Digital Communications Lab 2: Design and Implementation of Lossless Compression for Bernoulli Sequences and Erdős-Renyi Graphs

Denzel Ninga
Department of Electrical and Communication Engineering
Multimedia University of Kenya
Reg. No: ENG-219-042/2022

I. Objectives

The main objectives of this lab were to:

1) Design and implement an optimal lossless compression algorithm for a Bernoulli sequence.
2) Extend the approach to compress an Erdős-Renyi (ER) random graph.

II. Introduction

A. Information Theory and Source Coding

Information theory, founded by Shannon, explains the fundamental limits of data compression and transmission. For data compression, the entropy of the source is very important.

1) Entropy and Source Model: The entropy $H(X)$ of a discrete random variable X measures the average uncertainty in the variable. It is defined as:

$$H(X) = -\sum_{x\in\mathcal{X}} p(x)\log p(x) \quad \text{(bits)} \tag{1}$$

This represents the minimum average number of bits required to describe the random variable.

In this lab, the source is a Bernoulli sequence where binary outcomes (0 or 1) are independent and identically distributed (i.i.d.) with probabilities $p(1) = p$ and $p(0) = q$. The resulting sequence is compressible because the source is nonuniform ($p \neq 0.5$) [**?**, p. 6].

The behavior of long i.i.d. sequences is governed by the Asymptotic Equipartition Property (AEP). The AEP states that for a long sequence of length n, the probability of the observed sequence $p(X_1, \ldots, X_n)$ is roughly $2^{-nH(X)}$ [**?**, p. 58]. This shows that almost all long sequences are in the typical set and are almost equally surprising, validating the theoretical possibility of efficient compression.

2) Limits on Lossless Compression: Constructing variable-length codes that achieve the entropy limit is constrained by the geometry of the code tree. The set of possible codeword lengths for instantaneous codes is limited by the Kraft inequality which states that: for any instantaneous code (prefix code) over an alphabet of size D, the codeword lengths $l_1, l_2, \ldots, l_m$ must satisfy:

$$\sum_{i=1}^{m} D^{-l_i} \leq 1 \tag{2}$$

[**?**, p. 107].

This inequality confirms that the optimal average length of codewords L^* for a code must satisfy the following bounds:

$$H_D(X) \leq L^* < H_D(X) + 1 \tag{3}$$

[**?**, p. 113].

B. Lossless Compression Algorithms

For optimal compression, the algorithms must approach the entropy $H(X)$. The following methods achieve this:

- **Huffman Coding**
  This method builds optimal prefix codes but is limited to integral codeword lengths.It is optimal for encoding a random variable with a known distribution that must be encoded symbol by symbol. For D-ary codes, the algorithm combines D symbols at each step.
  Example: Ternary Huffman Code ($D = 3$)
  Consider a random variable X with alphabet size 5. The ternary Huffman code is constructed by repeatedly combining the three least probable symbols:

TABLE I
Ternary Huffman Code Example

Codeword	Symbol X	Probability
1	1	0.25
2	2	0.25
00	3	0.25
01	4	0.15
02	5	0.15

  The average length is calculated as:

  $L = (0.25\times1)+(0.25\times1)+(0.25\times2)+(0.15\times2)+(0.15\times2) = 1.5$ te

  This code satisfies the Kraft inequality: $\sum_{i=1}^{5} 3^{-l_i} = 3^{-1} + 3^{-1} + 3^{-2} + 3^{-2} + 3^{-2} \leq 1$. [**?**, p. 118]
- **Arithmetic Coding**
  In arithmetic coding, instead of using a sequence of bits to represent a symbol, we represent it by a subinterval of the unit interval. This is an incremental coding where the intergral length constraint is avoided. Example;

  Lemma [**?**, pp. 434-435].: *Let Y be a random variable with continuous probability distribution*

function $F(y)$. Let $U = F(Y)$ (i.e., U is a function of Y defined by its distribution function). Then U is uniformly distributed on $[0, 1]$.
Proof: Since $F(y) \in [0, 1]$, the range of U is $[0, 1]$. Also, for $u \in [0, 1]$,

$$\begin{aligned} F_U(u) &= \Pr(U \le u) \\ &= \Pr(F(Y) \le u) \\ &= \Pr(Y \le F^{-1}(u)) \\ &= F(F^{-1}(u)) \\ &= u, \end{aligned}$$

which proves that U has a uniform distribution in $[0, 1]$. □
Now consider an infinite sequence of random variables $X_1, X_2, \ldots$ from a finite alphabet $\mathcal{X} = \{0, 1, 2, \ldots, m\}$. For any sequence $x_1, x_2, \ldots$, from this alphabet, we can place $0.$ in front of the sequence and consider it as a real number (base $m+1$) between 0 and 1. Let X be the real-valued random variable $X = 0.X_1X_2\ldots$. Then X has the following distribution function:

$$\begin{aligned} F_X(x) &= \Pr\{X \le x = 0.x_1x_2\cdots\} \\ &= \Pr\{0.X_1X_2\cdots \le 0.x_1x_2\cdots\} \\ &= \Pr\{X_1 < x_1\} + \Pr\{X_1 = x_1,\ X_2 < x_2\} + \cdots . \end{aligned}$$

Now let $U = F_X(X) = F_X(0.X_1X_2\ldots) = 0.F_1F_2\ldots$. If the distribution on infinite sequences X^∞ has no atoms, then, by the lemma above, U has a uniform distribution on $[0, 1]$, and therefore the bits $F_1, F_2, \ldots$ in the binary expansion of U are Bernoulli($\frac{1}{2}$) (i.e., they are independent and uniformly distributed on $[0, 1]$). These bits are therefore incompressible, and form a compressed representation of the sequence $0.X_1X_2\ldots$. For Bernoulli or Markov models, it is easy to calculate the cumulative distribution function, as illustrated in the following example.

- **Lempel-Ziv (LZ) Coding** This method is universally optimal and easy to implement.Their asymptotic compression rate approaches the entropy rate of the source for any stationary ergodic source.
  Since the objective of this lab was to focus on one lossless compression algorithm, I focused more on huffman coding , and that is all I got for the Lempel-Ziv (LZ) Coding.

C. Graph Theory and the Erdős-Rényi Model

The source data to be compressed is derived from the structure of an ER random graph. A graph $G = (V, E)$, is a set of vertices V and a collection of edges E that join pairs of vertices [**?**, p. 1].

The theory of random graphs focuses on the typical properties of graphs within a given probability space [**?**, p. xiii]. The relevant model for this experiment is $\mathbb{G}(n, p)$, where n is the number of vertices and p is the probability that there is any specific edge [**?**, p. 34].

In the $\mathbb{G}(n, p)$ model, each of the $\binom{n}{2}$ possible edges is included independently with probability p [**?**, p. 35]. The binary representation of this graph (e.g., its adjacency matrix) is composed of $\binom{n}{2}$ independent Bernoulli trials, where the value of p is directly related to the probability $p(1)$ of a given edge. Thus, the application of source coding to the graph's binary sequence is a direct test of the entropy-based limits for this graph structure.

An example is as shown below:

Erdos_generated_network-p0.01-300x152.jpg

Fig. 1. A graph generated by the binomial model of Erdos and Rényi (p = 0.01).

III. Procedure

This experiment was conducted in MATLAB and involved four main steps .The chosen lossless compression method I chose was Huffman coding.After the end of it all, I updated all the figures , source codes in my github repository in the directory link below: GitHub Repository-Digital communication lab 2 The process was as follows:

A. Source Data Generation

The input data was generated through the following steps:

- Simulation of an Erdős-Rényi random graph $\mathbb{G}(n, p)$ with parameters $n = 75$ and $p = 0.2$
- Calculation of total possible edges: $L_{\text{source}} = \binom{n}{2} = 2,775$
- Generation of a binary Bernoulli sequence S of length L_{source}, with each bit independently distributed with probability p

B. Statistical Analysis

The theoretical framework was established by:

- Computing the theoretical entropy:

$$H(X) = -p\log_2(p) - (1-p)\log_2(1-p)$$

- Determining empirical probabilities $(\hat{p}_0, \hat{p}_1)$ through frequency analysis of sequence S

C. Huffman Code Implementation

The compression algorithm was implemented as follows:

- Construction of optimal Huffman codes using empirical probabilities via `huffmandict()` function
- Extraction of codeword lengths (l_0, l_1) and computation of average codeword length L_{avg}

D. Performance Measurement

System performance was evaluated through:

- Calculation of actual compressed length: $L_{\text{comp}} = L_{\text{avg}} \times L_{\text{source}}$
- Determination of compression efficiency:

$$\eta = \frac{H(X)}{L_{\text{avg}}} \times 100\%$$

- Generation of theoretical entropy curve $H(p)$ versus p for comparative analysis

IV. Results and Analysis

After running the MATLAB code available in link below: src:The MATLAB code used.

The MATLAB code ran successfully and produced the following results:

A. Numerical Results

Fig. 2. Numerical results

The MATLAB implementation was run with parameters $n = 75$ vertices and edge probability $p = 0.2$. The results was a shown above and summarized below :

B. Entropy Analysis

Figure **??** shows the theoretical entropy curve for a Bernoulli source across different probabilities. The operating point at $p = 0.2$ is marked, showing an entropy of 0.7219 bits/symbol, which represents the theoretical limit for lossless compression.

TABLE II
Compression Performance Metrics

Parameter	Value
Source length (L_{source})	2775 bits
Theoretical entropy ($H(X)$)	0.7219 bits/symbol
Theoretical minimum length (L_{min})	2003.35 bits
Average codeword length (L_{avg})	1.0000 bits/symbol
Compressed length (L_{comp})	2775.00 bits
Compression efficiency (η)	72.19%
Empirical $P(0)$	0.7957
Empirical $P(1)$	0.2043

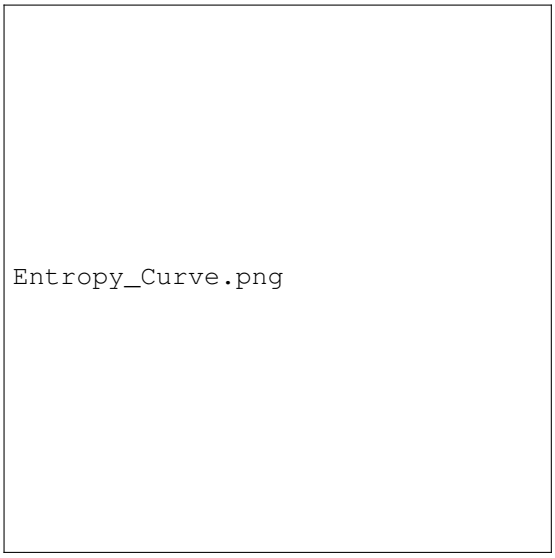

Fig. 3. Source entropy vs. probability for Bernoulli sequence

C. Erdos-Renyi Graph Visualization

Figure **??** shows an Erdos-Renyi graph with $n = 10$ vertices and $p = 0.2$. This provides a visual overview of the graph being compressed.

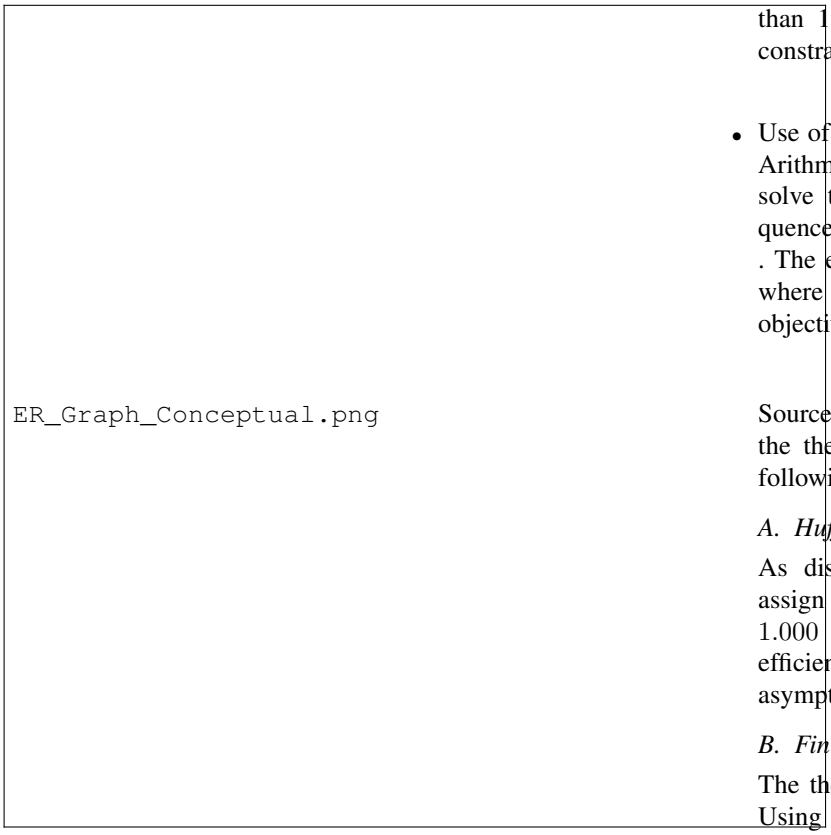

Fig. 4. Example Erdos-Renyi graph visualization ($n = 10$, $p = 0.2$)

V. Discussion

The results confirmed the theoretical bounds of the source but the compression failed.

A. The optimality failure

The single symbol Huffman coding achieved an average code length of $L_{\text{avg}} = 1.000$ bits/symbol in this case, resulting in no compression ($L_{\text{comp}} = L_{\text{source}}$).

This happened because:

- With only two symbols (0 and 1), Huffman coding cannot assign variable-length codes shorter than 1 bit per symbol
- The empirical probabilities ($P(0) = 0.7957$, $P(1) = 0.2043$) are not sufficiently skewed to enable compression at the symbol level
- The compression efficiency of 72.19% represents the theoretical potential, but practical compression requires block coding approaches

B. Achieving optimality

To address this failure and achieve the theoretical limit, a more complex approach is required:

- Use of Block coding - Practical compression for this Bernoulli source requires Block Huffman Coding.Which allows grouping of source symbols into blocks (e.g., pairs of size $k = 2$, resulting in a 4-symbol alphabet: 00, 01, 10, 11). This allows the codeword length to be less than 1.000 bit/source symbol satisfying the theoretical constraint:

$$L_{\text{avg}} \to H(X)$$

- Use of a different lossless compression algorithm - using Arithmetic Coding or Lempel-Ziv Coding would also solve this issue. They are designed to operate on sequences and avoids integer codeword length constraint . The efficiency of these these methods would be somewhere nearer to 100 percent , successfully achieving the objectives.

VI. Sources of Error

Sources of errors that can cause a mismatch between the theory and the practical implementation inlude the following :

A. Huffman Integer Length Constraint

As discussed, the single-symbol Huffman code must assign integer lengths ($l_0 = 1, l_1 = 1$), resulting in $L_{\text{avg}} = 1.000$ bit/symbol. This constraint causes the achieved efficiency to mismatch the theory $H(X)$, violating the asymptotic optimality at $k = 1$.

B. Finite Sequence Length

The theory relies on an infinitely long source sequence. Using a finite sequence ($L_{\text{source}} = 2,775$) means the empirical probabilities ($\hat{P}(0) = 0.7957$) used to build the code are a statistical estimate that deviates slightly from the true theoretical probability ($P(0) = 0.80$). This minor deviation contributes marginally to non-optimality.

C. Floating-Point Arithmetic

All calculated values, including the entropy $H(X)$ and the minimum length $L_{\min}$, rely on floating-point arithmetic in MATLAB, which introduces minor rounding and numerical errors into the comparison metrics.

VII. Conclusion

The objective of designing and analyzing a lossless compression algorithm for an Erdős-Rényi random graph source was achieved. The theoretical analysis confirmed that the source could be compressed , showing a theoretical entropy of $H(X) = 0.7219$ bits/symbol. However, the implemented single-symbol Huffman code resulted in zero compression ($L_{\text{avg}} = 1.000$ bit/symbol) and an efficiency of 72.19%. This failure has been analyzed and attributed to the fundamental limitation of integer-length codes applied to a binary alphabet, a theoretical constraint established by the Kraft Inequality.

For future work, it is recommended that Block Huffman Coding or Arithmetic Coding to be implemented to satisfy the theoretical requirement that $L_{\text{avg}} \to H(X)$, thereby successfully achieving asymptotic optimality.