# Bootstrap Analysis of Compression Algorithms

Antonio A. R. Beserra, Leandro C. Souza, and Daniel F. L. Souza.

***Abstract*—Compression algorithms have been proposed with the technology advance. However, there are not objective analysis procedures to guide a future choice of an algorithm directed for the type of data in the system they are intended for. This paper introduces a statistical framework, based on the bootstrap method, to execute the analysis of compression algorithms using an objective comparison parameter as a criterion. A case study using the compression ratio as the parameter and file samples of 4 different types was analyzed. The proposed scheme allowed us to infer which algorithm is better to be used for each data type. RLE has proven more suitable to image, audio and video files with Huffman obtaining comparable performance. For text files, LZW has remarkably outperformed all other algorithms.**

***Index Terms*—Compression algorithm, Statistical analysis, Bootstrap.**

## I. Introdução

Com o aumento do número de serviços oferecidos na Internet, como as redes sociais e serviços de *stream*, fluxos de dados cada vez maiores passaram a ser transmitidos, como imagens e vídeos. Para manter a integridade e/ou confidencialidade desses dados, foram desenvolvidos protocolos que se incorporaram aos protocolos de comunicação existentes, tais como o *IP Security* (IPSec) e o *Security Socket Layer* (SSL) [1], [2]. Em suas especificações, esses protocolos opcionalmente utilizam de compressão a fim de enviar um maior fluxo de dados em um menor tempo. Entretanto, não há qualquer regra determinando qual algoritmo será utilizado. Apenas um consenso entre as partes comunicantes é realizado a fim de que utilizem o mesmo método.

Estudos que avaliaram algoritmos de compressão o fizeram com diversas metodologias e diferentes critérios de análise [3]–[6]. Jambek e Khairi [3] compararam Huffman e LZW com suas combinações, HLZ e LZH. Eles avaliaram taxa de compressão, tempo de compressão e descompressão em dados típicos de redes de sensores sem fio, tais como temperatura e umidade, e arquivos de texto extraídos do site do *Mother Goose Club*. Contudo, não explicitaram como os dados dos sensores estariam estruturados e se estavam em formato binário ou texto, o que compromete a replicabilidade do estudo. Gupta, Bansal e Khanduja [4] avaliaram algoritmos de compressão modernos como *Deflate*, *Bzip2*, LZMA, PPMd e PPMonstr. A avaliação foi feita ao comparar as médias da taxa de compressão e velocidade de compressão e descompressão (em Megabytes por segundo) de diversos tipos de dados como executáveis, imagens, html, xhtml e textos em inglês do *Silesia corpus*. Eles concluíram que se torna necessário decidir o algoritmo de compressão baseado no tipo do arquivo e requisitos de aplicação. Nesse sentido, o *framework* proposto neste estudo auxilia na tomada desse tipo de decisão, além de garantir a replicabilidade dos experimentos dado que a base de dados seja fornecida e esteja disponível.

Hidayat, Zakaria e Pee [5] avaliaram o tempo, fator e taxa de compressão de Huffmann e codificação aritmética a partir de um único arquivo de áudio em formato WAV. As conclusões a partir dessa metodologia experimental tornam-se precipitadas de serem tomadas considerando testes feitos em um único arquivo e que não replicação de amostra, o que é abordado no método estatístico *bootstrap* embutido no *framework* proposto. Rahman e Hamada [6] obtiveram diversas medidas de comparação dos algoritmos RLE, Shannon-Fano, Huffman, LZE e codificação aritmética a partir de 3 fotografias geradas por PC e 22 imagens médicas do *DICOM Image Dataset*. Dentre essas medidas estão os tempos de codificação e decodificação, número de bits para armazenar um pixel na média, taxa de compressão, PSNR e eficiência. A eficiência foi calculada como a razão entre entropia e comprimento médio de código em porcentagem. O processo de geração das imagens não foi descrito, o que inviabiliza a replicação desse estudo.

Diante desse contexto em que há diferentes medidas de análise entre algoritmos de compressão e diversas metodologias para compará-los, surge a necessidade de se estabelecer métricas e parâmetros confiáveis em relação a esses algoritmos a fim de se classificá-los. A uniformidade nessas escolhas não só garante fidedignidade aos resultados como também facilita futuras decisões em protocolos e sistemas nos quais eles são utilizados. Com esse objetivo em mente, neste trabalho é proposto um *framework* baseado no método estatístico chamado *bootstrap* [7] para análise e comparação de métodos de compressão através da avaliação de intervalos de confiança. Desta forma, foi realizado um estudo de caso para investigar se o tipo de dado impacta diretamente na taxa de compressão do algoritmo utilizado e indicar uma lista de recomendações de uso de algoritmos de compressão de forma a otimizar o uso deles. Com a aplicação do *framework* nesse estudo de caso, pôde-se ver que RLE é mais apropriado para arquivos de imagem, áudio e vídeo, com *Huffman* também obtendo desempenho comparável em vídeo e áudio. E que para arquivos de texto, LZW superou consideravelmente os outros algoritmos.

Na seção 2 são apresentados os trabalhos correlatos mais recentes. Na seção 3, são apresentados os embasamentos teóricos sobre algoritmos de compressão e sobre a utilização do método estatístico *bootstrap*. Na seção 4 é apresentado o *framework* que implementa o método proposto. Na seção 5 é discutido um estudo de caso e seus resultados. E na seção 6 são discutidas as considerações finais assim como propostas de trabalhos futuros.

Antonio A. R. Beserra is with the Universidade de São Paulo, São Carlos, São Paulo, Brazil (e-mail: antonio.alessandro@usp.br)

Leandro C. Souza is with Universidade Federal da Paraíba, João Pessoa, Paraíba, Brazil (e-mail: leandro@ci.ufpb.br)

Daniel F. L. Souza is with Universidade Federal Rural do Semi-Árido, Mossoró, Rio Grande do Norte, Brazil (e-mail: danielfaustino@ufersa.edu.br)

## II. Trabalhos Relacionados

Alguns trabalhos da literatura têm analisado e comparado a eficácia de diferentes algoritmos de compressão em diferentes contextos. Hussain, Al-Fayadh e Radi [8] discutiram as principais técnicas de compressão voltadas para imagens na literatura existente. Os autores as agruparam entre si e analisaram as técnicas sem perdas e com perdas. Os métodos Codificação Aritmética (AC, *Arithmetic Coding* em inglês), *Huffman* e Codificação Preditiva Sem Perdas foram categorizados como sem perdas e suas taxas de compressão avaliadas apenas como "baixa". Já os métodos Codificação Preditiva, Codificação de Transformação, JPEG, JPEG 2000 e Quantização de Vetor figuraram como as com perdas e suas taxas de compressão foram avalidas desde "baixa" até "alta".

Kalajdzic, Ali e Patel [9] desenvolveram um algoritmo para compressão de curtas mensagens de texto, *b64pack*, e compararam com programas de compressão do UNIX: *compress*, *gzip* e *bzip2*. O tempo de execução da compressão foi avaliado ao comprimir um conjunto de mais de 50 mil arquivos e o tempo médio de compressão por mensagem foi analisado. A taxa de compressão também foi comparada ao comprimir 1984 mensagens com mais de 160 caracteres e analisaram quantas versões comprimidas apresentavam 160 ou menos caracteres.

Campobello et al. [10] apresentaram um algoritmo de compressão sem perdas de baixas complexidades computacional e de memória voltado para redes de sensores sem fio. Foram comprimidos 100 blocos de dados cada um composto por 100 palavras de 16 bits cada, referentes a estatísticas meteorológicas como umidade do ar e temperatura, e compararam as máximas e médias das taxas de compressão com outros algoritmos com essa finalidade como *S-Huffman*, *ND-Encoding* e *Two-Modal*. Análises qualitativas também foram levantadas, como a complexidade computacional e o consumo de memória.

Anotny e Ganapathy [11] propuseram um algoritmo de predição intra seletiva para codificação sem perdas na codificação de vídeo de alta eficiência. 6 sequências de testes de 100 quadros cada foram categorizadas conforme a resolução e quadros por segundo. Essas sequências foram comprimidas comparando com outros algoritmos de compressão relacionados analisando a taxa de bits do vídeo comprimido e o tempo de execução da codificação e decodificação.

Ghido e Tabus [12] introduziram um algoritmo eficiente para calcular preditores lineares estéreos esparsos para compressão de áudio sem perdas. Os 10 segundos do meio de cada uma das 1.191 trilhas de áudio de vários gêneros musicais foram extraídos em formato WAVE a 44,1 kHz, 16 bits e estéreo. Como critérios de comparação, foram analisados a razão do tamanho total dos arquivos comprimidos pelo tamanho total dos arquivos originais em termos percentuais e o tempo médio de decodificação medido em múltiplos de tempo-real. Além disso, para avaliar a complexidade dos algoritmos testados, os autores se basearam no número médio de multiplicações feitas para decodificar um valor de amostra. Com essa medida, também foi avaliado o pior caso de complexidade computacional para um dos arquivos de teste.

Masmoudi e Puech [13] modelaram um esquema que realiza tanto compressão sem perdas através da AC, como a encriptação de imagens por meio de um gerador de números aleatórios. Eles comprimiram 12 imagens 8-bits em escala de cinza da coleção *Greyset2* da Universidade de *Waterloo* que variam de 464 x 352 a 672 x 498 pixels em tamanho. Foi avaliado o tamanho em bytes dos arquivos comprimidos comparando a versão estática da AC com o método proposto.

Chang et al. [14] criaram um método de compressão para tabela de índices de quantização vetorial. Comparando com outros 4 métodos de compressão propostos por outros autores para essa finalidade, eles comprimiram 4 imagens e analisaram a taxa de compressão e o tempo de execução dos algoritmos. Eles também analisaram a qualidade das imagens descomprimidas comparando com um outro método de um outro autor avaliando a relação sinal-ruído de pico (PSNR, *Peak Signal-to-Noise Ratio* em inglês).

Asif et al. [15] projetaram um método de compressão "quase" sem perdas, como dito por eles, para dados de tráfego aplicando o princípio da codificação residual e com perdas. Eles comprimiram dados de velocidade obtidos de uma rede composta de mais de 18 mil segmentos de estrada em Singapura no período correspondente a Agosto de 2011 aplicando codificação de *Huffman* em 4 modelos diferentes de baixa dimensão usados para representar os dados: a transformada discreta de cosseno, a decomposição em valores singulares, a fatorização de matriz não-negativa e a *wavelet*. Como critérios de análise, eles avaliaram a taxa de compressão e a PSNR para diferentes níveis de tolerância.

Através de uma análise dos trabalhos presentes na literatura de compressão de dados, observa-se que não há um consenso entre os pesquisadores sobre como avaliar ou analisar algoritmos de compressão entre si. Critérios como taxa de compressão, tempo de execução da codificação ou da decodificação são levados em conta a depender do contexto do estudo, mas o método usado para analisá-los é variado.

Souza et al. [16] elaboraram um método para extração de conhecimento em intervalos de dados. Em um estudo posterior [17], eles propuseram uma distância híbrida em agrupamento dinâmico de intervalos de dados baseada na distância $L_q$. Nesses trabalhos, os métodos propostos pelos autores foram comparados e analisados com outros do estado da arte através da geração de intervalos de confiança com *bootstrap*.

## III. Referencial Teórico

Nesta seção é apresentado um resumo das discussões feitas por outros autores sobre os assuntos nos quais se baseiam este trabalho.

### A. Algoritmos de Compressão

As tecnologias usadas no armazenamento de dados estão sendo gradualmente aprimoradas, assim como também vem aumentando o volume de dados gerados pela atividade decorrente da intervenção humana. No entanto, esse volume se desenvolve a uma taxa de crescimento maior do que as tecnologias de armazenamento conseguem acompanhar [18].

Para reduzir a quantidade de dados processados e armazenados, os algoritmos de compressão de dados têm sido estudados

e melhorados. Esses algoritmos são categorizados em duas classes: algoritmos de compressão sem perdas, em que a versão decodificada do dado é idêntica bit-a-bit à versão fonte; e algoritmos de compressão com perdas, em que a versão decodificada não é idêntica à versão fonte. Dentre os algoritmos de compressão sem perdas, podemos citar a codificação de *Huffman*, *Run Length Encoding* (RLE), *Lempel-Ziv 1* (LZ1) e *Lempel-Ziv-Welch* (LZW) [19].

RLE é um algoritmo de compressão que consiste em codificar uma sequência de símbolos repetidos, denominada carreira, como uma tupla contendo esse símbolo e o tamanho dessa carreira. Devido a essa particularidade, esse algoritmo toma vantagem da redundância espacial, especialmente presente em textos e imagens [20].

LZ1, também conhecido como LZ77 [20], é um algoritmo baseado em dicionário proposto pelos autores que lhe deram o nome. Ele se baseia em símbolos já lidos no arquivo, presentes em uma janela de busca, como um dicionário para substituir as próximas ocorrências das mesmas sequências de símbolos pela posição e tamanho de sua última ocorrência. Já o seu sucessor, LZW, adota uma estratégia alternativa. Ele insere entradas repetidas e cada vez mais longas em um dicionário conforme codifica o arquivo e então substitui uma sequência de símbolos presente nesse dicionário por um código correspondente a ela [19].

A codificação de *Huffman* é um algoritmo de codificação de comprimento variável. Nela são atribuídos códigos para símbolos do alfabeto presentes no arquivo, de tal maneira que os símbolos mais frequentes tenham os menores códigos, enquanto que os menos repetidos possuam os maiores. Para isso, uma lista dos símbolos do alfabeto em ordem decrescente de frequência é disposta e uma árvore binária que associa um símbolo em cada folha é construída, de baixo para cima, a partir dessa lista. Iterativamente, os dois símbolos com as menores frequências são adicionados ao topo da árvore, deletados da lista e substituídos por um símbolo auxiliar que representa a frequência conjunta deles. Quando a lista contém apenas um símbolo auxiliar, que representa o alfabeto inteiro, a árvore está completa e é percorrida para caracterizar o código dos símbolos [20].

### B. Bootstrap

O *bootstrap* é uma técnica computacional capaz de simular a partir de dados fornecidos e estimar o viés e a variância de uma determinada estatística [7], [21]. Também pode ser usado para construir intervalos de confiança de maneira não paramétrica ao reproduzir a distribuição de probabilidade dos dados [22].

Suponha um conjunto de valores reais $\Phi = \{x_1, x_2, ..., x_n\}$. Considere $B$ amostras independentes $\{\Phi^{*1}, \Phi^{*2}, \ldots, \Phi^{*B}\}$, todas de tamanho $n$, geradas de maneira aleatória e com reposição, do conjunto inicial, $\Phi$. Seja $\mu$ a média desse conjunto. Um intervalo de confiança é um conjunto de valores que um parâmetro pode assumir, com um erro predeterminado. Para calculá-lo, computa-se a média de cada uma das $B$ amostras. Com um nível de confiança $\alpha \times 100\%$ (em que $0 \leq \alpha \leq 1$), o intervalo de confiança é obtido pelas médias referentes aos índices $B \times \frac{1-\alpha}{2}$ e $B \times (1 - \frac{1-\alpha}{2})$, arredondados para cima, do conjunto de médias ordenado. O Algoritmo 1 descreve os passos para a construção de intervalos de confiança usando os percentis *bootstrap* [22].

**Algoritmo 1** Construção de intervalos de confiança

**Require:** Conjunto de valores reais $\Phi = \{x_1, x_2, ..., x_n\}$; Número de amostras *bootstrap*: $B$; Nível de confiança: $\alpha$, em que $0 < \alpha < 1$;
**Ensure:** Intervalo de confiança $[a, b]$.
1: $\epsilon = 1 - \alpha$
2: **for** $i = 1$ **to** $B$ **do**
3: $amostra_i \leftarrow$ uma amostra de $\Phi$ de tamanho $n$ e com reposição
4: $m_i \leftarrow$ média de $amostra_i$
5: **end for**
6: Ordene o conjunto de médias $\{m_1, m_2, ..., m_B\}$ em ordem crescente
7: $x \leftarrow \lceil B \times \frac{\epsilon}{2} \rceil$
8: $y \leftarrow \lceil B \times (1 - \frac{\epsilon}{2}) \rceil$
9: $a \leftarrow m_x$
10: $b \leftarrow m_y$

Sejam dois intervalos de confiança *bootstrap* $I_1$ e $I_2$, gerados para as médias $\mu_1$ e $\mu_2$ de dois conjuntos $\Phi_1$ e $\Phi_2$. Pode-se analisar testes de hipóteses relacionados a elas com esses intervalos. Se o valor presente na hipótese nula não pertence ao intervalo, então ela deve ser rejeitada [7]. A relação que há entre testes de hipóteses e intervalos de confiança facilita a comparação das médias de métricas de performance quando vários métodos são analisados entre si. Se há uma intersecção não-vazia entre os intervalos, consideram-se as médias como iguais. Se todos os valores em $I_1$ são maiores que todos em $I_2$ deduz-se que $\mu_1 > \mu_2$. Sendo assim, os resultados do estudo de caso deste trabalho são mostrados através de intervalos de confiança gerados a partir dos percentis *bootstrap* oriundos de distribuições de métricas de desempenho, aplicados na avaliação dos algoritmos de compressão presentes na literatura.

## IV. Framework Proposto

Para estabelecer uma metodologia bem definida para o processo de comparação entre algoritmos de compressão, foi elaborado um *framework* baseado no método estatístico *bootstrap*. Na Fig. 1 o modelo da arquitetura geral do *framework* proposto é apresentado.

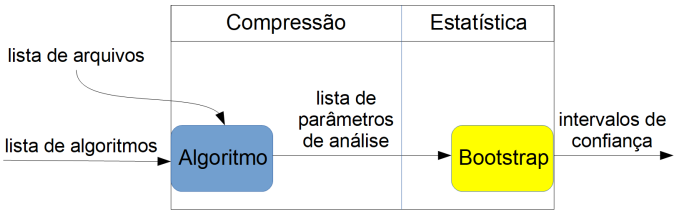



Fig. 1. Arquitetura geral do *framework*

No *framework*, toma-se como entrada uma lista de arquivos a serem comprimidos e uma lista de algoritmos a serem

analisados. Cada método de compressão da lista de algoritmos é instanciado para comprimir a lista de arquivos. Em cada execução dessas instâncias, é gerada uma lista do parâmetro de análise utilizado. O *framework* pode ser estendido para que esse parâmetro seja o tempo de execução do algoritmo testado ou sua utilização de memória, assim como qualquer outra métrica de desempenho apropriada a uma técnica de compressão e que possa ser mensurada quantitativamente.

Após gerar essa lista contendo os parâmetros de análise, ela é tomada como entrada no módulo estatístico do *framework*, que usa o método *bootstrap* para gerar e retornar os intervalos de confiança referentes aos algoritmos testados.

### A. Desenvolvimento

O *framework* foi implementado em linguagem C++ para o sistema operacional Windows 10. Ele foi construído para se configurar a partir de um arquivo de configuração. Nele são listados os algoritmos de compressão a serem testados e os diretórios onde se encontram os arquivos a serem comprimidos. Uma classe *Properties* foi projetada para ler esse arquivo e obter os devidos parâmetros de inicialização do *framework*.

Para avaliar o desempenho de um algoritmo de compressão é usada a taxa de compressão. Considere $B_0$ como o número de bytes usados para armazenar o dado antes da compressão e $B_1$ como o número de bytes usados para armazená-lo após a compressão, então se define a taxa de compressão como $B_0/B_1$. Em geral, busca-se num método de compressão que sua taxa de compressão seja a maior possível, de preferência maior que 1. Caso ela seja menor que 1, o arquivo compactado acaba sendo maior que o original. Assim, quanto maior a taxa de compressão, melhor o algoritmo de compressão [19].

Cada algoritmo de compressão é representado em uma classe que implementa a interface *Compactor*. Essa interface, por sua vez, possui os métodos *encode*, que recebe um caminho de um arquivo a ser comprimido e retorna a taxa de compressão obtida para aquele arquivo; e *decode*, que recebe um caminho de um arquivo já comprimido previamente e o descomprime, gerando o arquivo original. Na Fig. 2 é mostrado o diagrama de classes referente a esse módulo do *framework*.

Após inicializar os devidos parâmetros de configuração, para cada algoritmo na lista de algoritmos, comprime-se cada arquivo da lista de arquivos e se obtém a taxa de compressão para cada um. Todas as taxas de compressão são então armazenadas em um vetor. Esse vetor é passado como argumento para o método *run* da classe *Bootstrap* para a geração dos intervalos de confiança, que é representado pela classe *Interval*. Na Fig. 3 é mostrado o diagrama de sequência para a execução desse módulo no *framework*.

O *framework* foi modelado para separar os arquivos por tipos. Sendo assim, no fim de sua execução é gerado um arquivo .csv com os registros dos dados obtidos: o algoritmo usado, o tipo dos arquivos comprimidos, os valores mínimo e máximo do intervalo de confiança referente a essa emulação.

Para cada algoritmo na lista de algoritmos que foi testada, comprime-se cada arquivo na lista de arquivos carregada nas configurações. De cada compressão executada, obtém-se a taxa de compressão e ela é salva num vetor. Esse vetor é usado como entrada no método do *bootstrap* e um intervalo de confiança referente à técnica de compressão utilizada é retornado. No Algoritmo 2 é mostrado o passo-a-passo da execução do *framework*.

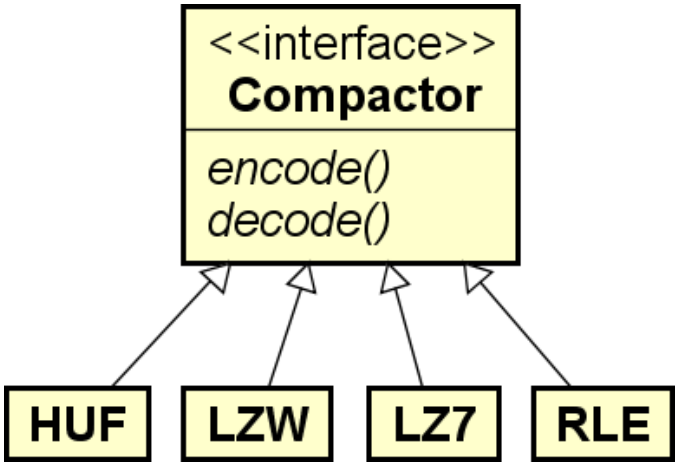



Fig. 2. Diagrama de classes do relacionamento entre as classes que encapsulam os algoritmos de compressão.

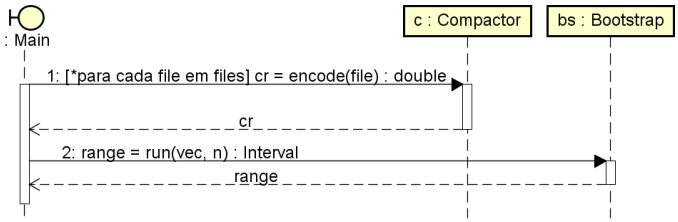



Fig. 3. Diagrama de sequência da execução do framework.

**Algoritmo 2** Execução do *framework*

**Require:** Lista de arquivos $files$ a serem comprimidos e lista de algoritmos de compressão $algorithms$ a serem testados;

**Ensure:** $n$ intervalos de confiança, onde $n$ = número de tipos de arquivos testados * número de algoritmos utilizados.

```
for cada algoritmo alg : algorithms do
    for cada arquivo arq : files do
        compressionRatio ← alg.encode(arq)
        compressionRatios.add(compressionRatio)
    end for
    range ← bootstrap.run(compressionRatios)
end for
```

### B. Estudo de Caso

Como estudo de caso para o *framework* proposto, foram obtidos 400 arquivos de quatro tipos diferentes igualmente divididos em quantidade: vídeo, áudio, imagem e texto, com os respectivos formatos: avi, wav, tiff e txt. Esses formatos foram escolhidos por usarem quase ou nenhuma compressão ao apresentarem os dados em sua forma mais bruta possível. Os arquivos de texto utilizados foram trechos extraídos do

livro *The Gospel of Budha* disponível em uma base de dados pública no *Kaggle*[1]. Os arquivos de vídeo e áudio foram recortados de uma base pública que contém vídeos amadores e animações com quase ou nenhuma edição disponibilizada pelo *Google*[2]. Os arquivos de imagem usados foram obtidos de uma amostragem aleatória da base *Caltech101*[3].

Os algoritmos de compressão que implementam a interface *Compactor* escolhidos foram os já citados: codificação de *Huffman*, *LZW*, *LZ77* e *RLE*. Para a leitura dos arquivos, foi considerada como se fossem arquivos binários e o tamanho da palavra foi definido em um byte. Para a execução do algoritmo *bootstrap*, foram consideradas 2.000 amostras *bootstrap* e um nível de confiança de 95%. Esses parâmetros podem variar a depender do *tradeoff* entre desempenho e eficácia, visto que um maior número de amostras e maior nível de confiança tornam o processo mais demorado mas garantem maior fidedignidade na análise dos critérios de comparação adotados.

### C. Resultados

Para cada amostra de 100 arquivos de um tipo, foi gerada uma tabela com 4 registros contendo o algoritmo de compressão usado e o intervalo de confiança referente a ele, junto de um gráfico representando esses intervalos. A variação do intervalo $\Delta$ foi definida como a diferença absoluta entre os extremos máximo, $b$, e mínimo, $a$, e a variação relativa $\Delta_{\%}$ como a variação em relação ao extremo mínimo em termos percentuais conforme as Equações (1) e (2). Nas tabelas apresentadas a seguir, referentes a essas estatísticas, os algoritmos com melhor desempenho estão em negrito.

$$\Delta = b - a \tag{1}$$

$$\Delta_{\%} = \frac{\Delta}{a} \times 100\% \tag{2}$$

Na Tabela I são apresentadas as estatísticas geradas para os arquivos de vídeo. Observa-se que os algoritmos que apresentaram melhores taxas de compressão foram a codificação de *Huffman* e o RLE e que esses apresentam desempenhos similares visto que seus intervalos se interceptam. Contudo, por apresentar uma menor variação relativa, o RLE se manteve mais estável. Os outros algoritmos, LZ7 e LZW, apresentaram uma taxa de compressão abaixo de 1, já que o arquivo comprimido gerado ficou maior que o original e apresentaram uma instabilidade nos intervalos superior aos demais. No geral, constata-se que o *overhead* de desempenho gerado pelas técnicas de compressão sem perdas não compensam um ganho de armazenamento de aproximadamente, no máximo, 1,72%. Na Fig. 4 é mostrado o gráfico para esses intervalos de confiança. A diferença de desempenho e estabilidade dos dois melhores casos se torna explícita em relação aos dois piores.

Na Tabela II são reveladas as estatísticas geradas para os arquivos de áudio. Observa-se que os algoritmos que apresentaram melhores taxas de compressão foram a codificação

TABELA I
ESTATÍSTICAS GERADAS PARA OS ARQUIVOS DE VÍDEO

| Algoritmo | Mínimo | Máximo | Variação | Variação relativa |
|---|---|---|---|---|
| **HUF** | **1,010780** | **1,017160** | **0,006380** | **0,63%** |
| LZ7 | 0,768323 | 0,790052 | 0,021729 | 2,83% |
| LZW | 0,827177 | 0,854179 | 0,027002 | 3,26% |
| **RLE** | **1,010500** | **1,016360** | **0,005860** | **0,58%** |

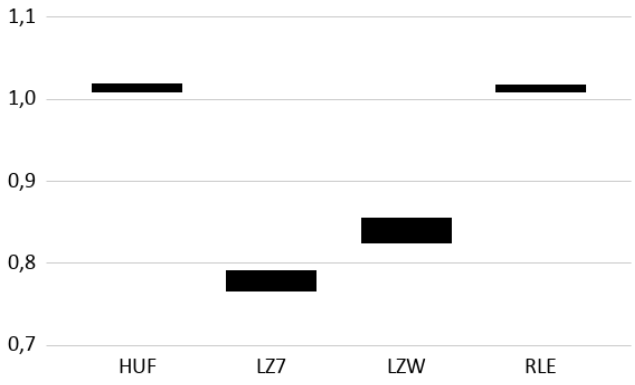



Fig. 4. Intervalos de confiança gerados para os arquivos de vídeo.

de *Huffman* e o RLE. Embora seus intervalos se interceptem por pouco, já que o extremo máximo do intervalo de RLE é ligeiramente maior que o extremo mínimo do intervalo de *Huffman*, todo o intervalo de *Huffman* está acima de um, o que implica que o algoritmo comprimiu eficientemente esse tipo de arquivo. O mesmo não pode ser dito do RLE visto que seu extremo mínimo se apresenta menor que 1. Além de comprimir melhor, a estabilidade de *Huffman* se apresentou melhor que a do RLE. Os outros dois algoritmos retornaram uma taxa de compressão abaixo de 1, além de apresentarem maior instabilidade em relação a seus concorrentes, o que os torna ineficientes diante desse tipo de arquivo. Na Fig. 5 é exibido o gráfico para esses intervalos de confiança.

Na Tabela III são retratadas as estatísticas geradas para os arquivos de imagem. Mesmo o RLE apresentando o melhor intervalo de confiança, nenhum dos algoritmos conseguiu gerar intervalos acima de 1, o que indica que, da forma como estão implementados e processando os arquivos, eles não foram desenhados para comprimir imagens de forma ótima. Além de seu desempenho superior aos demais, sua estabilidade também provou ser a melhor, com apenas 0,02% de variação relativa nos extremos do intervalo. Na Fig. 6 é apresentado o gráfico para esses intervalos de confiança.

Na Tabela IV são exibidas as estatísticas geradas para os arquivos de texto. Embora o RLE tenha tido menor instabilidade nos extremos de seu intervalo, foi o LZW que obteve melhor desempenho em relação aos demais devido a seu intervalo não se interceptar com nenhum outro. Em relação aos outros tipos de arquivos, esses algoritmos, de maneira geral, performaram melhor pois tomaram vantagem de processarem os arquivos byte a byte, que é o tamanho de um char. Na Fig. 7 é mostrado o gráfico para esses intervalos de confiança.

[1]https://www.kaggle.com/tentotheminus9/religious-and-philosophical-texts#35895-0.txt

[2]https://research.google.com/audioset///eval/music.html

[3]http://www.vision.caltech.edu/Image_Datasets/Caltech101/

TABELA II
ESTATÍSTICAS GERADAS PARA OS ARQUIVOS DE ÁUDIO

| Algoritmo | Mínimo | Máximo | Variação | Variação relativa |
|---|---|---|---|---|
| **HUF** | **1,071200** | **1,129300** | **0,058100** | **5,42%** |
| LZ7 | 0,813681 | 0,862646 | 0,048965 | 6,02% |
| LZW | 0,878650 | 0,978875 | 0,100225 | 11,41% |
| **RLE** | **0,996895** | **1,071230** | **0,074335** | **7,46%** |

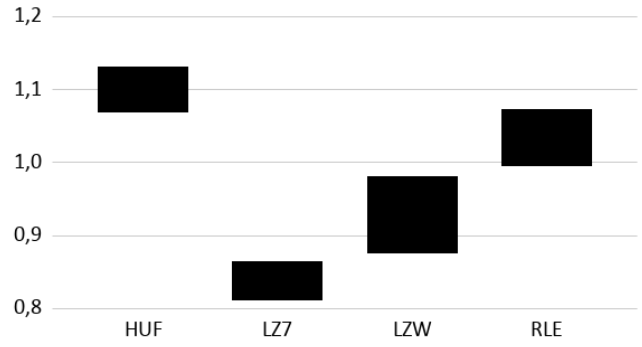



Fig. 5. Intervalos de confiança gerados para os arquivos de áudio.

TABELA III
ESTATÍSTICAS GERADAS PARA OS ARQUIVOS DE IMAGEM

| Algoritmo | Mínimo | Máximo | Variação | Variação relativa |
|---|---|---|---|---|
| HUF | 0,980639 | 0,986402 | 0,005763 | 0,59% |
| LZ7 | 0,719733 | 0,722419 | 0,002686 | 0,37% |
| LZW | 0,676890 | 0,685077 | 0,008187 | 1,21% |
| **RLE** | **0,995823** | **0,996023** | **0,000200** | **0,02%** |

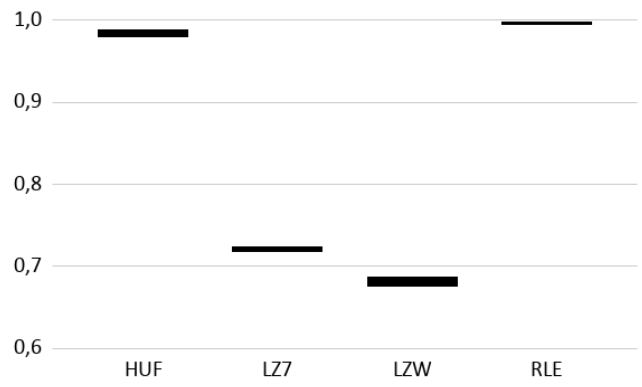



Fig. 6. Intervalos de confiança gerados para os arquivos de imagem.

TABELA IV
ESTATÍSTICAS GERADAS PARA OS ARQUIVOS DE TEXTO

| Algoritmo | Mínimo | Máximo | Variação | Variação relativa |
|---|---|---|---|---|
| HUF | 1,373700 | 1,450500 | 0,076800 | 5,59% |
| LZ7 | 1,432520 | 1,499610 | 0,067090 | 4,68% |
| **LZW** | **1,717010** | **1,771740** | **0,054730** | **3,19%** |
| RLE | 1,119150 | 1,137750 | 0,018600 | 1,66% |

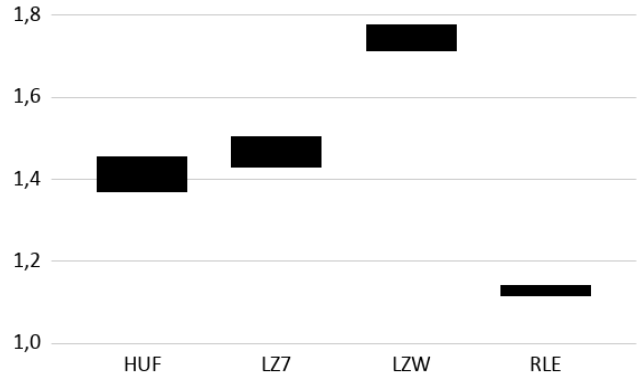



Fig. 7. Intervalos de confiança gerados para os arquivos de texto.

## V. CONCLUSÃO

Este trabalho propôs a utilização de um método estatístico para analisar algoritmos de compressão entre si perante um ou mais critérios quantitativamente mensuráveis. Foi implementado em um *framework* composto por dois módulos: o compressor e o estatístico. Como estudo de caso, para o módulo compressor foram analisados 4 algoritmos de compressões: codificação de *Huffman*, LZW, RLE e LZ77. O parâmetro de comparação utilizado para análise foi a taxa de compressão. Já para o módulo estatístico, foi utilizado o método *bootstrap* para a geração de intervalos de confiança usados na análise dos métodos. Os testes foram realizados conforme o tipo de arquivo. Para arquivos de vídeo, *Huffman* e RLE performaram melhor que os outros, mas apresentaram poucos ganhos em compressão; para os de áudio, os mesmos performaram melhor, mas *Huffman* manteve-se superior por não haver comprimido com taxa de compressão menor que 1; para os de imagem, RLE foi o melhor mas nenhum obteve taxas de compressão maior que 1 e para os de texto, LZW superou os outros.

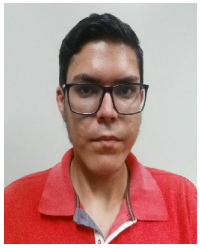
O *framework* proposto tem grande pontencial para ser utilizado na tomada de decisão automatizada em contextos nos quais a escolha do método de compressão se faz necessária e uma mais eficiente é vantajosa. Neste sentido, propõe-se como trabalhos futuros a sua integração para que este incorpore a escolha automática do melhor algoritmo de compressão a ser utilizado nesses contextos. Contudo, a análise feita ainda é considerada de uma perspectiva estática. Após a execução do módulo de compressão, os dados que servem de entrada para a análise estática, ou seja, a lista de parâmetros de análise, não são atualizados. Isso implica ter que rodar novamente o módulo estatístico após o acréscimo de novos dados decorrentes do módulo de compressão ou de um novo algoritmo na lista de algoritmos.

## REFERÊNCIAS

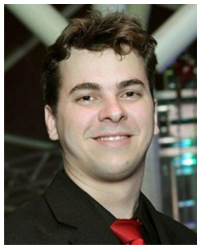
[1] S. Frankel and S. Krishnan, "Ip security (ipsec) and internet key exchange (ike) document roadmap," Internet Requests for Comments, RFC Editor, RFC 6071, February 2011.

[2] A. Freier, P. Karlton, and P. Kocher, "The secure sockets layer (ssl) protocol version 3.0," Internet Requests for Comments, RFC Editor, RFC 6101, August 2011.

[3] A. B. Jambek and N. A. Khairi, "Performance comparison of huffman and lempel-ziv welch data compression for wireless sensor node application," *American Journal of Applied Sciences*, vol. 11, no. 1, pp. 119–126, 2014. [Online]. Available: https://thescipub.com/abstract/ajassp.2014.119.126
[4] A. Gupta, A. Bansal, and V. Khanduja, "Modern lossless compression techniques: Review, comparison and analysis," in *2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT)*. IEEE, 2017, pp. 1–8.
[5] T. Hidayat, M. H. Zakaria, and N. C. Pee, "Comparison of lossless compression schemes for WAV audio data 16-bit between huffman and coding arithmetic," *International journal of simulation: systems, science & technology*, Feb. 2019. [Online]. Available: https://doi.org/10.5013/ijssst.a.19.06.36
[6] M. A. Rahman and M. Hamada, "Lossless image compression techniques: A state-of-the-art survey," *Symmetry*, vol. 11, no. 10, p. 1274, Oct. 2019. [Online]. Available: https://doi.org/10.3390/sym11101274
[7] B. Efron and R. Tibshirani, *An Introduction to the Bootstrap*, ser. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis, 1994.
[8] A. Hussain, A. Al-Fayadh, and N. Radi, "Image compression techniques: A survey in lossless and lossy algorithms," *Neurocomputing*, vol. 300, pp. 44–69, 2018.
[9] K. Kalajdzic, S. H. Ali, and A. Patel, "Rapid lossless compression of short text messages," *Computer Standards & Interfaces*, vol. 37, pp. 53–59, 2015.
[10] G. Campobello, O. Giordano, A. Segreto, and S. Serrano, "Comparison of local lossless compression algorithms for wireless sensor networks," *Journal of Network and Computer Applications*, vol. 47, pp. 23–31, 2015.
[11] A. Antony and S. Ganapathy, "Highly efficient near lossless video compression using selective intra prediction for hevc lossless mode," *AEUE - International Journal of Electronics and Communications*, vol. 69, no. 11, pp. 1650–1658, 2015.
[12] F. Ghido and I. Tabus, "Sparse modeling for lossless audio compression," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 1, pp. 14–28, 2013.
[13] A. Masmoudi and W. Puech, "Lossless chaos-based crypto-compression scheme for image protection," *IET Image Processing*, vol. 8, no. 12, pp. 671–686, 2014.
[14] C.-C. Chang, T.-C. Lu, G. Horng, and Y.-H. Huang, "Very efficient variable-length codes for the lossless compression of vq indices," *Multimedia Tools and Applications*, vol. 75, no. 6, pp. 3537–3552, 2016.
[15] M. T. Asif, K. Srinivasan, N. Mitrovic, J. Dauwels, and P. Jaillet, "Near-lossless compression for large traffic networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 4, pp. 1817–1826, 2015.
[16] L. C. Souza, R. M. Souza, G. J. Amaral, and T. M. S. Filho, "A parametrized approach for linear regression of interval data," *Knowledge-Based Systems*, vol. 131, pp. 149 – 159, 2017.
[17] L. C. de Souza, R. M. C. R. de Souza, and G. J. A. do Amaral, "Dynamic clustering of interval data based on hybrid $L_q$ distance," *Knowledge and Information Systems*, May 2019.
[18] C. McAnlis and A. Haecky, *Understanding Compression: Data Compression for Modern Developers*. O'Reilly Media, 2016.
[19] Z. Li and M. Drew, *Fundamentals of Multimedia*. Pearson Prentice Hall, 2004.
[20] D. Salomon, G. Motta, and D. Bryant, *Data Compression: The Complete Reference*. Springer London, 2007.
[21] A. C. Davison and D. V. Hinkley, *Bootstrap Methods and their Application*, ser. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1997.
[22] W. L. Martinez and A. R. Martinez, *Computational statistics handbook with MATLAB*. Chapman and Hall/CRC, 2015.



**Antonio Alessandro Rocha Beserra** tem bacharelado em Ciência da Computação pela Universidade Federal Rural do Semi-Árido (2019), atualmente é mestrando em Ciência da Computação e Matemática Computacional pela Universidade de São Paulo. http://lattes.cnpq.br/9361804202244207



**Daniel Faustino Lacerda de Souza** tem bacharelado em Ciência da Computação pela Universidade Federal da Paraíba (2008), mestrado em Informática pela Universidade Federal da Paraíba (2010) e doutorado em Engenharia Elétrica e da Computação pela Universidade Federal do Rio Grande do Norte (2017). http://lattes.cnpq.br/7175882793842898

**Leandro Carlos de Souza** tem bacharelado em Ciências da Computação pela Universidade Federal da Paraíba (2008), mestrado em Matemática pela Pontifícia Universidade Católica do Rio de Janeiro (2011) e doutorado em Ciências da Computação pela Universidade Federal de Pernambuco (2016). http://lattes.cnpq.br/7894153744845649