UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE CIÊNCIAS - CAMPUS BAURU
DEPARTAMENTO DE COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

EXPLORANDO ALGORITMOS DE COMPRESSÃO DE DADOS: TEORIA, IMPLEMENTAÇÃO E DESEMPENHO

Nome: Gustavo Yujii Silva Kadooka	RA: 221021582
Nome do Orientador: Andréa Carla Gonçalves Vianna	Assinatura:

BAURU
Março/2025

GUSTAVO YUJII SILVA KADOOKA

EXPLORANDO ALGORITMOS DE COMPRESSÃO DE DADOS: TEORIA, IMPLEMENTAÇÃO E DESEMPENHO

Proposta para Trabalho de Conclusão de Curso do Curso de Bacharelado em Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, Campus Bauru.
Orientador: Andréa Carla Gonçalves Vianna

BAURU
Março/2025

Sumário

1 Introdução

O tema deste trabalho é a exploração dos algoritmos clássicos de compressão de dados, abordando seus aspectos teóricos, implementação prática e análise de desempenho. A compressão de dados é uma técnica essencial na ciência da computação, utilizada para reduzir o tamanho dos arquivos, otimizando o uso de recursos e acelerando a transmissão de dados, o que se torna crucial na era digital em que vivemos (SALOMON; MOTTA, 2007).

O desenvolvimento dessa área teve início na década de 1950, com os primeiros métodos que visavam otimizar a utilização do espaço de armazenamento. Entre esses métodos, destaca-se o algoritmo de Huffman, que utiliza uma técnica de codificação de prefixo variável. Nesse algoritmo, símbolos mais frequentes recebem códigos de comprimento menor, enquanto símbolos menos frequentes recebem códigos mais longos (SALOMON; MOTTA, 2007). O método de Huffman é amplamente empregado em compressão sem perdas e se tornou fundamental para a criação de arquivos compactados em formatos como ZIP e GZIP (DEUTSCH, 1996). Sua simplicidade, aliada à sua eficácia, fez com que se consolidasse como um dos pilares da compressão de dados.

Nos anos seguintes, com o aumento da demanda por transmissão de grandes volumes de dados, novos algoritmos surgiram. Um exemplo significativo é o LZ77 (Lempel-Ziv 77), introduzido por Abraham Lempel e Jacob Ziv em 1977 (ZIV; LEMPEL, 1977). O LZ77 aplica uma técnica de compressão baseada em dicionários, onde sequências de dados repetidas são substituídas por referências a posições anteriores. Essa abordagem inspirou a criação de outros algoritmos, como o LZW (Lempel-Ziv-Welch), que aprimora o LZ77 ao criar dinamicamente um dicionário de strings enquanto os dados são comprimidos (WELCH, 1984). O LZW é particularmente famoso pela sua aplicação no formato de compressão GIF, bem como em arquivos TIFF.

Além desses, o GZIP se destaca como outro algoritmo amplamente utilizado, especialmente em sistemas Unix e na web. O GZIP combina a codificação de Huffman com o algoritmo LZ77, permitindo uma compressão eficiente e rápida, sem perda de dados (DEUTSCH, 1996). Sua popularidade decorre da combinação de alta taxa de compressão e facilidade de descompressão, sendo uma escolha recorrente em aplicações como transferência de arquivos e armazenamento de dados.

Os métodos de Huffman, LZ77, LZW e GZIP formam a espinha dorsal dos algoritmos clássicos de compressão de dados e continuam sendo amplamente utilizados em diversas áreas, apesar do surgimento de novas abordagens.

Atualmente, a compressão de dados continua a ser um campo dinâmico e em constante evolução, com diversos algoritmos sendo constantemente estudados e aprimorados. Entre os

mais recentes, destacam-se os métodos baseados em compressão adaptativa, que ajustam seus parâmetros conforme as características dos dados a serem comprimidos, oferecendo maior eficiência dependendo do conteúdo (SALOMON; MOTTA, 2007). Um exemplo é o Brotli, um algoritmo de compressão sem perdas desenvolvido pela Google, amplamente utilizado para comprimir arquivos em navegadores web. O Brotli combina codificação de Huffman e transformações de fluxo de dados, permitindo taxas de compressão superiores às oferecidas por algoritmos tradicionais como o GZIP (ALAKUIJALA et al., 2016).

Além disso, algoritmos de compressão em tempo real têm ganhado relevância devido ao crescente volume de dados gerados em tempo real, como no streaming de vídeo e nas comunicações móveis. O Zstandard (ou Zstd), desenvolvido pelo Facebook, é um exemplo notável. Ele oferece uma excelente combinação entre alta taxa de compressão e velocidade, podendo ser utilizado tanto em compressões em tempo real quanto em grandes volumes de dados (COLLET, 2016). O Zstandard é considerado uma evolução do LZ4, sendo mais eficiente tanto na compressão quanto na descompressão.

Em áreas como compressão de imagens e áudio, técnicas baseadas em aprendizado de máquina e redes neurais também têm atraído atenção. No campo da compressão de imagens, os Modelos Generativos Adversariais (GANs) estão sendo explorados para desenvolver algoritmos que geram representações comprimidas de alta qualidade, mantendo os detalhes visuais. Na compressão de áudio, algoritmos baseados em redes neurais, como o WaveNet (DeepMind, 2016), estão sendo experimentados para aprimorar a qualidade da compressão, tanto com perdas quanto sem perdas.

Outro campo de grande importância é a compressão de vídeo, especialmente para plataformas de streaming como Netflix e YouTube. Algoritmos como o HEVC (High Efficiency Video Coding), uma evolução do H.264, continuam a ser aprimorados para garantir uma compressão mais eficiente, sem perda significativa de qualidade. Além disso, o desenvolvimento de novos codecs como o VVC (Versatile Video Coding) está em andamento, com a promessa de oferecer compressão ainda mais eficiente para vídeos de alta definição e 4K.

A análise de desempenho, que inclui a eficiência em termos de tempo de execução e taxa de compressão, também desempenha um papel crucial. Com o aumento exponencial da quantidade de dados gerados e transmitidos na sociedade moderna, os algoritmos de compressão tornaram-se ainda mais essenciais. A eficiência na compressão não só reduz os custos de armazenamento e acelera a transmissão de dados, mas também impacta diretamente áreas como streaming de vídeos, comunicação móvel, redes de computadores e sistemas de armazenamento em nuvem. Compreender os diferentes métodos e avaliar sua aplicabilidade em cenários diversos é fundamental para o desenvolvimento de soluções mais eficazes e eficientes (SALOMON; MOTTA, 2007).

2 Problema

O problema a ser investigado neste trabalho é: "Como os diferentes algoritmos clássicos de compressão de dados (Huffman, LZ77, LZW, GZIP) se comparam em termos de eficiência de compressão e tempo de execução, e qual é o impacto dessas variáveis em aplicações práticas?"(SALOMON; MOTTA, 2007). A pesquisa busca analisar como o desempenho desses algoritmos varia dependendo do tipo de dado a ser comprimido (como texto, imagem e áudio) e qual algoritmo oferece o melhor equilíbrio entre eficiência e tempo de processamento em diferentes cenários.

A hipótese inicial que orienta este estudo é que os algoritmos clássicos de compressão apresentam diferenças significativas em termos de tempo de execução e taxa de compressão, dependendo do tipo de dado e da aplicação específica. Por exemplo, o GZIP pode ser mais eficiente em termos de tempo de execução (DEUTSCH, 1996), enquanto o Huffman pode ser mais eficaz em termos de compressão de arquivos de texto (SALOMON; MOTTA, 2007). No entanto, a escolha do algoritmo deve ser feita levando em consideração as necessidades específicas de cada aplicação, como a necessidade de compressão rápida em tempo real ou a maximização da taxa de compressão em arquivos grandes.

A delimitação deste estudo inclui a análise de algoritmos de compressão sem perdas (*lossless*), com foco nos quatro algoritmos clássicos mencionados, aplicados a três tipos principais de dados: arquivos de texto, imagens em formato PNG e pequenos arquivos de áudio. A pesquisa não incluirá algoritmos de compressão com perdas (*lossy*), como JPEG e MP3, ou algoritmos adaptativos modernos, como Brotli (ALAKUIJALA et al., 2016) ou Zstandard (COLLET, 2016).

Este problema é relevante, pois a escolha do algoritmo de compressão adequado tem um impacto direto no desempenho de sistemas de armazenamento e comunicação de dados, afetando, por exemplo, a velocidade de transferência de arquivos em redes e a eficiência no uso de recursos de armazenamento em nuvem (DEUTSCH, 1996). Além disso, a análise comparativa permitirá uma compreensão mais aprofundada das vantagens e limitações de cada algoritmo em diferentes cenários de uso.

A pesquisa será viável por meio da implementação prática dos algoritmos selecionados e da realização de experimentos de compressão, medindo tanto a eficiência em termos de taxa de compressão quanto o tempo de execução. Será possível gerar conclusões sobre qual algoritmo se destaca em cada tipo de dado e qual oferece o melhor desempenho global em diferentes cenários.

3 Justificativa

Este estudo justifica-se pela necessidade de compreender de forma aprofundada como diferentes algoritmos clássicos de compressão se comportam em termos de taxa de compressão e tempo de execução, analisando sua aplicabilidade em diferentes cenários (SALOMON; MOTTA, 2007). Embora os algoritmos de Huffman, LZ77, LZW e GZIP sejam amplamente utilizados, muitas vezes a escolha entre eles é feita sem uma análise detalhada de suas vantagens e limitações para tipos específicos de dados (ZIV; LEMPEL, 1977; WELCH, 1984). Um estudo comparativo rigoroso pode fornecer insights valiosos para desenvolvedores, engenheiros de software e administradores de sistemas que buscam otimizar o uso de recursos computacionais.

Além disso, a pesquisa contribui para o conhecimento técnico ao oferecer um panorama detalhado da eficiência desses algoritmos, permitindo um entendimento mais preciso sobre sua aplicabilidade em diferentes tipos de arquivos. Os resultados obtidos podem servir como referência para a escolha do algoritmo mais adequado em aplicações práticas, como armazenamento de dados compactados, otimização de tráfego em redes e sistemas embarcados com restrições de processamento e memória (DEUTSCH, 1996; COLLET, 2016).

Os fatores motivadores para a escolha deste problema incluem a crescente demanda por soluções eficientes de compressão de dados na era digital (ALAKUIJALA et al., 2016), a necessidade de balancear taxa de compressão e tempo de execução em diferentes contextos e a relevância contínua dos algoritmos clássicos, mesmo diante do surgimento de novas técnicas (COLLET, 2016). Dessa forma, o estudo não apenas revisita fundamentos essenciais da compressão de dados, mas também propõe uma análise comparativa aplicada, fornecendo contribuições práticas e teóricas para a área.

4 Objetivos

4.1 Objetivo Geral

Analisar comparativamente a eficiência dos algoritmos clássicos de compressão de dados (Huffman, LZ77, LZW e GZIP), avaliando suas taxas de compressão e tempos de execução em diferentes tipos de arquivos, a fim de determinar o impacto dessas variáveis na escolha do algoritmo mais adequado para aplicações práticas.

4.2 Objetivos Específicos

a) Descrever os princípios teóricos dos algoritmos de compressão de dados Huffman, LZ77, LZW e GZIP, destacando seus fundamentos e aplicações.

b) Implementar os algoritmos selecionados, garantindo sua funcionalidade para compressão e descompressão de diferentes tipos de arquivos.

c) Comparar o desempenho dos algoritmos em termos de taxa de compressão e tempo de execução, utilizando arquivos de texto TXT, imagens PNG e pequenos arquivos de áudio WAV.

d) Analisar os resultados obtidos, identificando vantagens e limitações de cada algoritmo em diferentes cenários.

e) Avaliar a aplicabilidade dos algoritmos em diferentes contextos, fornecendo recomendações para a escolha do método mais adequado conforme a necessidade específica.

5 Metodologia

O estudo tem caráter **exploratório e descritivo**, pois busca compreender e comparar o desempenho de algoritmos clássicos de compressão de dados em diferentes tipos de arquivos. Além disso, trata-se de uma pesquisa **experimental e bibliográfica**, pois envolve tanto a revisão da literatura sobre os algoritmos estudados quanto a implementação prática e execução de testes para coleta de dados.

5.1 Delineamento da Pesquisa

A pesquisa será conduzida em duas etapas principais:

- **Revisão bibliográfica**: levantamento de referências acadêmicas sobre os algoritmos Huffman, LZ77, LZW e GZIP, incluindo seus princípios teóricos, implementação e aplicações.
- **Análise experimental**: implementação dos algoritmos e realização de testes comparativos, medindo a taxa de compressão e o tempo de execução em diferentes tipos de arquivos (texto, áudio e imagem).

5.2 Coleta de Dados

Os dados serão coletados por meio da execução dos algoritmos em um ambiente controlado, utilizando conjuntos de arquivos representativos dos três principais tipos de dados abordados:

- **Texto**: arquivos de livros e artigos científicos em formato `.txt`.
- **Imagem**: arquivos `.png`, que utilizam compressão sem perdas.
- **Áudio**: pequenos arquivos de áudio sem perdas, como `.wav`.

Cada algoritmo será executado sobre os mesmos conjuntos de dados e os seguintes parâmetros serão registrados:

- Taxa de compressão (% de redução do tamanho do arquivo);
- Tempo de execução (medido em milissegundos);

5.3 Análise dos Dados

Os resultados serão tabulados e analisados estatisticamente para identificar padrões e diferenças entre os algoritmos. A análise será feita considerando:

- Comparação direta entre a taxa de compressão e tempo de execução;
- Eficiência relativa dos algoritmos para cada tipo de dado;
- Gráficos de desempenho para facilitar a interpretação dos resultados.

5.4 Ferramentas e Ambiente de Teste

Os experimentos serão conduzidos em um ambiente computacional padronizado para garantir reprodutibilidade. As ferramentas utilizadas incluem:

- Linguagem de programação: C ou C++, para implementação e execução dos algoritmos;
- Bibliotecas: `pandas` e `matplotlib` para análise e visualização dos dados;
- Sistema operacional: Windows 11 Pro com Debian WSL;
- Hardware: Processador AMD Ryzen 3 PRO 8300GE w/ Radeon 740M Graphics, 3501 Mhz, 4 Núcleo(s), 8 Processador(es) Lógico(s), 16GB RAM.

Para gerar os gráficos coletaremos os dados como CSV e depois utilizaremos através das bibliotecas de análise e visualização de dados mencionados.

Com esse delineamento metodológico, espera-se obter uma avaliação quantitativa e qualitativa dos algoritmos estudados, possibilitando uma conclusão fundamentada sobre suas aplicações ideais.

6 Cronograma

O cronograma a seguir apresenta a distribuição das atividades do Trabalho de Conclusão de Curso ao longo do período disponível, considerando os prazos estipulados pela instituição.

Semana	**Atividades presenciais**	**Atividades extraclasse**
05 (24 a 30/03)	Redação da proposta do TCC	Desenvolvimento de biblioteca wav para lidar com .wav
06 (31/03 a 05/04)	Entrega da proposta do TCC	Desenvolvimento de Huffman Coding para .txt
07 (07 a 12/04)	Impressão da proposta e apresentação para os colegas	Desenvolvimento de Huffman Coding para .wav
08 (14 a 17/04)	Aguardando feedback da proposta	Continuação do desenvolvimento de Huffman Coding para .wav
09 (21 a 26/04)	Revisão e adequação da proposta (se necessário)	Desenvolvimento de Huffman Coding para .png
10 (28 a 30/04)	Postagem e reapresentação da proposta com adequações (se necessário)	Desenvolvimento de Huffman Coding para .png
11 a 15 (05/05 a 07/06)	-	Pesquisa bibliográfica aprofundada, leitura de artigos e livros, planejamento da implementação. Desenvolvimento do algoritmo LZ77 para .txt, .png e .wav
16 (09 a 14/06)	Preparação da apresentação do primeiro semestre, incluindo resumo do projeto e progresso até o momento	Desenvolvimento do algoritmo LZW para .txt
17 (16 a 18/06)	Apresentação do progresso do primeiro semestre (caso necessário)	Desenvolvimento do algoritmo LZW para .wav
Recesso (19/06 a 03/08)	-	Desenvolvimento da implementação dos algoritmos e testes preliminares (Desenvolvimento do algoritmo LZW para .png e do algoritmo Gzip para .txt, .png e .wav)
18 e 19 (04 a 16/08)	-	Ajustes nos algoritmos
20 (18 a 23/08)	Aula sobre regulamento do TCC e normas da ABNT	Testes de desempenho dos algoritmos, coleta de dados para análise comparativa
21 (25 a 30/08)	Apresentação do trabalho desenvolvido entre junho e agosto	Criação de gráficos com bibliotecas `Python`

Semana	**Atividades presenciais**	**Atividades extraclasse**
22 a 24 (01 a 20/09)	-	Criação de gráficos com bibliotecas `Python`
25 (22 a 27/09)	Apresentação das ferramentas utilizadas no desenvolvimento do TCC	-
26 (29/09 a 04/10)	Apresentação complementar das ferramentas (se necessário)	-
27 a 29 (06 a 25/10)	Entrega da cópia da monografia para cada membro da banca até 20/10	Discussão dos resultados, revisão e finalização da monografia
30 (29/10 a 01/11)	-	Revisão e finalização da apresentação
31 (03/11 a 08/11)	-	Revisão da monografia
32 (10/11 a 14/11)	Realização das Bancas Examinadoras	-

Tabela 1 – Cronograma de desenvolvimento do TCC

Referências

ALAKUIJALA, J.; SZABADKA, Z.; VANDEVENNE, L.; FARRUGGIA, R.; KLIUCHNIKOV, E.; SZABADKAI, I.; ASSCHE, Z. V. Brotli compressed data format. In: INTERNATIONAL WORLD WIDE WEB CONFERENCES STEERING COMMITTEE. *Proceedings of the 25th International Conference on World Wide Web*. [S.l.], 2016. p. 1593–1603.

COLLET, Y. *Zstandard Compression Algorithm*. 2016. <https://facebook.github.io/zstd/>. Accessed: 2025-03-20.

DeepMind. *WaveNet: A generative model for raw audio*. 2016. Accessed: 2025-03-24. Disponível em: <https://deepmind.google/research/breakthroughs/wavenet/>.

DEUTSCH, P. *GZIP file format specification version 4.3*. 1996. <https://tools.ietf.org/html/rfc1952>. Accessed: 2025-03-20.

SALOMON, D.; MOTTA, G. *Data Compression: The Complete Reference*. 4th. ed. [S.l.]: Springer, 2007. ISBN 978-1-84628-602-5.

WELCH, T. A. A technique for high-performance data compression. *Computer*, IEEE, v. 17, n. 6, p. 8–19, 1984.

ZIV, J.; LEMPEL, A. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, IEEE, v. 23, n. 3, p. 337–343, 1977.