para calcular a taxa dos códigos estudados e desenvolvidos, geramos 1 milhão de bits aleatoriamente, cada um igualmente provável de ser 0 ou 1. Então, simulamos a codificação, transmissão com ruído e decodificação desses bits e verificamos o número de bits trocados entre a mensagem original e a decodificada. A transmissão foi implementada considerando o canal como um BSC e a simulação foi feita para diversos valores da probabilidade de troca de bit do canal. Por fim, foi plotada a taxa de erro do código de Hamming, do nosso código e a taxa esperada sem correção alguma.

IV. Resultados e discussões

A seguir, o gráfico obtido por meio de nosso experimento.

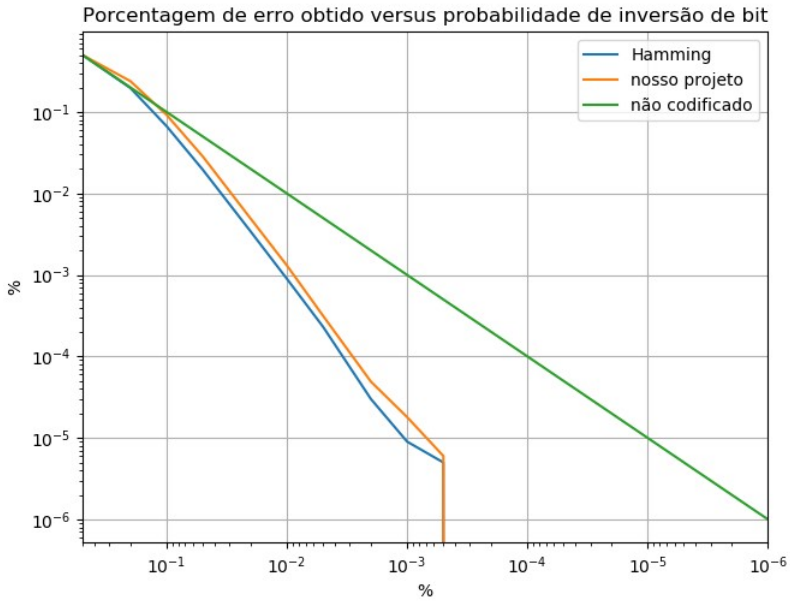

Fig. 1. Resultados obtidos.

Em que, a partir de p = $5 \cdot 10^{-4}$, tanto para Hamming como para a nossa codificação, o erro vai a 0. O gráfico a seguir representa um zoom na região antes de p = $5 \cdot 10^{-4}$:

Podemos perceber que a nossa codificação apresenta inicialmente um erro maior que a própria ausência de codificação, uma vez que para altos valores de *p*, na tentativa de corrigir síndromes que foram mapeadas como se fossem de 1 bit, o algoritmo inverte bits antes corretos, sendo que muito

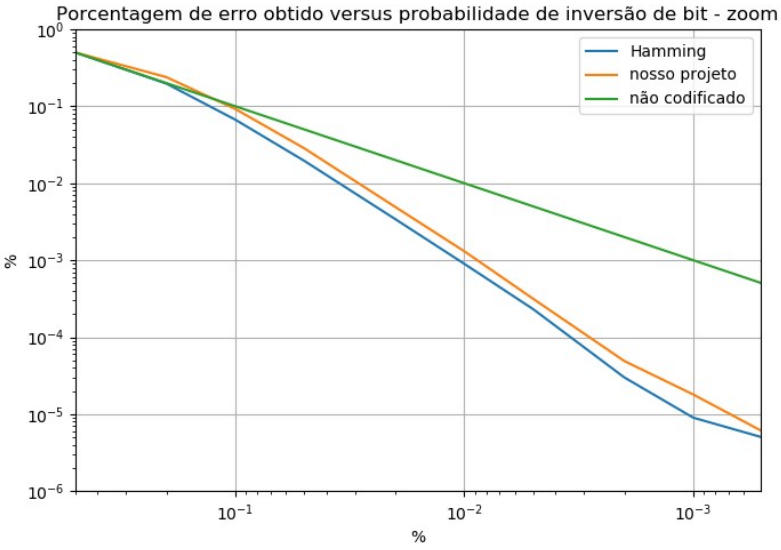

Fig. 2. Resultados obtidos com zoom em $p < 5 \cdot 10^{-4}$.

provavelmente a síndrome seria de mais de um bit (uma vez que p é grande).

Entretanto, a medida que p diminui, o algoritmo se torna satisfatório, mas não suficiente para ser melhor que a codificação de Hamming, embora muito próximo. Assim, percebemos margem para aprimoramento do experimento. Ao corrigirmos apenas erros de 1 bit (14 síndromes diferentes), utilizamos apenas 14 dos 64 mapeamentos de síndrome possíveis, deixando 50 mapeamentos (erros de 2 ou mais bits) a serem definidos de maneira arbitrária, o que acreditamos que poderia ser melhorado. Mesmo assim, acreditamos que o resultado atingido foi satisfatório, aproximando-se razoavelmente à eficiência da codificação de Hamming.

A seguir, algumas perguntas sugeridas que permitem explicar o experimento:

1. Qual foi a maior dificuldade de implementar o decodificador para o cóodigo de Hamming?

A maior dificuldade foi mapear as síndromes para quais padrões de erro representavam, mas não foi tão complicado quanto para o nosso código, pois foram poucos casos.

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

Como era desejado manter a taxa do código, apenas foi dobrado o número de bits de informação e os auxiliares. Depois, similarmente ao Hamming, foi definido um bit de informação para entrar na soma de todos os bits auxiliares, seis outros para participar de todos exceto um, e o restante não participar da soma de dois bits de paridade. O método pode ser extendido e terá desempenho razoável para até um erro por bloco de código, mas sua performance será significantemente pior para dois ou mais erros. Basta setar um bit de informação para entrar em todos os bits de paridade, outros para não entrarem em um, os restantes não devem entrar em dois, os restantes em três, assim por diante.

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

Para um bloco maior, o desempenho foi pior, o que pode ser justificado pela maior probabilidade de ocorrerem ao menos dois erros em um bloco, algo com que o nosso algoritmo não lida satisfatoriamente.

4. Qual a complexidade de codificação e de decodificação de seu sistema?

Tanto a codificação quanto a decodificação são lineares, ou seja, O(n) [6] em relação ao tamanho da mensagem, pois a codificação é apenas multiplicar partes da entrada por uma matriz, assim para n grupos, o número de multiplicações será n vezes maior do que para um grupo e, para um vetor de tamanho fixo, a sua multiplicação por uma matriz é feita em O(1). A decodificação, por sua vez, é O(n) pois cada mensagem codificada recebida (um grupo) é multiplicada por uma matriz, gerando um padrão de erros, esse padrão é usado para computar uma correção olhando em uma tabela (assim levando apenas O(1) de complexidade de tempo) e essa correção é somada com módulo 2 à mensagem decodificada, o que para um vetor de tamanho fixo também é feito em O(1).

V. Conclusão

Com as atividades realizadas, foi possível perceber a alta eficiência da codificação de Hamming, e a dificuldade de implementar uma codificação que consiga superá-la em termos de correção de ruídos de transmissão. Uma das principais razões se dá pelo fato de, ao utilizar um bloco maior, temos a maior probabilidade de ocorrer duas inversões de bits dentro do bloco, o que não é previsto pelo nosso algoritmo, capaz de corrigir apenas inversões de 1 bit.

Por fim, foi possível abstrair, compreender e assimilar como podemos realizar codificação, tentativa e correção e decodificação de sinais binários, o que julgamos ser muito útil e importante no aprendizado de telecomunicações.

References

[1] https://en.wikipedia.org/wiki/Binary_symmetric_channel
[2] https://en.wikipedia.org/wiki/Hamming_code
[3] https://en.wikipedia.org/wiki/Parity_bit
[4] https://en.wikipedia.org/wiki/Generator_matrix
[5] https://en.wikipedia.org/wiki/Parity-check_matrix
[6] https://en.wikipedia.org/wiki/Big_O_notation