UNIVERSIDADE FEDERAL DO PAMPA

FERNANDO ANTÔNIO ALVES BERTOLDI

ANS ADAPTATIVO: UMA ABORDAGEM PARA CODIFICAÇÃO EM TEMPO REAL

Bagé
2025

FERNANDO ANTÔNIO ALVES BERTOLDI

ANS ADAPTATIVO: UMA ABORDAGEM PARA CODIFICAÇÃO EM TEMPO REAL

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Engenharia de Computação como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação.

Orientador: Fábio Luís Livi Ramos

Bagé
2025

Ficha catalográfica elaborada automaticamente com os dados fornecidos pelo(a) autor(a) através do Módulo de Biblioteca do Sistema GURI (Gestão Unificada de Recursos Institucionais).

B546a Bertoldi, Fernando Antônio Alves

ANS Adaptativo: Uma Abordagem para Codificação em Tempo Real / Fernando Antônio Alves Bertoldi.

110 f.: il.

Orientador: Fábio Luís Livi Ramos

Trabalho de Conclusão de Curso (Graduação) - Universidade Federal do Pampa, Engenharia de Computação, 2025.

1. Compressão de dados. 2. Codificação entrópica. 3. TANS. 4. Codificação adaptativa. I. Título.

FERNANDO ANTÔNIO ALVES BERTOLDI

ANS ADAPTATIVO: UMA ABORDAGEM PARA CODIFICAÇÃO EM TEMPO REAL

Trabalho de Conclusão de Curso apresentado ao curso de Engenharia de Computação como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação.

Dissertação defendida e aprovada em: 8 de dezembro de 2025.

Banca examinadora:

_____

Prof. Dr. Fábio Luís Livi Ramos

Orientador

(UNIPAMPA)

_____

Prof. Dr. Bruno Silveira Neves

(UNIPAMPA)

_____

Prof. Dr. Carlos Michel Betemps

(UNIPAMPA)

Assinado eletronicamente por **FABIO LUIS LIVI RAMOS**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 18/12/2025, às 18:58, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **CARLOS MICHEL BETEMPS**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 18/12/2025, às 20:42, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

Assinado eletronicamente por **BRUNO SILVEIRA NEVES**, **PROFESSOR DO MAGISTERIO SUPERIOR**, em 19/12/2025, às 16:22, conforme horário oficial de Brasília, de acordo com as normativas legais aplicáveis.

A autenticidade deste documento pode ser conferida no site https://sei.unipampa.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **1919674** e o código CRC **8736B0B4**.

Dedico este trabalho ao meu fiel companheiro Xis, que esteve ao meu lado em muitos momentos importantes da minha vida. Mesmo após sua partida, sua presença continua viva em minha memória e em meu coração. Que essa conquista também seja sua.

AGRADECIMENTO

A conclusão deste trabalho representa não apenas o fim de uma etapa acadêmica, mas também a realização de um percurso marcado por desafios, conquistas e, principalmente, por pessoas que estiveram ao meu lado e contribuíram para que esse momento se tornasse possível.

Agradeço, em primeiro lugar, aos meus pais, Ricardo Bertoldi e Carla Bica, pelo amor, apoio constante e pelos valores que sempre me transmitiram. Aos meus irmãos, Cristiano Cardoso e Péricles Cardoso, pelo incentivo e pela presença em minha vida.

Aos meus amigos Guilherme Becker, Vanessa Kissler, Vitor Hugo Macedo e Marcelo Fajardo, por toda a amizade, companheirismo e pelas palavras de apoio nos momentos difíceis.

Ao meu orientador, professor Fábio Ramos, por toda a orientação, paciência e dedicação durante o desenvolvimento deste trabalho. Seu acompanhamento foi essencial para que este projeto alcançasse seus objetivos.

Aos professores Bruno Neves, Gerson Nunes e Júlio Saraçol, por sua constante disponibilidade, conselhos e incentivo em momentos decisivos da minha formação acadêmica. Suas contribuições foram valiosas para meu crescimento pessoal e profissional.

E deixo também um agradecimento especial ao meu querido cachorro Xis, que faleceu em 2022, mas que esteve presente em muitos momentos da minha vida, deixando lembranças de amor e companheirismo que levarei comigo para sempre.

A todos vocês, minha mais profunda gratidão.

“Seu tempo é limitado, então não o desperdice vivendo a vida de outra pessoa.”
— Steve Jobs

RESUMO

A compressão de dados é uma área fundamental da ciência da computação e da teoria da informação, com aplicações que vão desde comunicações digitais até a redução de modelos de redes neurais para dispositivos embarcados. Este trabalho tem como foco o estudo e desenvolvimento do *Asymmetric Numeral Systems* (ANS), uma técnica moderna de codificação entrópica que combina a eficiência da codificação aritmética com a simplicidade computacional da codificação de Huffman. Neste trabalho, é investigada a variante tabelada do ANS, conhecida como *tANS*, com ênfase em sua implementação prática e geração automatizada das tabelas de codificação e decodificação. O trabalho aborda os fundamentos teóricos da codificação entrópica, discute a evolução dos algoritmos clássicos como Huffman e aritmética, e introduz os princípios matemáticos e computacionais do ANS. Como resultados, foi desenvolvido um compressor tANS adaptativo funcional em linguagem C capaz de fazer o processo de codificação e decodificação e construir saídas do bitstream. Entre os desafios enfrentados, destaca-se a complexidade conceitual da técnica e a necessidade de adequação do decodicador as trocas do modelo de probabilidade, para que seja possível seguir o mesmo caminho do codificador. A proposta visa atender demandas de compressão eficiente em ambientes com restrições de recursos e baixa latência.

Palavras-chave: Compressão de dados. Codificação entrópica. tANS. Codificação adaptativa.

ABSTRACT

Data compression is a fundamental field of computer science and information theory, with applications ranging from digital communications to the reduction of neural network models for embedded devices. This work focuses on the study and development of the *Asymmetric Numeral Systems* (ANS), a modern entropy coding technique that combines the efficiency of arithmetic coding with the computational simplicity of Huffman coding. In this research, the table-based variant of ANS, known as *tANS*, is investigated with emphasis on its practical implementation and the automatic generation of encoding and decoding tables. The work discusses the theoretical foundations of entropy coding, reviews the evolution of classical algorithms such as Huffman and arithmetic coding, and introduces the mathematical and computational principles of ANS. As results, a functional adaptive tANS compressor in C was developed, capable of performing both encoding and decoding processes and generating bitstream outputs. Among the challenges faced, the conceptual complexity of the technique stands out, as well as the need to ensure that the decoder appropriately follows the probability model updates performed during encoding. The proposal aims to address efficient compression demands in resource-constrained and low-latency environments.

Keywords: Data compression; Entropy coding; tANS; Adaptive coding.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

4G/5G	Redes móveis de quarta e quinta geração
AC	*Arithmetic Coding* (Codificação Aritmética)
ANS	*Asymmetric Numeral Systems*
ASIC	*Application-Specific Integrated Circuit*
AV1	*AOMedia Video 1* (Padrão de compressão de vídeo)
BAE	*Binary Arithmetic Encoder* (Codificador Aritmético Binário)
CABAC	*Context-Adaptive Binary Arithmetic Coding* (Codificação Aritmética Binária Adaptativa ao Contexto)
CDF	*Cumulative Distribution Function* (Função de Distribuição Acumulada)
CPU	*Central Processing Unit*
GPU	*Graphics Processing Unit*
H.264	Padrão de compressão de vídeo *Advanced Video Coding*
H.265	Padrão de compressão de vídeo *High Efficiency Video Coding* (HEVC)
HEVC	*High Efficiency Video Coding* (Codificação de Vídeo de Alta Eficiência)
Huffman	Codificação de Huffman
IoT	*Internet of Things* (Internet das Coisas)
JPEG	*Joint Photographic Experts Group* (Padrão de compressão de imagem)
JPEG XL	Novo padrão de compressão de imagem sucessor do JPEG
MBBS	*Multiple Bypass Bin Processing* (Processamento Múltiplo de Bins Bypass)
PDF	*Probability Density Function* (Função Densidade de Probabilidade)
PMF	*Probability Mass Function* (Função Massa de Probabilidade)
RAM	*Random Access Memory*
rANS	*Range-based Asymmetric Numeral Systems*
TCC	Trabalho de Conclusão de Curso
tANS	*Table-based Asymmetric Numeral Systems*

UNIPAMPA Universidade Federal do Pampa

VLSI *Very Large Scale Integration* (Integração em Larga Escala)

SUMÁRIO

1 INTRODUÇÃO

A codificação entrópica representa um dos pilares fundamentais da teoria da informação moderna, cujas origens remontam aos trabalhos seminais de Claude Shannon em 1948 (SHANNON, 1948). Em sua obra, Shannon estabeleceu que a entropia de uma fonte de informação define um limite teórico inferior para a taxa de compressão possível, representando a quantidade média mínima de bits necessários para codificar cada símbolo sem perda de informação. Este marco teórico não apenas formalizou o conceito de compressão de dados, como também estabeleceu as bases para o desenvolvimento de algoritmos cada vez mais eficientes.

As técnicas de codificação entrópica visam representar informações com o menor número possível de bits, aproximando-se desse limite definido pela entropia. Na prática, essas técnicas são amplamente aplicadas em cenários críticos como transmissão de vídeo digital (padrões H.264, H.265, AV1 (BROSS et al., 2021)), armazenamento em nuvem, comunicações sem fio (4G/5G) e compressão de modelos de aprendizado de máquina para dispositivos de borda (CHENG et al., 2018; HAN et al., 2016).

Neste contexto, o presente trabalho investiga o Sistema Numérico Assimétrico - *Asymmetric Numeral Systems* (ANS) como um novo método promissor na compressão de dados, com foco em sua adaptação para cenários dinâmicos. O ANS se destaca por combinar:

- Eficiência de compressão comparável à codificação aritmética (DUDA, 2013)
- Baixa complexidade computacional, próxima à codificação de Huffman
- Capacidade de paralelização em hardware e GPUs
- Flexibilidade para implementações adaptativas e *streamings* de baixa latência

Enquanto a codificação de Huffman é limitada por representações de código com número inteiro de bits, e a codificação aritmética sofre com alta complexidade e dificuldade de paralelização, o ANS surge como uma alternativa intermediária e eficaz. Diferentemente dos métodos tradicionais, que operam sobre símbolos individuais, o ANS codifica informações como estados de um sistema numérico dinâmico. Essa abordagem, proposta por Duda (DUDA, 2013), combina a elegância matemática da codificação aritmética com operações computacionalmente eficientes (operações de *shift* e *lookup*), sendo particularmente adequada para cenários modernos que exigem paralelização e baixa latência.

1.1 Problema de Pesquisa

Apesar das vantagens do ANS, sua aplicação em cenários dinâmicos enfrenta um desafio crítico: a dependência de distribuições de probabilidade estáticas. Essa limitação reduz sua eficácia em aplicações como:

- **Transmissão de vídeo adaptativa em tempo real**: Em uma sequência de vídeo, mudanças abruptas de cena (ex.: de uma paisagem estática para uma ação rápida) alteram drasticamente a distribuição estatística dos pixels. Um ANS estático, treinado em dados anteriores, torna-se subótimo, gerando redundância ou perda de eficiência na compressão.
- **Processamento de fluxos de dados não estacionários**: Em sensores IoT ou redes de comunicação, variações temporais (ex.: picos de tráfego) exigem ajustes contínuos nas probabilidades para manter a compressão eficiente.

O problema central desta pesquisa consiste em desenvolver um ANS adaptativo eficiente que permita ajustes dinâmicos nas distribuições de probabilidade sem comprometer sua eficiência computacional característica, atendendo a requisitos de aplicações em tempo real e sistemas embarcados. O desafio está na estrutura da codificação ANS, especialmente em variantes como o tANS. Nesta variante, ocorre o fato de que os símbolos são decodificados em ordem inversa àquela em que foram codificados. Essa característica impõe restrições em aplicações que exigem processamento sequencial, como fluxos contínuos de dados e transmissão de vídeo em tempo real, dificultando a adoção direta do método sem estratégias específicas de reordenação.

1.2 Objetivo Geral

Projetar, implementar em software e avaliar um sistema adaptativo para o *Asymmetric Numeral Systems* que permita a atualização dinâmica de probabilidades durante a codificação de símbolos binários de 0s e 1s.

1.3 Objetivos Específicos

Os objetivos específicos deste trabalho incluem:

1. **Projeto do ANS Adaptativo**:
    - Implementação com atualização em tempo real de distribuições de probabilidade, utilizando estimação bayesiana, janelas deslizantes e decaimento exponencial
2. **Otimização Conjunta**:
    - Metodologia de aprendizado que ajuste simultaneamente as distribuições de probabilidade, balanceando taxa de bits e acurácia
3. **Avaliação Experimental**:
    - Testes em domínios variados (vídeo, classificação de imagens em *edge devices*)
4. **Comparação com Métodos Tradicionais**:
    - Análise quantitativa contra Huffman, codificação aritmética e quantização, usando métricas como taxa de compressão, latência e distorção.

1.4 Organização do Trabalho

Este trabalho está organizado em seis capítulos, além das referências bibliográficas ao final. No Capítulo 1, apresenta-se a introdução geral da pesquisa, contextualizando o problema, a motivação, os objetivos gerais e específicos, bem como a estrutura do estudo. O Capítulo 2 descreve a metodologia adotada, contemplando a classificação da pesquisa, os procedimentos experimentais, a abordagem quantitativa utilizada, a delimitação do estudo e as etapas que estruturam o desenvolvimento do sistema de compressão. O Capítulo 3 apresenta o referencial teórico que fundamenta o trabalho, incluindo os conceitos essenciais da teoria da informação, modelos probabilísticos, codificação entrópica tradicional (Huffman e Aritmética) e os trabalhos correlatos. O Capítulo 4 é dedicado ao estudo aprofundado do *Asymmetric Numeral Systems* (ANS), detalhando sua formulação matemática, funcionamento operacional e variantes relevantes, como rANS, streaming-rANS e tANS. São discutidos também os mecanismos de codificação, decodificação, modelagem probabilística e aspectos práticos de implementação. No Capítulo 5, são apresentados o desenvolvimento do sistema tANS dinâmico, sua estrutura interna, o funcionamento do encoder e decoder adaptativos, assim

como a análise dos resultados dos testes. São discutidos o comportamento das diferentes configurações de *range* e intervalos adaptativos, a eficiência obtida e as implicações práticas sobre estabilidade, compressão e convergência estatística. Por fim, o Capítulo 6 apresenta as considerações finais, discutindo as contribuições da pesquisa, o potencial inovador do compressor desenvolvido, as limitações identificadas e possíveis trabalhos futuros. Após esses capítulos, são listadas as Referências, que reúnem todas as fontes utilizadas para fundamentar e desenvolver o trabalho.

2 METODOLOGIA

2.1 Classificação da Pesquisa

O presente trabalho se enquadra dentro de uma abordagem metodológica de pesquisa aplicada, exploratória e quantitativa, com forte componente de desenvolvimento experimental e computacional, conforme as classificações propostas por Gil (GIL, 2008) e Prodanov e Freitas (PRODANOV; FREITAS, 2013).

2.1.1 Natureza: Pesquisa Aplicada

Este estudo caracteriza-se como uma pesquisa aplicada, uma vez que busca gerar conhecimentos voltados à aplicação prática de técnicas avançadas de compressão de dados em contextos de computação embarcada e sistemas de transmissão de dados dinâmicos. O objetivo central não é apenas a ampliação do conhecimento teórico, mas sim o desenvolvimento de soluções computacionais que possam ser integradas a sistemas reais, contribuindo para a eficiência e desempenho de aplicações emergentes em áreas como edge computing, transmissão adaptativa de vídeo e cenários onde existam equipamentos com baixo poder computacional.

2.1.2 Objetivos: Pesquisa Exploratória e Descritiva

A pesquisa apresenta um caráter exploratório, pois investiga possibilidades de aprimoramento do *Asymmetric Numeral Systems* (ANS) em cenários adaptativos, uma área ainda em desenvolvimento e com poucos estudos consolidados na literatura. A adaptação dinâmica das distribuições de probabilidade no ANS, representa um avanço metodológico ainda não amplamente explorado. Esse cenário motivou o surgimento de um conjunto de questões norteadoras desta pesquisa, entre as quais destacam-se: seria possível projetar um codificador baseado em ANS que atualiza suas distribuições de probabilidade por lote? Qual seria o tamanho ideal desse lote para garantir um bom equilíbrio entre desempenho computacional e precisão estatística? Qual é o ganho real de acurácia e eficiência de compressão ao adotar essa abordagem em comparação com modelos estáticos?

Simultaneamente, possui também caráter descritivo, ao documentar e analisar o comportamento do ANS adaptativo sob diferentes cenários experimentais. A pesquisa descreve com detalhamento as técnicas empregadas, os algoritmos desenvolvidos e os resultados obtidos durante os experimentos de compressão em contextos variados (quantidade de símbolos e a forma como estão distribuídos).

2.1.3 Procedimentos: Pesquisa Experimental e Desenvolvimento Computacional

Em termos de procedimentos técnicos, este trabalho adota uma abordagem experimental e de desenvolvimento computacional. São desenvolvidos algoritmos e sistemas de codificação adaptativa, os quais são implementados, integrados e testados em ambientes de simulação.

O desenvolvimento inclui:

- Implementação de algoritmos adaptativos para o ANS com atualização dinâmica de probabilidades;

- Definição de métricas e protocolos de avaliação quantitativa de desempenho (eficiência de compressão, latência, impacto na acurácia dos modelos comprimidos, etc.);

- Execução de experimentos controlados para comparação com métodos tradicionais de compressão.

2.1.4 Abordagem do problema: Pesquisa Quantitativa

A abordagem predominante da pesquisa é quantitativa, pois os resultados são avaliados com base em métricas objetivas e mensuráveis, como taxa de compressão (bits por símbolo), latência de codificação/decodificação, overhead computacional e impacto no desempenho de inferência dos modelos comprimidos. Os experimentos geram dados numéricos que são analisados estatisticamente para validar as hipóteses propostas e comparar o desempenho das soluções implementadas.

2.1.5 Delimitação do estudo

Embora o ANS possua ampla aplicabilidade em diversas áreas de compressão de dados, este trabalho delimita seu escopo à compressão de simbolos e fluxos de dados com comportamento dinâmico, com foco em aplicações de computação em borda e transmissão adaptativa, onde as exigências de baixa latência e eficiência energética são particularmente críticas.

2.2 Etapas da pesquisa

A presente pesquisa adota uma metodologia composta por estudo teórico, desenvolvimento computacional e validação funcional de um sistema inicial de compressão baseado no algoritmo *Asymmetric Numeral Systems* (ANS) com tabelas pré-computadas. Na Figura 1, encontra-se um fluxograma que apresenta, de forma estruturada, os passos executados ao longo deste trabalho.

Inicialmente, foi realizada uma revisão teórica abrangente sobre algoritmos de codificação entrópica e, em especial, sobre o funcionamento do tANS, seus princípios matemáticos e aplicações em sistemas modernos de compressão. Essa etapa fundamentou a compreensão do mecanismo de estados, normalização e operação de *push*/*pop* de símbolos, além de permitir a definição dos parâmetros relevantes para os experimentos, como *range*, distribuição de probabilidades e tamanho da janela adaptativa.

Na sequência, partiu-se para o desenvolvimento computacional, que envolveu a implementação do codificador tANS em linguagem C, a criação das rotinas de geração automática das tabelas (*Symbol Table*, *Decoding Table* e *Next State Table*) e a elaboração de mecanismos de diagnóstico interno para acompanhar o comportamento do codificador a cada atualização do modelo. Essa fase contemplou ainda o desenvolvimento de scripts auxiliares para automação de testes, coleta de métricas e geração de gráficos.

Após a implementação, procedeu-se à etapa de experimentação, na qual diferentes combinações de parâmetros como valores de *range* e tamanhos de intervalos adaptativos foram aplicadas a conjuntos de dados sintéticos. Cada experimento produziu bitstreams e relatórios de execução que permitiram analisar a eficiência, estabilidade e comportamento estatístico do codificador, com foco especial na relação entre entropia teórica, comprimento médio de código e impacto das flutuações adaptativas.

Por fim, realizou-se a validação funcional, verificando a consistência entre

codificação e decodificação e confirmando que as tabelas geradas mantêm reversibilidade perfeita, conforme esperado para sistemas ANS bem formados. Os resultados obtidos, aliados à análise comparativa entre diferentes configurações, possibilitaram consolidar o entendimento das limitações e potencialidades do uso de tANS adaptativo com tabelas pré-computadas.

Fluxograma – Metodologia do TCC2

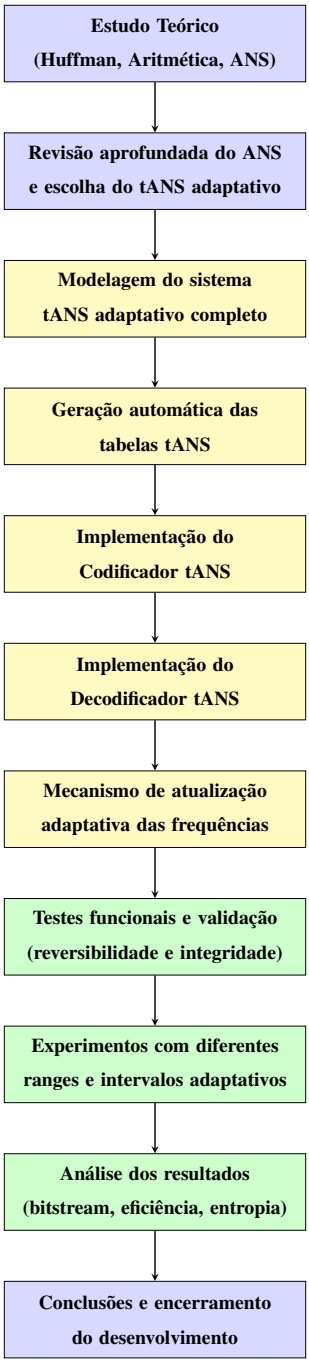

Figura 1 – Fluxograma das etapas da metodologia do TCC2.

Fonte: Autor (2025).

2.2.1 Revisão da Literatura

A revisão da literatura deste trabalho foi conduzida com o objetivo de mapear o estado da arte sobre compressão de dados utilizando *Asymmetric Numeral Systems* (ANS), bem como identificar abordagens existentes que tratam da adaptação dinâmica de modelos probabilísticos em codificação entrópica. A estratégia de busca foi estruturada em etapas, com a utilização combinada de mecanismos de busca abertos e bases de dados científicas reconhecidas.

Inicialmente, foram realizadas buscas exploratórias nas seguintes fontes:

- Google Scholar
- IEEE Xplore
- ACM Digital Library
- SpringerLink
- ScienceDirect

O objetivo das buscas foi identificar publicações relevantes que abordassem:

- O *Asymmetric Numeral Systems* (ANS) e suas variantes, especialmente o rANS e o tANS;
- Aplicações do ANS em ambientes diferentes;
- Métodos de atualização dinâmica de distribuições de probabilidade em tempo de execução.

Foram utilizados os seguintes termos de busca:

- `adaptive entropy coding`
- `asymmetric numeral systems with dynamic updates`
- `probability model adaptation for entropy coding`

Durante o estudo, observou-se uma lacuna significativa na literatura quanto à aplicação de mecanismos adaptativos no ANS, particularmente no que diz respeito à atualização de probabilidades por lote ou em tempo real.

Além da busca por artigos científicos, foram consultados materiais técnicos e repositórios de código-fonte que contêm implementações de algoritmos ANS, especialmente em linguagens como C e C++, com o intuito de compreender melhor os detalhes de implementação do ANS tabelado (tANS) e identificar oportunidades de inserção de mecanismos adaptativos. Ferramentas como GitHub (GitHub, Inc., 2025) e GitLab (GitLab Inc., 2025) foram utilizadas como apoio complementar para investigar implementações práticas e bibliotecas *open-source* relacionadas ao tema.

A seleção dos materiais considerou a atualidade das publicações, seu número de citações e a relevância direta para os objetivos desta pesquisa. Trabalhos relacionados à compressão de parâmetros de redes neurais, aprendizado de máquina e técnicas de quantização com codificação entrópica foram especialmente priorizados. Essa revisão fundamenta o direcionamento técnico da proposta e embasa a formulação do sistema adaptativo de compressão desenvolvido neste trabalho.

2.2.2 Estudo Teórico e Revisão de Algoritmos

Inicialmente, foi realizado um estudo aprofundado dos principais algoritmos de codificação entrópica comumente empregados em sistemas de compressão de dados, incluindo:

- Codificação de Huffman;
- Codificação Aritmética;
- Codificação Asymmetric Numeral Systems (ANS), com ênfase na variante Table-based ANS (tANS).

A partir desta revisão, foram identificadas as vantagens específicas do ANS, particularmente sua elevada eficiência de compressão, baixo custo computacional e facilidade de paralelização, tornando-o um candidato promissor para aplicações futuras em sistemas adaptativos.

2.2.3 Modelagem do Sistema de Compressão

Com base na fundamentação teórica, foi modelado e desenvolvido um sistema completo de compressão baseado no algoritmo Asymmetric Numeral Systems (ANS),

utilizando especificamente a variante tabelada (tANS — *Table-based ANS*). Inicialmente, foi implementado um módulo capaz de gerar automaticamente todas as tabelas necessárias para o funcionamento do algoritmo, incluindo as tabelas de codificação e decodificação. A partir dessa base, foi construído o compressor tANS adaptativo completo, composto por um codificador e um decodificador plenamente funcionais.

Essa abordagem permitiu estruturar o sistema de forma modular e mensurável, possibilitando:

- Validar o funcionamento correto do mecanismo central do ANS;
- Investigar a relação entre distribuições de probabilidade e eficiência de compressão;
- Implementar um compressor funcional, capaz de operar com adaptação periódica das frequências.

2.2.4 Implementação em Linguagem C

A implementação foi realizada integralmente em linguagem C, devido ao seu desempenho elevado e à ampla adoção em algoritmos de compressão de baixo nível. O sistema foi estruturado em módulos, permitindo clareza e reuso de código. Os principais requisitos do tANS estão listados a baixo:

- Definição e leitura do conjunto de símbolos utilizado pelo codificador;
- Cálculo automático das frequências normalizadas e construção do modelo probabilístico;
- Geração das tabelas de codificação (*encoding table*) e decodificação (*decoding table*);
- Execução completa do processo de codificação adaptativa, atualizando o modelo periodicamente conforme definido pelo usuário;
- Decodificação total do bitstream, validando a correta restauração dos dados originais.

Além disso, mecanismos de verificação interna asseguram que os estados gerados pelo codificador correspondam perfeitamente aos estados esperados pelo decodificador, garantindo a integridade do bitstream reconstruído.

2.2.5 Verificação Funcional

Para validar o funcionamento do sistema, foram realizados testes unitários, integrados e funcionais, utilizando diferentes combinações de:

- distribuições de símbolos balanceadas e desbalanceadas;
- valores de *range* utilizados para construir a tabela tANS;
- intervalos adaptativos variados durante a atualização do modelo;
- sequências de entrada com diferentes níveis de entropia.

Os testes verificaram:

- a construção correta das tabelas de codificação e decodificação;
- a integridade do processo de compressão;
- a reversibilidade completa do algoritmo, garantindo que a decodificação recupera exatamente a sequência original;
- o impacto das escolhas de parâmetros sobre o tamanho final do bitstream.

2.2.6 Limitações e Considerações

Embora o sistema final seja um compressor tANS completo, incluindo a adaptação dinâmica das frequências, algumas limitações naturais do método foram observadas, tais como:

- a sensibilidade do desempenho a fontes extremamente assimétricas;
- a dependência da eficiência em relação ao *range* escolhido;
- a necessidade de equilibrar a frequência de atualização do modelo para evitar ruídos estatísticos.

Essas observações, entretanto, não comprometem o funcionamento do compressor, mas enriquecem a análise experimental, permitindo compreender melhor o comportamento do tANS em diferentes cenários.

2.2.7 Estrutura de Módulos do Sistema

A Tabela 1 apresenta uma descrição dos módulos que compõem o sistema de compressão tANS desenvolvido. Esses módulos formam a base estrutural do encoder e do decoder adaptativos, permitindo uma organização clara das responsabilidades de cada componente. Além disso, a divisão modular facilita a manutenção, a extensibilidade do sistema e a compreensão das interações entre as partes que integram o processo completo de codificação e decodificação.

Tabela 1 – Descrição dos módulos do sistema de compressão tANS.

Módulo	**Descrição**
`Symbol`	Representa um símbolo do alfabeto. Armazena o valor do símbolo e sua frequência de ocorrência.
`Probability Model`	Gerencia o conjunto de símbolos, suas frequências e probabilidades normalizadas. Calcula a distribuição acumulada (CDF) usada na construção das tabelas.
`TableBuilder`	Responsável pela geração das tabelas tANS de codificação e decodificação, utilizando as frequências e probabilidades do modelo.
`TANSEncoder`	Implementa o codificador tANS adaptativo, produzindo o bitstream comprimido a partir da sequência de entrada.
`TANSDecoder`	Implementa o decodificador tANS, reconstituindo exatamente a sequência original a partir do bitstream.
`BitstreamIO`	Gerencia a leitura e escrita de bits, garantindo que o codificador e o decodificador operem de forma consistente.

Fonte: Autor (2025).

Além disso, a Figura 2 apresenta o diagrama de classes que sintetiza a interação entre os módulos do sistema.

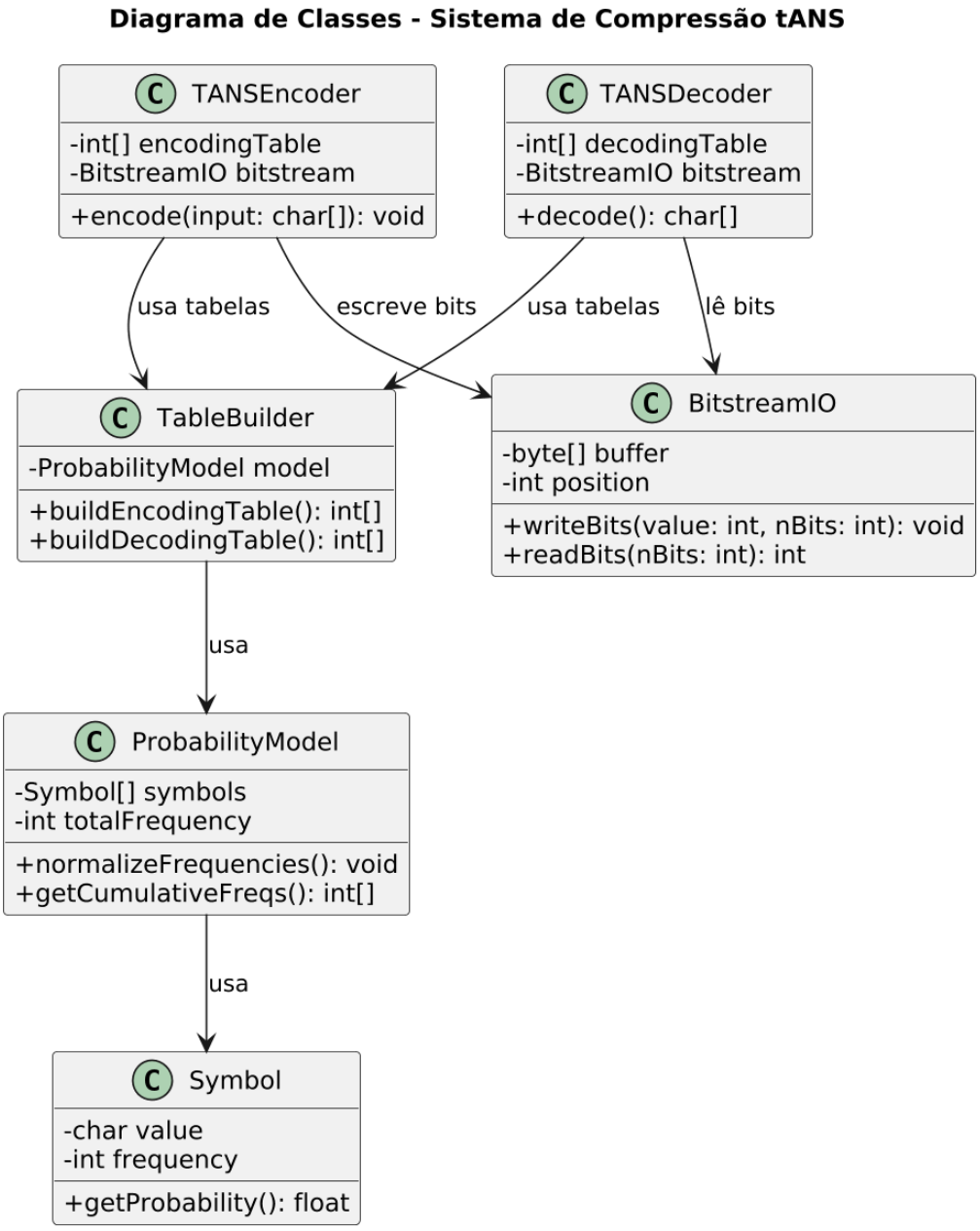

Figura 2 – Diagrama de Classes do Sistema de Compressão ANS Adaptativo

Fonte: Autor (2025).

3 REFERENCIAL TEÓRICO

3.1 Fundamentos da Teoria da Informação

A Teoria da Informação surgiu como um campo formalizado por Claude E. Shannon em sua obra seminal *A Mathematical Theory of Communication*, publicada em 1948 (SHANNON, 1948). Este trabalho fundacional estabeleceu as bases matemáticas para o estudo da transmissão, codificação e compressão de dados, tratando a informação de forma abstrata, desvinculada de seu conteúdo semântico, e centrando-se em sua estrutura estatística. Shannon demonstrou que qualquer fonte de dados pode ser modelada como um processo estocástico, cujas propriedades probabilísticas permitem quantificar a incerteza associada à emissão de símbolos. O paradigma de Shannon ilustrado na Figura 3, descreve como a informação se desloca ao mesmo tempo que introduz conceitos de entropia da informação.

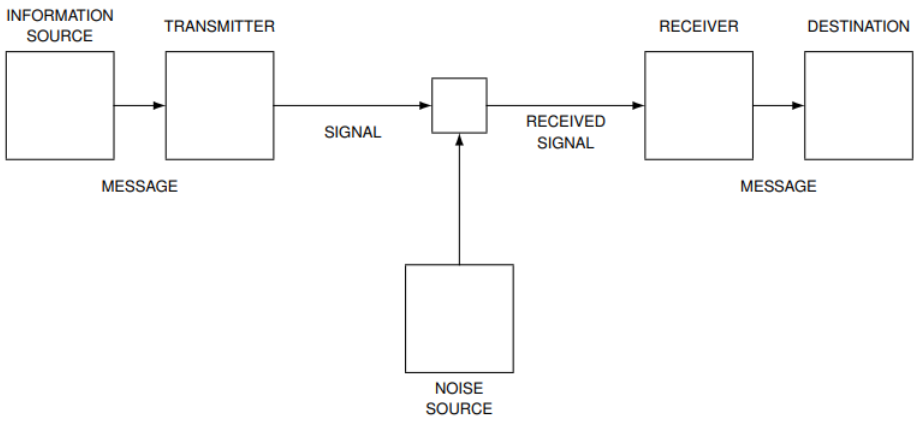

Figura 3 – O Paradigma de Shannon (extraído do artigo de Shannon de 1948).
Fonte: (SHANNON, 1948).

> *"O problema fundamental da comunicação é o de reproduzir em um ponto dado, exatamente ou aproximadamente, uma mensagem selecionada em um outro ponto."* (SHANNON, 1948, p. 1).

O conceito central introduzido por Shannon é o de **entropia** da informação, que mede a quantidade média de incerteza ou surpresa associada a uma fonte emissora. Para uma fonte discreta X, composta por símbolos $x_i \in \mathscr{X}$ com probabilidades $p_i = \mathbb{P}(X = x_i)$, a entropia $H(X)$ é definida como:

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i \tag{1}$$

Esta expressão representa o limite inferior teórico, em bits por símbolo, para qualquer processo de codificação sem perda. Ou seja, nenhum sistema de compressão que opere sob as restrições da codificação sem perda pode, em média, representar os dados com menos de $H(X)$ bits por símbolo, sem eliminar redundância ou estrutura estatística.

Um dos resultados mais impactantes da teoria foi o *Teorema da Codificação de Fonte de Shannon*, que afirma que é possível aproximar arbitrariamente o limite de compressão determinado pela entropia por meio de esquemas de codificação suficientemente longos e adequados. Este teorema fundamenta a distinção entre compressão com e sem perdas, definindo os parâmetros de eficiência para algoritmos de codificação entrópica, como os desenvolvidos posteriormente por Huffman (HUFFMAN, 1952), Rissanen (RISSANEN, 1976), Pasco (PASCO, 1976) e Duda (DUDA, 2013).

Além da compressão, a Teoria da Informação também define limites para a capacidade de canais de comunicação ruidosos. No entanto, no contexto da compressão de dados, seu impacto reside principalmente na formulação dos princípios fundamentais de codificação ótima e na motivação para algoritmos que aproximem esse limite teórico. A entropia de uma fonte pode ser interpretada como a taxa mínima de bits necessária para representar a informação de forma inequívoca, enquanto o excesso sobre essa taxa — introduzido por algoritmos subótimos — corresponde à redundância do sistema.

Shannon também introduziu o conceito de *informação mútua*, que quantifica a dependência estatística entre variáveis aleatórias, sendo relevante na modelagem de canais de compressão com memória ou na compactação de estruturas correlacionadas, como imagens e sequências temporais. A compressão eficiente depende, portanto, não apenas das probabilidades marginais dos símbolos, mas também de estruturas mais complexas como n-gramas, correlações temporais e dependências espaciais.

Com o tempo, a teoria evoluiu e se desdobrou em diversas subáreas aplicadas, como codificação de canal, teoria algorítmica da informação, compressão universal e codificação de fontes com estrutura. Em particular, as ideias de Shannon inspiraram o desenvolvimento de algoritmos que buscam incorporar conhecimento probabilístico adaptativo, como é o caso do ANS adaptativo estudado neste trabalho, o qual explora a variação dinâmica das distribuições de probabilidade como meio para melhorar a taxa de compressão em cenários não estacionários.

Ao estabelecer os limites fundamentais e fornecer as ferramentas matemáticas

para seu estudo, a Teoria da Informação permanece como o alicerce teórico sobre o qual repousa toda a ciência da compressão de dados. Sua influência é particularmente visível na avaliação do desempenho de esquemas de codificação, na análise da eficiência de algoritmos e na determinação de compromissos entre taxa de bits, latência e complexidade computacional.

3.1.1 Tratamento da Informação e Canais de Comunicação

Após estabelecer as bases teóricas da entropia e da codificação de fontes, Shannon expandiu sua teoria para considerar o processo de transmissão da informação por meio de canais de comunicação (SHANNON, 1948). Neste contexto, o termo *canal* refere-se a um sistema físico ou lógico que conecta uma fonte emissora a um receptor, permitindo a transferência de mensagens. A modelagem matemática desses canais visa quantificar a eficiência e a confiabilidade da transmissão, levando em consideração fenômenos como ruído, interferência e perda de dados. A análise dos canais é central para o entendimento das limitações fundamentais impostas à transmissão de informação e permite derivar estratégias ótimas de codificação que maximizem a taxa de comunicação sem erro.

Formalmente, um canal de comunicação é caracterizado por uma função de transição de probabilidade $P(Y|X)$, que descreve a probabilidade de o símbolo de saída Y ser observado dado que o símbolo de entrada X foi transmitido. Quando os conjuntos de entrada e saída são finitos, o canal é dito **discreto**. Em contrapartida, canais **contínuos** envolvem variáveis aleatórias com distribuições densas, como no caso de sinais analógicos ou modelos com ruído gaussiano aditivo. Além disso, os canais podem ser **determinísticos**, quando a saída é uma função fixa da entrada, ou **estocásticos**, quando há incerteza na transmissão.

Outro aspecto importante diz respeito à reciprocidade do canal. Um canal é dito **recíproco** se suas propriedades de transmissão se mantêm simétricas entre os dois pontos terminais, o que nem sempre é o caso em sistemas reais, como em canais de rádio ou redes assimétricas de sensores. Essa característica afeta a modelagem da capacidade e a escolha de estratégias de codificação.

No estudo desses canais, o conceito de **informação mútua** emerge como uma ferramenta fundamental para quantificar a quantidade de informação que a saída de um canal revela sobre sua entrada. A informação mútua entre duas variáveis aleatórias X e Y é definida como:

$$I(X;Y) = \sum_{x \in \mathscr{X}} \sum_{y \in \mathscr{Y}} p(x,y) \log_2 \left(\frac{p(x,y)}{p(x)p(y)} \right) \tag{2}$$

Esta medida expressa o ganho de informação obtido sobre X após a observação de Y. Quando X e Y são independentes, $I(X;Y) = 0$, indicando ausência de qualquer relação informacional entre as variáveis. Já quando são totalmente dependentes, a informação mútua coincide com a entropia de X, representando máxima previsibilidade. A informação mútua também pode ser interpretada como a redução da incerteza sobre X após a recepção de Y, formalizada como $I(X;Y) = H(X) - H(X|Y)$, onde $H(X|Y)$ é a entropia condicional.

A importância prática deste conceito reside na definição da **capacidade do canal**, que corresponde à máxima taxa de transmissão confiável possível. Para canais discretos com memória nula, a capacidade C é definida por:

$$C = \max_{p(x)} I(X;Y) \tag{3}$$

Esse resultado, conhecido como o *Teorema da Capacidade do Canal*, estabelece um limite teórico para a comunicação sem erro arbitrariamente pequeno, desde que se utilizem códigos suficientemente longos e estratégias adequadas de codificação e decodificação. Para canais contínuos, como o canal gaussiano aditivo com potência limitada e ruído branco, Shannon demonstrou que a capacidade é dada por:

$$C = \frac{1}{2} \log_2 \left(1 + \frac{S}{N} \right) \quad \text{[bits por segundo por Hz]} \tag{4}$$

onde S é a potência do sinal e N a potência do ruído, definindo a razão sinal-ruído (SNR). Essa expressão, conhecida como **fórmula de Shannon-Hartley**, estabelece um limite fundamental para a taxa de transmissão confiável em sistemas analógicos sujeitos a ruído gaussiano.

A generalização desses conceitos tem implicações diretas em aplicações modernas como canais MIMO (Multiple Input Multiple Output), comunicação quântica e compressão com fontes correlacionadas. Trabalhos contemporâneos, como os de Cover e Thomas (COVER; THOMAS, 2006), continuam a expandir o entendimento teórico dos canais, integrando abordagens probabilísticas com modelos práticos de rede. Além disso, em contextos de compressão com aprendizado de máquina, medidas como informação mútua vêm sendo empregadas para regularização de modelos, estimativa de dependências latentes e otimização de codificadores neurais (TSCHANNEN et al., 2020).

Dessa forma, o estudo dos canais de comunicação e do fluxo de informação entre variáveis aleatórias não apenas fundamenta os limites da compressão e transmissão, mas também influencia diretamente o design de algoritmos modernos, como aqueles empregados no ANS adaptativo estudado neste trabalho. A caracterização precisa da estrutura probabilística dos dados e das transformações envolvidas é essencial para garantir que a taxa de bits obtida se aproxime dos limites estabelecidos pela teoria.

3.1.2 Entropia e Variáveis Aleatórias

A entropia é o conceito fundamental da Teoria da Informação e serve como medida da incerteza associada a uma variável aleatória. Introduzida por Claude Shannon em 1948 (SHANNON, 1948), a entropia formaliza, em termos probabilísticos, a quantidade média de informação contida em um conjunto de possíveis eventos. No contexto da compressão de dados, a entropia representa o limite teórico mínimo para a codificação de uma fonte, isto é, a menor taxa média de bits por símbolo necessária para representar a saída de uma fonte sem perdas.

Para compreender plenamente o conceito de entropia, é necessário primeiro considerar o tipo de variável aleatória em estudo. Variáveis aleatórias podem ser classificadas como **discretas** ou **contínuas**, e a definição de entropia difere em cada caso, embora exista uma notação unificada que permita expressar ambos os conceitos de maneira geral.

Variável Aleatória Discreta: Seja X uma variável aleatória discreta que assume valores em um alfabeto finito $\mathscr{X} = \{x_1, x_2, ..., x_n\}$, com função de probabilidade de massa $p(x) = \mathbb{P}(X = x)$. A entropia de X, denotada por $H(X)$, é definida como:

$$H(X) = -\sum_{x \in \mathscr{X}} p(x) \log_2 p(x) \tag{5}$$

Esta fórmula quantifica a média ponderada da imprevisibilidade associada à ocorrência dos eventos. Quanto menor a probabilidade de um símbolo, maior a quantidade de informação associada à sua ocorrência. O logaritmo é usualmente na base 2, o que expressa a entropia em bits.

Variável Aleatória Contínua: No caso contínuo, seja X uma variável aleatória com função densidade de probabilidade $f(x)$, definida sobre um subconjunto $\mathbb{R} \subseteq \mathbb{R}^d$. A entropia diferencial de X, denotada por $h(X)$, é dada por:

$$h(X) = -\int_{\mathbb{R}} f(x) \log_2 f(x)\, dx \tag{6}$$

Embora similar em forma, a entropia diferencial não representa uma quantidade absoluta de bits. Ela pode ser negativa e está sujeita a transformações contínuas que afetam a densidade, mas não a estrutura informacional da variável. Por esse motivo, $h(X)$ deve ser interpretada com cautela e é mais útil para comparação entre distribuições contínuas do que para medidas absolutas de compressibilidade.

Notação Unificada: Diversos autores modernos, como Cover e Thomas (COVER; THOMAS, 2006), adotam uma notação unificada da entropia baseada na expectativa da auto-informação de X, seja discreta ou contínua:

$$\mathbb{H}(X) = \mathbb{E}[-\log_2 p(X)] \tag{7}$$

onde $p(X)$ pode ser interpretado como a função de massa de probabilidade (no caso discreto) ou densidade de probabilidade (no caso contínuo). Essa abordagem permite tratar formalmente distribuições mistas e facilita a análise de transformações aplicadas a X.

Propriedades: A entropia satisfaz diversas propriedades que a tornam adequada como medida de informação: - $H(X) \geq 0$, com igualdade se e somente se X é determinística; - $H(X)$ é máxima quando X é uniformemente distribuída sobre $\mathscr{X}$; - Para duas variáveis X e Y, a entropia conjunta $H(X,Y)$ satisfaz a desigualdade subaditiva $H(X,Y) \leq H(X) + H(Y)$, com igualdade se e somente se X e Y são independentes.

Além disso, a entropia pode ser condicionada. A **entropia condicional** $H(X|Y)$ mede a incerteza remanescente sobre X dado conhecimento de Y, e é definida por:

$$H(X|Y) = -\sum_{x,y} p(x,y) \log_2 p(x|y) \tag{8}$$

Do ponto de vista da compressão de dados, a entropia define um limite fundamental: qualquer algoritmo de codificação que produza, em média, uma quantidade de bits menor que $H(X)$ inevitavelmente perderá informação. Por isso, algoritmos como Huffman, codificação aritmética e ANS são projetados para aproximar esse limite inferior, respeitando a estrutura probabilística da fonte.

Trabalhos recentes, como os de Ver Steeg e Galstyan (STEEG; GALSTYAN, 2014), exploram generalizações da entropia no contexto de aprendizado de máquina e representação informacional, empregando extensões como entropia condicional

multivariada, entropia cruzada e divergência de Kullback-Leibler. Tais ferramentas são fundamentais para a integração de modelos estatísticos com sistemas de compressão baseados em aprendizado profundo, como os analisados neste trabalho.

3.1.3 Informação e Entropia

A entropia, conforme definida por Shannon (SHANNON, 1948), quantifica a incerteza associada a uma variável aleatória e, por consequência, estabelece um limite teórico para a compressão de dados. No entanto, a informação, em seu sentido mais técnico, está intrinsecamente ligada à redução dessa incerteza. A diferença entre o que se sabe antes e depois da observação de um evento representa a quantidade de informação adquirida. Nesse contexto, surge o conceito de **informação mútua**, que mede precisamente quanto a observação de uma variável reduz a incerteza sobre outra.

Sejam X e Y duas variáveis aleatórias com distribuição conjunta $p(x,y)$; a informação mútua $I(X;Y)$ é definida como:

$$I(X;Y) = \sum_{x \in \mathscr{X}} \sum_{y \in \mathscr{Y}} p(x,y) \log_2 \left(\frac{p(x,y)}{p(x)p(y)} \right) \tag{9}$$

Essa quantidade representa o ganho de informação sobre X proporcionado pela observação de Y, ou vice-versa, dado que $I(X;Y) = I(Y;X)$. A informação mútua é sempre não-negativa e é nula se, e somente se, X e Y são independentes. Outra forma equivalente de expressá-la, em termos de entropias, é:

$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) = H(X) + H(Y) - H(X,Y) \tag{10}$$

Essa representação mostra que a informação mútua quantifica a interseção das incertezas das variáveis, e por isso pode ser interpretada visualmente por meio de **diagramas de Venn**, onde cada conjunto representa a entropia de uma variável e a sobreposição representa a informação mútua. Essa abordagem, comum em textos como o de Cover e Thomas (COVER; THOMAS, 2006), auxilia na construção de intuições visuais sobre os relacionamentos informacionais entre variáveis.

Pode-se observar através do Diagrama de Venn na Figura 4 a representação da informação mútua entre as entropias.

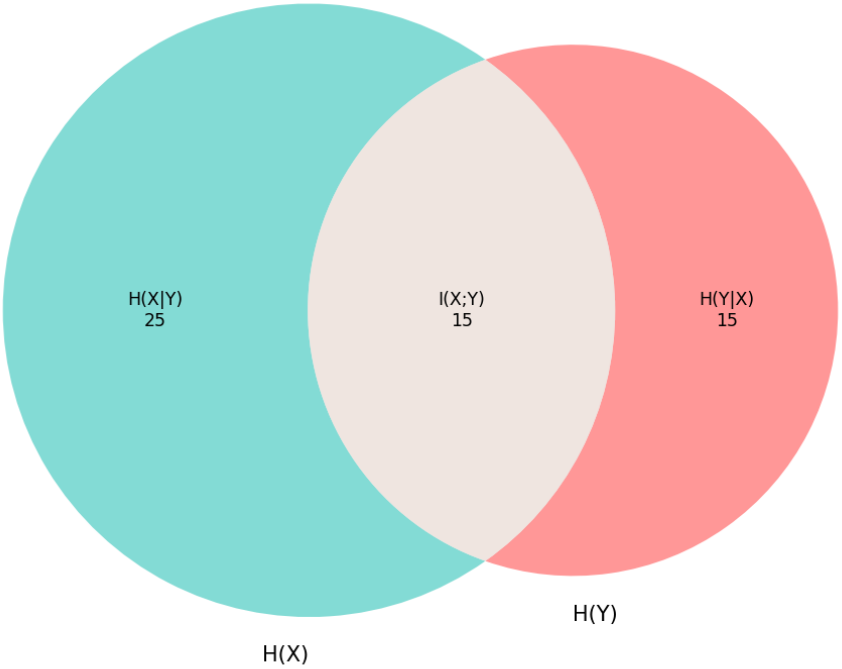

Figura 4 – Representação das entropias $H(X)$, $H(Y)$ e da informação mútua $I(X;Y)$ como interseção entre conjuntos

Fonte: Autor (2025).

Informação Transmitida em Canais Discretos: No contexto da comunicação por canais, a informação mútua entre a variável de entrada X e a variável de saída Y mede a eficiência do canal em transmitir informação. A capacidade C de um canal discreto sem memória é definida como a máxima informação mútua entre entrada e saída, considerando todas as distribuições possíveis $p(x)$:

$$C = \max_{p(x)} I(X;Y) \tag{11}$$

Esse é o núcleo do *Teorema da Capacidade do Canal*, que afirma que é possível transmitir informação através do canal com taxa arbitrariamente próxima de C, com probabilidade de erro tendendo a zero, desde que se utilizem blocos de codificação suficientemente longos.

A informação mútua, portanto, assume papel central tanto em compressão de dados (como medida de dependência entre símbolos consecutivos ou entre dados e representações latentes) quanto em comunicação (como métrica da quantidade de informação efetivamente transferida pelo canal).

Entropias Diferenciais e Informação em Variáveis Contínuas: Quando lidamos com variáveis contínuas, a entropia passa a ser substituída por sua versão

diferencial $h(X)$, conforme discutido anteriormente. A informação mútua entre variáveis contínuas é definida por analogia:

$$I(X;Y) = \int_{\mathscr{X}} \int_{\mathscr{Y}} f(x,y) \log_2 \left(\frac{f(x,y)}{f(x)f(y)} \right) dx\,dy \tag{12}$$

Apesar de a entropia diferencial $h(X)$ poder assumir valores negativos, a informação mútua entre variáveis contínuas é sempre não negativa, mantendo sua interpretação como quantidade de dependência estatística entre variáveis.

Esse conceito é amplamente utilizado em sistemas modernos de compressão baseados em aprendizado profundo, como codificadores variacionais (VAE) e compressão neural, onde a informação mútua entre dados e representações latentes é usada para controlar a taxa de bits e preservar a estrutura informacional (ALEMI et al., 2017; TSCHANNEN et al., 2020).

Informação e Incerteza: A relação entre informação e incerteza é dual. A entropia mede a incerteza associada a uma variável aleatória, enquanto a informação é interpretada como a redução dessa incerteza. Assim, o conhecimento de uma variável correlacionada, como a saída de um canal, diminui a entropia condicional da variável de interesse, e a diferença entre a incerteza inicial e a incerteza residual constitui a informação adquirida.

No contexto deste trabalho, a compressão adaptativa baseada em ANS e redes neurais busca, essencialmente, maximizar a quantidade de informação relevante que pode ser codificada com o menor número possível de bits. Para isso, o modelo de probabilidade usado durante a codificação deve se aproximar da verdadeira distribuição dos dados, minimizando a entropia condicional e, portanto, otimizando a taxa de compressão conforme os limites estabelecidos pela informação mútua entre a fonte e o codificador.

3.1.4 Modelos Probabilísticos: Fundamentos e Aplicações em Compressão de Dados

A compressão eficiente de dados depende diretamente da capacidade de modelar, com precisão, a distribuição de probabilidade da fonte geradora da informação. A ideia de que a compressibilidade de uma mensagem está associada à sua previsibilidade foi introduzida já por Ralph Hartley em 1928 (HARTLEY, 1928), que propôs uma medida logarítmica para a quantidade de informação em termos do número de escolhas possíveis. No entanto, foi com Claude Shannon, em 1948 (SHANNON, 1948), que a

teoria probabilística da informação foi formalmente estabelecida. Shannon demonstrou que o limite teórico de compressão sem perdas é determinado pela entropia da distribuição de probabilidade dos símbolos emitidos por uma fonte.

Um **modelo probabilístico** é, essencialmente, uma representação matemática que descreve a probabilidade de ocorrência de diferentes símbolos ou eventos. Dado um alfabeto $\mathscr{X} = \{x_1, x_2, \ldots, x_n\}$, um modelo define uma distribuição $P(X = x_i) = p_i$, tal que $\sum_i p_i = 1$. Quanto mais precisa for essa modelagem em relação à real distribuição da fonte, mais próxima estará a taxa de compressão obtida do limite teórico $H(X)$.

Considere o seguinte exemplo simples: uma fonte binária que gera os símbolos '0' e '1' com probabilidades $P(0) = 0.9$ e $P(1) = 0.1$. Um codificador que utiliza um modelo uniforme, isto é, assumindo $P(0) = P(1) = 0.5$, não conseguirá aproveitar o caráter redundante da fonte, produzindo códigos de comprimento médio maior. Já um codificador informado sobre a verdadeira distribuição poderá atribuir códigos mais curtos aos símbolos mais prováveis, como ocorre na codificação de Huffman, reduzindo a taxa de bits média.

Modelos Estáticos vs. Adaptativos: Tradicionalmente, os modelos probabilísticos podem ser classificados como:

- *Estáticos*, quando a distribuição de probabilidade é fixada previamente com base em dados históricos ou suposições.
- *Adaptativos*, quando o modelo é ajustado dinamicamente conforme novos símbolos são observados, permitindo acompanhar mudanças na distribuição da fonte ao longo do tempo.

Modelos Probabilísticos e Entropia Cruzada: Um aspecto crucial do uso de modelos probabilísticos em compressão é a entropia cruzada, definida como:

$$H(p,q) = -\sum_{x \in \mathscr{X}} p(x) \log_2 q(x) \tag{13}$$

onde $p(x)$ é a distribuição verdadeira da fonte e $q(x)$ é o modelo utilizado para a codificação. A entropia cruzada representa a taxa média de bits por símbolo quando se utiliza q para codificar uma fonte que, na verdade, segue a distribuição p. A diferença $H(p,q) - H(p)$ é justamente a **divergência de Kullback-Leibler** $D_{KL}(p \parallel q)$, que quantifica a ineficiência causada por usar um modelo incorreto.

Modelagem Probabilística Moderna: Na era do aprendizado de máquina, modelos probabilísticos tornaram-se ainda mais sofisticados. Abordagens baseadas em

aprendizado profundo — como modelos autoregressivos, modelos variacionais e redes neurais normalizadoras — são capazes de capturar dependências complexas e contextos de longo alcance nos dados. Em compressão neural, como mostrado por Ballé et al. (BALLÉ et al., 2018) e Tschannen et al. (TSCHANNEN et al., 2020), redes neurais são treinadas para aprender representações latentes z que seguem distribuições bem modeladas, permitindo codificação eficiente com estimativas explícitas de probabilidade.

Por exemplo, considere um codificador baseado em VAE (Variational Autoencoder). Ele tenta aprender a distribuição $p(x|z)$, onde z é uma variável latente comprimida, e também a distribuição aproximada $q(z|x)$, que permite inferir z a partir dos dados. O uso de técnicas como estimadores de informação mútua e regularização por entropia cruzada permite não apenas alcançar compressões eficientes, mas também preservar informações semânticas.

Quanto melhor o modelo conseguir antecipar a ocorrência dos símbolos, menor será a quantidade média de bits necessária para representá-los. A integração de modelos probabilísticos precisos com esquemas de codificação eficientes é essencial para se atingir os limites da compressão sem perdas estabelecidos pela teoria da informação.

3.2 Trabalhos Correlatos

A construção deste trabalho foi fundamentada em uma análise crítica de pesquisas anteriores que abordam temas centrais à compressão de dados eficiente, com foco em arquiteturas digitais de alta vazão e baixo consumo energético, codificação entrópica para formatos de vídeo como AV1 e HEVC, e técnicas de compressão aprendida aplicadas a cenários de aprendizado de máquina. Os trabalhos correlatos selecionados contribuíram de maneira significativa em diferentes fases do desenvolvimento desta pesquisa, desde a compreensão dos desafios computacionais envolvidos na implementação de codificadores entrópicos até o direcionamento de soluções viáveis para integração com redes neurais e aplicações em tempo real.

Parte da literatura consultada descreve arquiteturas otimizadas para a implementação de codificadores e decodificadores aritméticos em hardware, destacando estratégias para reduzir latência e consumo de energia, onde os recursos de processamento, memória e energia são limitados e exigem soluções altamente eficientes, aspectos essenciais para aplicações embarcadas e cenários de *edge computing*. Outros estudos se concentram em técnicas de compressão aprendida, oferecendo subsídios

teóricos e práticos para a aplicação de modelos probabilísticos adaptativos em codificadores modernos. Tais contribuições também ajudaram a embasar as perguntas centrais deste trabalho, como a viabilidade da atualização das distribuições de probabilidade por lotes, o impacto do tamanho desses lotes no desempenho e o possível ganho de acurácia na compressão adaptativa.

Com base nessas referências, foi possível formar a proposta de um sistema adaptativo baseado no ANS, alinhado às exigências de desempenho e flexibilidade de cenários contemporâneos. A seguir, apresentam-se os principais trabalhos correlatos que subsidiaram esta pesquisa, com destaque para suas contribuições e limitações frente ao objetivo deste estudo.

A tese de Fábio Ramos intitulada *Efficient High-Throughput and Power-Saving Hardware Architectural Design for the HEVC Entropy Encoder* (RAMOS, 2019) contribui de maneira relevante para este trabalho ao apresentar soluções arquiteturais voltadas para a codificação de entropia em cenários de alto desempenho e baixo consumo energético. Embora o foco esteja no algoritmo CABAC, adotado pelo padrão HEVC, as técnicas desenvolvidas para otimização do bloco BAE (Binary Arithmetic Encoder) como o processamento paralelo de múltiplos *bins* e estratégias de redução de potência em nível de circuito fornecem insights valiosos para a implementação de sistemas de compressão eficientes em hardware. A abordagem configurável proposta pelo autor pode inspirar o desenvolvimento de versões adaptativas do ANS que se ajustem dinamicamente ao contexto de execução, aproximando-se dos requisitos de compressão em tempo real com suporte a atualizações de probabilidade.

A dissertação de Tulio Bitencourt que propõe o codificador AE-AV1 para o padrão de vídeo AV1 apresenta contribuições relevantes no contexto da compressão de dados de alta performance, especialmente em arquiteturas VLSI para codificação aritmética (BITENCOURT, 2023). Embora o foco do trabalho esteja na implementação de um codificador específico para AV1, as estratégias adotadas para superar gargalos de paralelização no codificador aritmético incluindo o uso de operações CDF e Booleanas fornecem referências úteis para o desenvolvimento de sistemas de codificação entrópica eficientes e escaláveis. Para este trabalho, que visa explorar uma abordagem adaptativa com ANS em cenários de baixa latência, os resultados obtidos com as arquiteturas AE-AV1, incluindo suas variantes de baixo consumo e multi-booleanas, mostram soluções de hardware aplicáveis e alinhadas aos desafios de eficiência computacional e suporte a fluxos de dados em tempo real.

O trabalho de Jacobellis e Yadwadkar propõe o *WaLLoC* (*Wavelet Learned Lossy Compression*), uma arquitetura de codec neural que combina transformadas wavelet invertíveis com autoencoders assimétricos e gargalos de entropia para compressão eficiente de dados voltada ao aprendizado em domínio comprimido (JACOBELLIS; YADWADKAR, 2023). Apesar de não utilizar diretamente o ANS ou suas variantes, o estudo oferece subsídios importantes para esta pesquisa, ao demonstrar como modelos de compressão podem ser integrados a pipelines de inferência. A ênfase do WaLLoC na eficiência computacional, através de codificadores predominantemente lineares, e sua compatibilidade com aplicações em dispositivos móveis e sensores remotos, convergem com os objetivos do presente trabalho, que busca desenvolver um sistema adaptativo baseado em ANS capaz de operar em contextos de baixa latência e recursos limitados.

O trabalho de conclusão de curso de Jiovanna Gomes propõe quatro arquiteturas digitais voltadas à etapa de decodificação aritmética do codec AV1, com foco em alcançar alto desempenho e baixo consumo de energia (GOMES, 2022). As soluções apresentadas abordam desde arquiteturas monociclo, capazes de decodificar qualquer um dos 16 símbolos possíveis em um único ciclo de clock, até versões de menor consumo energético, utilizando técnicas como *Operand Isolation* e adaptação multi-ciclo baseada em estatísticas de frequência dos símbolos. Os resultados mostram que todas as propostas atingem taxa de transferência suficiente para suportar decodificação de vídeos 8K a 60 fps em tempo real. Para este trabalho, tais abordagens são relevantes tanto do ponto de vista da viabilidade de implementação em hardware quanto da inspiração para estratégias adaptativas baseadas na dinâmica de ocorrência de símbolos, o que reforça a importância de modelos probabilísticos atualizáveis em tempo de execução no contexto do ANS.

3.3 Histórico e Evolução da Codificação Entrópica

Os algoritmos de compressão de dados desempenham um papel central na sociedade da informação, possibilitando desde o armazenamento eficiente de grandes volumes de dados até a transmissão em tempo real de conteúdo multimídia por redes de comunicação modernas.

Desde então, uma série de algoritmos foram propostos para se aproximar ou atingir esse limite. Entre os primeiros, destaca-se o método de codificação de Huffman, proposto em 1952 (HUFFMAN, 1952), que introduziu um esquema eficiente de codificação de prefixo com base nas probabilidades dos símbolos. Poucas décadas depois,

na década de 1970, surgiram os algoritmos de codificação aritmética (RISSANEN, 1976; PASCO, 1976), que permitiram representar frações de bits por símbolo, aproximando-se ainda mais da entropia da fonte. Esses algoritmos marcaram uma revolução ao permitirem maior flexibilidade e precisão na representação da informação, sendo amplamente utilizados em padrões de compressão como JPEG (ISO/IEC; ITU-T, 1992), H.264 (ISO/IEC; ITU-T, 2006) e H.265 (ITU-T, 2016).

Com o avanço da computação e o crescimento exponencial de dados na era digital, a busca por algoritmos mais rápidos, compactos e adaptáveis levou à criação de novas abordagens. Entre os exemplos mais notáveis estão os algoritmos Lempel-Ziv (LZ77 e LZ78) (ZIV; LEMPEL, 1977; ZIV; LEMPEL, 1978), que introduziram a ideia de compressão baseada em dicionários e se tornaram a base para formatos amplamente utilizados como ZIP, PNG e GIF. Em tempos mais recentes, algoritmos como o Brotli (utilizado por navegadores modernos) e o Zstandard (desenvolvido pelo Facebook) aliam técnicas clássicas de compressão a otimizações modernas e modelos estatísticos refinados.

A partir de 2009, um novo paradigma emergiu com a proposta do *Asymmetric Numeral Systems* (ANS) por Jarek Duda (DUDA, 2013). O ANS combina a eficiência da codificação aritmética com a simplicidade computacional do Huffman, possibilitando compressão próxima ao limite da entropia com alta performance em hardware moderno. Seu impacto prático foi imediato, sendo adotado em sistemas de compressão amplamente utilizados como Apple LZFSE e Facebook Zstandard.

Além das aplicações em compressão de arquivos e mídias, os algoritmos entrópicos têm encontrado novos papéis em domínios como inteligência artificial, computação embarcada e comunicação em tempo real, onde a necessidade de modelos probabilísticos eficientes e baixa latência é cada vez mais crítica. No contexto de aprendizado de máquina, por exemplo, técnicas de compressão entrópica são empregadas para reduzir o tamanho de modelos neurais sem comprometer significativamente sua acurácia, viabilizando sua execução em dispositivos de borda.

Diante desse panorama, os algoritmos de compressão baseados em entropia continuam a evoluir, impulsionados tanto por avanços teóricos quanto por demandas tecnológicas emergentes. Nas seções seguintes, exploraremos em detalhes três dos principais paradigmas da codificação entrópica: a codificação de Huffman, a codificação aritmética e o Asymmetric Numeral Systems (ANS), destacando suas estruturas, propriedades matemáticas, aplicações práticas e limitações.

3.3.1 Codificação de Huffman

A codificação de Huffman, introduzida por David A. Huffman em seu artigo seminal de 1952 (HUFFMAN, 1952), representa um marco na história da compressão de dados. Seu trabalho, desenvolvido como parte de um curso ministrado por Robert Fano no MIT, propôs um método sistemático para construir códigos binários de comprimento variável com base nas probabilidades de ocorrência dos símbolos. O algoritmo resultante gera códigos de prefixo *ótimos*, no sentido de minimizar o comprimento médio de codificação para uma fonte de informação com distribuição de probabilidade conhecida.

A ideia fundamental por trás do algoritmo de Huffman é simples, mas poderosa: atribuir códigos curtos aos símbolos mais frequentes e códigos mais longos aos símbolos menos frequentes, garantindo que nenhum código seja prefixo de outro (propriedade de prefixo). Isso evita ambiguidades na decodificação e assegura que o processo seja instantaneamente decodificável.

O algoritmo constrói uma árvore binária a partir das probabilidades dos símbolos, combinando iterativamente os dois símbolos (ou subárvores) de menor frequência até que reste uma única árvore. A travessia da árvore da raiz até cada folha produz o código binário correspondente a cada símbolo.

O algoritmo de Huffman pode ser formalizado matematicamente através dos seguintes passos:

1. Dada uma fonte discreta com alfabeto $\mathscr{X} = \{x_1, ..., x_n\}$ e probabilidades associadas $P = \{p_1, ..., p_n\}$

2. Construir uma floresta de árvores binárias, onde cada símbolo x_i é uma folha com peso $w_i = p_i$

3. Enquanto houver mais de uma árvore na floresta:

   (a) Selecionar as duas árvores T_1 e T_2 com menores pesos w_1 e w_2

   (b) Combiná-las em uma nova árvore T' com:

$$T'.\text{left} = T_1, \quad T'.\text{right} = T_2, \quad T'.\text{weight} = w_1 + w_2 \tag{14}$$

4. Atribuir códigos através do percurso da árvore resultante:

$$c(x_i) = \begin{cases} \varepsilon & \text{se } x_i \text{ é a raiz} \\ c(\text{parent}) \circ 0 & \text{se } x_i \text{ é filho esquerdo} \\ c(\text{parent}) \circ 1 & \text{se } x_i \text{ é filho direito} \end{cases} \tag{15}$$

onde $\circ$ denota concatenação de bits e ε a string vazia

A eficiência do código resultante pode ser analisada através do comprimento esperado:

$$L(H) = \sum_{i=1}^{n} p_i l_i \tag{16}$$

onde l_i é o comprimento do código para o símbolo x_i. O Teorema de Huffman garante que este código é ótimo entre todos os códigos de prefixo, satisfazendo:

$$H(X) \leq L(H) < H(X) + 1 \tag{17}$$

onde $H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i$ é a entropia da fonte.

Considere a tabela 2 com a seguinte fonte com cinco símbolos e suas respectivas probabilidades:

Tabela 2 – Distribuição de Probabilidades da Fonte

Símbolo	**Probabilidade**
A	0,35
B	0,20
C	0,20
D	0,15
E	0,10

Fonte: Autor (2025).

O algoritmo de Huffman segue os seguintes passos:

1. Inicializa uma floresta onde cada símbolo é uma árvore separada com peso igual à sua probabilidade.

2. Combina os dois nós de menor peso em uma nova árvore cujo peso é a soma dos dois.

3. Repete o processo até que reste uma única árvore.

4. Atribui 0 para a ramificação esquerda e 1 para a direita (ou vice-versa), formando os códigos dos símbolos.

A árvore de Huffman gerada pode ser observada na figura 5, abaixo:

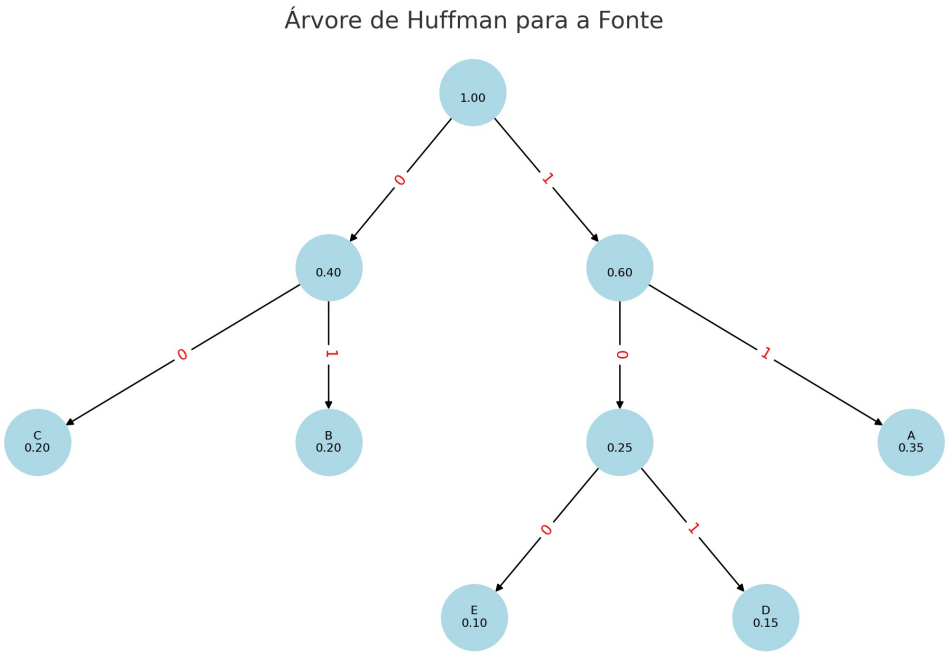

Figura 5 – Arvore de Huffman Gerada para a Fonte
Fonte: Autor (2025).

Após a construção da árvore, os códigos gerados para os símbolos podem ser representados na tabela 3, por exemplo:

Tabela 3 – Códigos de Huffman Gerados

Símbolo	**Código de Huffman**
A	0
B	10
C	11
D	110
E	111

Fonte: Autor (2025).

Neste exemplo, o comprimento médio do código, $L = \sum p(x_i) \cdot l_i$, é dado por:

$$L = 0{,}35 \cdot 1 + 0{,}20 \cdot 2 + 0{,}20 \cdot 2 + 0{,}15 \cdot 3 + 0{,}10 \cdot 3 = 1{,}95 \text{ bits/símbolo}$$

Comparado à entropia da fonte, calculada como:

$$H(X) = -\sum p(x_i) \log_2 p(x_i) \approx 1{,}92 \text{ bits/símbolo}$$

o código de Huffman é próximo do limite ótimo teórico estabelecido por Shannon (SHANNON, 1948), demonstrando sua eficiência.

As principais vantagens da codificação de Huffman incluem:

- **Ótimo entre códigos de prefixo**: Nenhum outro código de prefixo pode ter comprimento médio menor para a mesma distribuição.
- **Simplicidade computacional**: Pode ser implementado de forma eficiente com filas de prioridade (heap) e árvores binárias.

No entanto, existem limitações:

- **Granularidade de 1 bit**: Não permite representar frações de bits por símbolo, o que o torna menos eficiente quando a entropia da fonte é menor que 1 bit/símbolo.
- **Eficiência limitada para fontes com baixa variação de probabilidades**: Diferenças sutis nas frequências podem não refletir significativamente no comprimento dos códigos.
- **Não se adapta dinamicamente**: Versões clássicas requerem conhecimento prévio da distribuição dos dados.

Embora suas limitações, a codificação de Huffman é amplamente utilizada em aplicações reais devido à sua simplicidade e eficácia prática. Alguns exemplos incluem:

- **Compressão de texto e dados gerais**: Algoritmo DEFLATE (ZIP, gzip), onde Huffman é combinado com LZ77.
- **Compressão de imagens**: Padrões como JPEG utilizam Huffman para codificar os coeficientes quantizados da transformada discreta do cosseno.
- **Codificação de áudio e vídeo**: Presente em MPEG-1, MP3, e outros formatos onde a compactação de entropia é necessária após a análise de frequência.

3.3.2 Codificação Aritmética

A codificação aritmética surgiu na década de 1970 como uma alternativa aos métodos tradicionais de compressão baseados em códigos de prefixo fixo, como a codificação de Huffman. O conceito foi inicialmente desenvolvido por Jorma Rissanen (RISSANEN, 1976) e, independentemente, por Richard Pasco em sua tese de doutorado (PASCO, 1976). Diferentemente da codificação de Huffman, que atribui um número inteiro de bits a cada símbolo com base em sua frequência, a codificação aritmética permite representar sequências de símbolos como intervalos reais, o que possibilita atingir taxas de compressão mais próximas do limite teórico definido pela entropia de Shannon (SHANNON, 1948).

A ideia central da codificação aritmética é representar uma mensagem como um número real único no intervalo $[0,1)$, refinando sucessivamente esse intervalo a cada símbolo codificado. Cada símbolo é associado a uma subfaixa do intervalo atual, proporcional à sua probabilidade. Ao final da codificação, qualquer número dentro do intervalo final pode ser usado para representar toda a sequência original.

Dada uma fonte de informação com alfabeto $\mathscr{X} = \{x_1, x_2, ..., x_n\}$ e probabilidades associadas $P = \{p_1, p_2, ..., p_n\}$, a codificação aritmética constrói, iterativamente, um intervalo $[L_n, H_n)$ que representa a sequência de entrada $s_1, s_2, ..., s_n$, segundo a fórmula:

$$[L_{i+1}, H_{i+1}) = \big[L_i + (H_i - L_i) \cdot \text{CDF}(s_i^-),\ L_i + (H_i - L_i) \cdot \text{CDF}(s_i)\big) \tag{18}$$

onde $\text{CDF}(s_i^-)$ é a soma acumulada das probabilidades dos símbolos anteriores a s_i, e $\text{CDF}(s_i)$ é a soma das probabilidades até s_i. O intervalo começa em $[0,1)$, e é progressivamente refinado à medida que os símbolos são processados.

Considere a seguinte distribuição de probabilidade para três símbolos:

Tabela 4 – Distribuição de Probabilidade

Símbolo	Probabilidade	CDF
A	0,5	[0,0, 0,5)
B	0,3	[0,5, 0,8)
C	0,2	[0,8, 1,0)

Fonte: Autor (2025).

Vamos codificar a sequência “AB”:

- Começamos com $[L_0, H_0) = [0, 1)$
- Primeiro símbolo: A

$$L_1 = 0 + (1-0) \cdot 0{,}0 = 0, \quad H_1 = 0 + (1-0) \cdot 0{,}5 = 0{,}5 \Rightarrow [0, 0{,}5)$$

- Segundo símbolo: B

$$L_2 = 0 + (0{,}5-0) \cdot 0{,}5 = 0{,}25, \quad H_2 = 0 + (0{,}5-0) \cdot 0{,}8 = 0{,}4 \Rightarrow [0{,}25, 0{,}4)$$

O número final pode ser qualquer valor entre 0,25 e 0,4. Por exemplo, poderíamos usar o número 0,3 para representar "AB". O decodificador, conhecendo a distribuição, pode reconstruir os símbolos a partir desse número, refinando os intervalos da mesma forma que o codificador.

As principais vantagens da codificação aritmética são:

- **Eficiência**: Permite representar a informação com precisão fracionária, atingindo compressão muito próxima à entropia teórica.
- **Generalidade**: Funciona bem mesmo quando as probabilidades dos símbolos não são potências de dois, onde Huffman se torna ineficiente.
- **Extensibilidade**: Pode ser estendida facilmente para codificadores adaptativos ou contextuais.

Entretanto, também apresenta limitações práticas:

- **Precisão numérica**: Exige manipulação de números com alta precisão, o que pode implicar em maior custo computacional.
- **Sensibilidade a erros**: Um pequeno erro em um dígito pode comprometer toda a mensagem.
- **Complexidade de hardware**: Implementações eficientes são mais difíceis do que as de Huffman, especialmente em arquiteturas embarcadas.

Apesar de suas limitações práticas iniciais, melhorias nas técnicas de normalização e uso de aritmética inteira permitiram o uso disseminado da codificação aritmética em diversas aplicações:

- **Compressão de imagem**: o padrão JPEG ISO/IEC 10918-1:1992 (ITU-TT.81) permite codificação aritmética, além da tradicional Huffman (ISO/IEC; ITU-T, 1992).
- **Compressão de vídeo**: O padrão H.264/AVC utiliza o método CABAC (*Context-Adaptive Binary Arithmetic Coding*), que é uma codificação binária aritmética adaptativa (ISO/IEC; ITU-T, 2006), oferecendo maior eficiência que Huffman.
- **Compressão de dados genéricos**: Algoritmos como PAQ, CMIX e outras variantes estatísticas avançadas utilizam codificação aritmética como núcleo da compressão.

Com o avanço da computação paralela e a necessidade crescente de compressão eficiente em domínios como vídeo de alta definição, transmissão de dados móveis e compressão de modelos neurais, a codificação aritmética permanece como uma ferramenta fundamental. Nas últimas décadas, surgiram algoritmos que combinam os princípios da codificação aritmética com abordagens de desempenho mais alto, como o *Asymmetric Numeral Systems* (ANS), proposto por Duda (DUDA, 2013). O ANS busca eliminar as operações de ponto flutuante, utilizando apenas aritmética inteira para alcançar desempenho semelhante, com compressão quase tão eficiente quanto a aritmética tradicional. O ANS será o enfoque central no capítulo seguinte, onde será abordado com mais profundidade.

4 ASYMMETRIC NUMERAL SYSTEMS

O *Asymmetric Numeral Systems* (ANS) é uma técnica de codificação entrópica introduzida por Jarek Duda em 2009 (DUDA, 2013), com o objetivo de combinar a eficiência compressiva da codificação aritmética com a velocidade e simplicidade computacional da codificação de Huffman. A proposta de Duda representa um avanço conceitual ao tratar a codificação como uma operação de mudança de base em um sistema numérico assimétrico, onde os símbolos são mapeados a inteiros com base em suas probabilidades associadas. Diferente das abordagens tradicionais que utilizam listas ou árvores de prefixos, o ANS mantém o estado de codificação em um único número inteiro que evolui conforme os símbolos são codificados ou decodificados.

A motivação original de Duda estava relacionada à dificuldade de aplicar codificação aritmética de forma eficiente em sistemas paralelos e embarcados, especialmente em contextos de baixa latência e uso restrito de recursos. A codificação de Huffman, embora rápida, é limitada a comprimentos de código inteiros, o que pode resultar em perda de eficiência em fontes com alta entropia fracionária. A codificação aritmética, por outro lado, é mais eficiente, mas possui maior custo computacional e complexidade de implementação, especialmente em arquiteturas modernas com restrições de precisão numérica e acesso à memória. O ANS, portanto, surgiu como uma solução intermediária que preserva a compressão próxima ao ótimo da codificação aritmética com custo computacional próximo ao da codificação de Huffman.

4.1 Fundamentos

O ANS codifica uma sequência de símbolos representando a fonte de informação como um único número inteiro x. A codificação de um símbolo s é realizada por meio de uma função de transição que mapeia o estado anterior x em um novo estado x':

$$x' = C(x,s) = \left\lfloor \frac{x}{f_s} \right\rfloor \cdot m + (x \mod f_s) + b_s \tag{19}$$

onde:

- f_s é a frequência absoluta do símbolo s na distribuição;
- m é o total de frequências (soma de todas as f_s);

- b_s é o deslocamento de base associado ao símbolo s na tabela de codificação.

O processo de decodificação é simétrico, baseado em operações inteiras de divisão e módulo, permitindo reconstruir o símbolo a partir do estado atual. Como o sistema é determinístico e reversível, cada estado mapeia de forma única para um símbolo e um estado anterior, permitindo compressão e descompressão eficientes.

Uma das vantagens centrais do ANS está no fato de operar exclusivamente com números inteiros, eliminando o uso de aritmética de ponto flutuante, o que facilita a sua implementação em dispositivos embarcados, GPUs (processadores especializados em executar muitas operações em paralelo, comuns em tarefas gráficas e de alto desempenho) e arquiteturas SIMD (conjuntos de instruções que permitem realizar a mesma operação simultaneamente sobre múltiplos dados). Além disso, o algoritmo permite fácil paralelização: como a codificação se dá em uma sequência reversa (último símbolo primeiro), múltiplos blocos de dados podem ser codificados ou decodificados em paralelo.

4.2 Modelagem de Probabilidades no ANS

A eficiência do ANS depende diretamente da qualidade da modelagem das probabilidades associadas aos símbolos da fonte. A compressão ótima só é atingida quando as probabilidades refletem com precisão a distribuição real dos dados. No contexto do ANS, essas probabilidades são expressas como **frequências relativas discretizadas**, que determinam o tamanho dos intervalos numéricos associados a cada símbolo durante a codificação.

Considere uma fonte de dados com alfabeto $\mathscr{A} = \{s_1, s_2, ..., s_n\}$ e distribuição de probabilidades associada $P = \{p_1, p_2, ..., p_n\}$. Para utilizar o ANS, é necessário mapear essa distribuição para uma tabela de frequências inteiras $\{f_1, f_2, ..., f_n\}$ tal que:

$$\sum_{i=1}^{n} f_i = M \tag{20}$$

onde M é um inteiro fixo que representa o total de slots disponíveis na tabela de símbolos (normalmente uma potência de 2, como $M = 2^k$). A discretização é feita com:

$$f_i = \max(1, \lfloor p_i \cdot M \rfloor) \tag{21}$$

Essa etapa assegura que cada símbolo receba ao menos um slot na tabela, mesmo que sua probabilidade seja muito baixa. O valor de M controla a resolução da codificação: quanto maior M, mais precisamente a tabela representa a distribuição original, mas maior será a memória usada.

Após a discretização, calcula-se o *deslocamento base* b_s de cada símbolo s, que determina onde na tabela começa o intervalo reservado a ele. Isso é necessário para garantir a reversibilidade da operação de codificação:

$$b_s = \sum_{i=1}^{s-1} f_i \tag{22}$$

Esse deslocamento é usado diretamente na função de codificação:

$$x' = \left\lfloor \frac{x}{f_s} \right\rfloor \cdot M + (x \mod f_s) + b_s \tag{23}$$

Durante a decodificação, o processo é inverso: com base em $x' \mod M$, determina-se qual símbolo foi codificado, acessando a posição correspondente na tabela de decodificação. Essa tabela pode ser construída diretamente a partir das frequências f_s, alocando cada símbolo s em f_s posições sequenciais.

Suponha uma distribuição simbólica com probabilidades reais estimadas:

Tabela 5 – Distribuição de probabilidades estimada.

Símbolo	Probabilidade (p_s)
A	0.5
B	0.33
C	0.17

Fonte: Autor (2025).

Utilizando $M = 12$, temos:

- $f_A = \lfloor 0.5 \cdot 12 \rfloor = 6$
- $f_B = \lfloor 0.33 \cdot 12 \rfloor = 3$
- $f_C = \lfloor 0.17 \cdot 12 \rfloor = 2$ (ajustado para garantir $\sum f_s = 11 \Rightarrow$ adiciona-se 1 para completar 12)

Os deslocamentos b_s seriam então:

$$b_A = 0, \quad b_B = f_A = 6, \quad b_C = f_A + f_B = 6 + 3 = 9$$

Esses valores serão usados tanto na codificação quanto na construção da tabela de decodificação. A seguir, os índices de 0 a 11 (M = 12) são preenchidos com os símbolos conforme suas frequências:

$$\text{Tabela: } [A,A,A,A,A,A,B,B,B,C,C,C]$$

Embora o ANS tradicionalmente use distribuições estáticas, é possível empregar técnicas de estimação adaptativa para recalcular as frequências conforme mais dados são processados. Dentre as abordagens modernas mais utilizadas, destacam-se:

- **Janelas deslizantes**: a distribuição é reestimada com base nos últimos N símbolos observados.
- **Decaimento exponencial**: frequências são ajustadas com base em pesos decrescentes ao longo do tempo.
- **Estimativa Bayesiana**: atualização das frequências com regularização para evitar zero frequente.

Essas abordagens são particularmente úteis em cenários com dados não estacionários, como fluxos de vídeo em tempo real, sensores IoT ou ativações de redes neurais durante a inferência.

A modelagem adequada das probabilidades no ANS é o principal fator que determina a proximidade da taxa de compressão à entropia teórica da fonte. Um modelo mal ajustado resulta em alocações incorretas na tabela e, portanto, em desperdício de bits. Esse componente é o elo fundamental entre a teoria da informação e a aplicação prática no ANS.

Considere um alfabeto simples $\mathscr{A} = \{a,b,c\}$ com as seguintes frequências associadas:

Tabela 6 – Exemplo de tabela de codificação ANS.

Símbolo	Frequência (f_s)	Base (b_s)
a	2	0
b	3	2
c	1	5

Fonte: Autor (2025).

Neste exemplo, $m = 2+3+1 = 6$. Supondo um estado atual $x = 31$ e que desejamos codificar o símbolo b com $f_b = 3$ e $b_b = 2$, a nova atualização de estado será:

$$x' = \left\lfloor \frac{31}{3} \right\rfloor \cdot 6 + (31 \mod 3) + 2 = 10 \cdot 6 + 1 + 2 = 60 + 3 = 63 \tag{24}$$

Esse processo pode ser revertido durante a decodificação ao realizar a operação inversa, identificando o símbolo a partir de x', recuperando x e o símbolo original.

4.3 Aplicações

Desde sua introdução, o ANS tem sido amplamente adotado em algoritmos de compressão modernos. Dentre as principais aplicações práticas, destacam-se:

- **Zstandard (Facebook)**: Utiliza uma variante do ANS como codificador entrópico padrão para compressão geral de dados, oferecendo alta velocidade e excelente taxa de compressão.

- **LZFSE (Apple)**: Emprega o ANS para compressão de dados em dispositivos iOS e macOS, priorizando eficiência energética e desempenho.

- **Google Draco**: Utiliza o ANS na compressão de malhas 3D para WebGL e aplicações em realidade aumentada.

- **Codificadores neurais**: O ANS também tem sido explorado como backend de compressão em arquiteturas autoencoder para imagens, como em Ballé et al. (BALLÉ et al., 2018), integrando codificação eficiente com aprendizado profundo.

A elegância conceitual e a eficiência prática do ANS tornam-no uma das contribuições mais importantes para a codificação entrópica nos últimos anos. Sua capacidade de operar com baixa complexidade, próxima ao tempo constante, e sua compatibilidade com implementações paralelas, fazem do ANS uma escolha estratégica para sistemas modernos que exigem compressão de alta performance.

Nos tópicos seguintes, serão discutidas as variantes do ANS desenvolvidas para aplicações específicas, como o **rANS** (range ANS), **tANS** (tabled ANS), cada uma com características particulares de eficiência, tabelamento e paralelização.

4.4 Range Asymmetric Numeral Systems (rANS)

O *Range Asymmetric Numeral Systems* (rANS) é uma das variantes do *Asymmetric Numeral Systems* (ANS), um método de codificação entrópica desenvolvido por Jarek Duda entre os anos de 2006 e 2013 (DUDA, 2013). Eficiente em ambientes de compressão de fluxo de dados, é amplamente adotado em algoritmos modernos como *Zstandard*, *JPEG XL*, *CRAM* e outros formatos de compressão de mídia e dados. Sua principal característica é a capacidade de codificar e decodificar símbolos, mantendo um único estado inteiro que cresce conforme a codificação avança, operando sobre um intervalo controlado.

4.4.1 Funcionamento Teórico do rANS

De forma geral, o rANS mantém um estado inteiro x, que representa a codificação da sequência de símbolos até aquele ponto. A cada novo símbolo, esse estado é atualizado por meio de funções matemáticas que consideram a frequência relativa do símbolo dentro do alfabeto.

O funcionamento do rANS se baseia em duas operações principais:

- **Codificação**: Atualiza o estado x incorporando o símbolo atual.
- **Decodificação**: Recupera o símbolo a partir do estado e atualiza x de forma inversa.

O estado x deve ser mantido dentro de um intervalo previamente definido, normalmente representado como $[L, b \cdot L)$, onde b é a base, geralmente uma potência de 2, para facilitar as operações de normalização e manipulação de bits.

4.4.1.1 Propriedades

Dada uma distribuição de símbolos com:

- f_s — frequência do símbolo s,
- C_s — valor cumulativo inicial do símbolo s na tabela cumulativa,
- M — soma total das frequências (geralmente uma potência de 2).

As operações são definidas da seguinte forma:

1. Calcula-se:

$$r = x \bmod M \tag{25}$$

2. Determina-se o símbolo s tal que:

$$C_s \leq r < C_{s+1} \tag{26}$$

3. Atualiza-se o estado:

$$x' = f_s \cdot \left\lfloor \frac{x}{M} \right\rfloor + (r - C_s) \tag{27}$$

A codificação e decodificação operam simetricamente, e o processo de normalização garante que o estado permaneça dentro de um intervalo seguro, evitando estouros e mantendo a eficiência do bitstream.

Considere um alfabeto simples composto pelos símbolos {A, B}, com as seguintes frequências:

- $f_A = 3$
- $f_B = 1$

A soma total das frequências é $M = 4$. A tabela cumulativa é definida como:

- $C_A = 0$
- $C_B = 3$

Vamos codificar a sequência `A B A`, partindo de um estado inicial $x = 1$.

1. **Codificando A:**

$$x' = \left\lfloor \frac{1}{3} \right\rfloor \cdot 4 + 0 + (1 \bmod 3) = 0 \cdot 4 + 1 = 1$$

Estado permanece em 1.

2. **Codificando B:**

$$x' = \left\lfloor \frac{1}{1} \right\rfloor \cdot 4 + 3 + (1 \bmod 1) = 1 \cdot 4 + 3 + 0 = 7$$

Estado passa a ser 7.

3. **Codificando A novamente:**

$$x' = \left\lfloor \frac{7}{3} \right\rfloor \cdot 4 + 0 + (7 \bmod 3)$$

$$x' = 2 \cdot 4 + 1 = 8 + 1 = 9$$

Estado final é 9.

O número 9 representa toda a sequência `A B A`.

4.4.1.2 Decodificação

Para decodificar, realiza-se o processo inverso:

1. Com $x = 9$, calcula-se $r = 9 \bmod 4 = 1$. Como $0 \leq 1 < 3$, o símbolo é A. Atualiza-se:
$$x = 3 \cdot \left\lfloor \frac{9}{4} \right\rfloor + (1 - 0) = 3 \cdot 2 + 1 = 7$$
2. Com $x = 7$, calcula-se $r = 7 \bmod 4 = 3$. Como $3 \leq 3 < 4$, o símbolo é B. Atualiza-se:
$$x = 1 \cdot \left\lfloor \frac{7}{4} \right\rfloor + (3 - 3) = 1 \cdot 1 + 0 = 1$$
3. Com $x = 1$, calcula-se $r = 1 \bmod 4 = 1$. Como $0 \leq 1 < 3$, o símbolo é A. Atualiza-se:
$$x = 3 \cdot \left\lfloor \frac{1}{4} \right\rfloor + (1 - 0) = 3 \cdot 0 + 1 = 1$$

Ao final do processo, recupera-se a sequência original `A B A`.

O rANS apresenta diversas vantagens em relação aos métodos tradicionais de codificação entrópica:

- Eficiência computacional elevada, operando predominantemente com multiplicações, divisões inteiras e operações de bits.
- Taxa de compressão próxima ao limite teórico de Shannon, equivalente ou superior à codificação aritmética.
- Otimização para arquiteturas modernas, especialmente com suporte a SIMD e buffers.
- Amplo uso em padrões industriais devido ao baixo overhead e alta performance.

4.5 Codificação Streaming-rANS

A codificação *Streaming-rANS* é uma variação prática do algoritmo *rANS* (*range Asymmetric Numeral Systems*), projetada para operar de maneira eficiente sobre fluxos contínuos de dados ou grandes volumes de entrada. No modelo tradicional do *rANS*, o estado interno cresce a cada símbolo codificado, podendo eventualmente exceder os limites da máquina, como, por exemplo, um valor máximo de $H = 2^{64}$. Para contornar esse problema, a abordagem streaming propõe mecanismos de controle do crescimento do estado, garantindo que a compressão possa ser realizada de forma contínua e sem perdas.

4.5.1 Delimitação do Estado

A principal estratégia da versão *streaming* consiste em restringir o estado X a permanecer dentro de um intervalo fixo $I = [L, H]$. Durante o processo de codificação, se a aplicação da função $C_{\text{rANS}}(X_{t-1}, s_t)$ resultar em um novo estado X_t que ainda pertence ao intervalo I, nenhuma ação adicional é necessária. No entanto, se X_t ultrapassar esse intervalo, é necessário reduzir o estado X_{t-1} retirando bits menos significativos até que ele pertença ao intervalo adequado I_{s_t}, específico para o símbolo s_t a ser codificado.

4.5.2 Intervalos por Símbolo

Cada símbolo $a \in \mathscr{A}$ do alfabeto possui um intervalo associado $I_a = [L_a, H_a]$, tal que:

$$\forall z \in I_a, \quad C_{\text{rANS}}(z, a) \in I$$

$$\text{e, para } z \notin I_a, \quad C_{\text{rANS}}(z, a) \notin I$$

Para garantir essa propriedade, os intervalos I_a devem satisfazer a condição:

$$H_a \geq 2L_a - 1$$

Esse critério assegura que, ao retirar bits de X_{t-1} (isto é, realizar divisões sucessivas por 2), o estado resultante não "salte"completamente o intervalo I_a, o que inviabilizaria a codificação correta.

4.5.3 Mapeamento de Estado e BitStream

O mapeamento do estado é feito por meio de divisões sucessivas:

$$X'_{t-1} = \left\lfloor \frac{X_{t-1}}{2^r} \right\rfloor$$

onde r é o menor inteiro tal que $X'_{t-1} \in I_{s_t}$. Os r bits descartados ($b_{t-1} = X_{t-1} \bmod 2^r$) são armazenados em um *bitstream* auxiliar.

Durante a decodificação, o processo reverso é realizado: o decodificador recupera os bits do *bitstream* e os adiciona ao estado mapeado até que ele pertença novamente ao intervalo I:

$$X_{t-1} = X'_{t-1} \cdot 2^r + b_{t-1}$$

4.5.4 Ambiguidade na Decodificação

Pode haver múltiplos estados X_{t-1} possíveis para um dado X'_{t-1} e sequência de bits b_{t-1} que estejam contidos em I. Para evitar essa ambiguidade, é necessário garantir que o intervalo global $I = [L, H]$ satisfaça:

$$H \leq 2L - 1$$

Essa restrição assegura que a reconstrução do estado seja unívoca, evitando erros de decodificação por parada prematura na leitura dos bits do *bitstream*.

4.5.5 Parâmetros Sugeridos

Jarek Duda propôs uma configuração simples para satisfazer todas as condições acima. Para um inteiro l, definem-se:

$$I = [lM, 2lM - 1]$$

$$I_a = [lF_a, 2lF_a - 1]$$

onde M é a soma das frequências de todos os símbolos, e F_a é a frequência do símbolo a. Essa parametrização assegura tanto a validade do mapeamento direto quanto da

recuperação reversa, garantindo robustez e eficiência ao *Streaming-rANS*.

4.5.6 Otimização e Convergência à Entropia

Ao restringir o estado do codificador ao intervalo finito $I = [L, H]$, a versão *Streaming-rANS* perde, em teoria, a garantia de que o comprimento médio dos códigos se iguala à entropia da fonte. A análise de otimalidade se torna mais complexa, pois deve incluir também o tamanho do *bitstream* auxiliar, cuja dimensão pode variar de acordo com a ordem dos símbolos na sequência de entrada.

Apesar disso, estudos demonstram que, à medida que o parâmetro de escala l aumenta, o desempenho do *Streaming-rANS* converge assintoticamente para a entropia. Essa propriedade foi discutida por Yokoo em sua pesquisa, que apresentou uma aproximação da distribuição dos estados dentro do intervalo I.

Embora ainda não haja uma demonstração teórica completa da taxa de convergência, Jarek Duda propôs uma relação empírica para o comprimento médio L_{avg} dos códigos gerados por um *Streaming-rANS* com parâmetro l:

$$L_{\text{avg}} = H + \mathcal{O}\left(\frac{1}{l^2}\right)$$

Essa expressão sugere que o excesso de bits por símbolo decresce quadraticamente à medida que l aumenta. A proposta é sustentada por evidências experimentais, embora careça de uma prova formal.

4.5.7 Aprimoramentos Práticos

Além da fundamentação teórica, diversas melhorias práticas podem ser aplicadas ao *Streaming-rANS*, tornando-o uma alternativa altamente competitiva a métodos como codificação aritmética e *range coding*.

4.5.7.1 Saída em Blocos de Bits

Embora a arquitetura original do *Streaming-rANS* opere extraindo e inserindo um bit por vez, na prática, é mais eficiente manipular blocos maiores, como bytes (8 bits) ou palavras de 64 bits. Para viabilizar essa estratégia, basta ajustar os intervalos de estado da

seguinte forma:

$$I = [lM, 2^k lM - 1] \quad \text{e} \quad I_a = [lF_a, 2^k lF_a - 1]$$

onde k representa o número de bits extraídos por etapa. Esse ajuste garante que a extração de múltiplos bits seja compatível com a dinâmica do codificador, mantendo a reversibilidade do processo.

4.5.7.2 Codificação Paralela (SIMD)

Fabian Giesen (GIESEN, 2014) apresentou um método eficiente para paralelizar a codificação e decodificação do *Streaming-rANS* utilizando instruções vetoriais (como SSE). Essa abordagem permite processar múltiplos fluxos simultaneamente, aproveitando ao máximo o paralelismo das arquiteturas modernas de hardware, o que resulta em ganhos expressivos de desempenho.

4.5.7.3 Uso de Cache e Tabelas de Codificação

Como o processo de codificação mapeia estados entre intervalos finitos e extrai bits para o *bitstream*, o *Streaming-rANS* pode ser modelado como um codificador de estados finitos — uma característica típica de um *Finite State Entropy* (FSE) encoder. Isso permite que as operações de codificação e decodificação sejam otimizadas por meio de tabelas de consulta (lookup tables), onde os resultados de cada transição de estado são armazenados previamente.

Essa técnica permite acelerar o algoritmo de forma significativa, aproximando sua velocidade da codificação de Huffman, com desempenho muito superior ao de algoritmos aritméticos tradicionais. Essa versão otimizada pertence à classe dos codificadores *tANS* (*tabled Asymmetric Numeral Systems*), cuja estrutura será detalhada na próxima seção.

A versão *Streaming* do *rANS* representa uma solução altamente eficiente e prática para compressão de dados contínuos, com forte fundamentação teórica e grande potencial de otimização. A combinação de controle de estado, extração de bits ajustável, paralelismo e uso de tabelas torna o *Streaming-rANS* uma ferramenta poderosa para aplicações em tempo real e sistemas embarcados.

Considere o seguinte exemplo com dois símbolos A e B, cujas frequências são definidas como $\mathscr{F} = \{4, 2\}$. Isso implica que o símbolo A ocorre com frequência 4 e o símbolo B com frequência 2, totalizando $M = 6$ estados distintos no codificador.

No contexto do *Streaming-rANS*, o codificador trabalha com um conjunto de estados válidos, aqui representados pelos inteiros no intervalo $[6, 11]$. Cada entrada na tabela representa a transição do estado interno após a codificação de um símbolo. Como o intervalo de estados foi expandido para $[6, 11]$ (ou seja, $[lM, 2lM - 1]$ com $lM = 6$), é necessário emitir bits para o *bitstream* sempre que o estado cresce além do permitido. Esses bits representam o "excedente"extraído do estado para manter a normalização.

As Tabelas 7 e 8 abaixo mostram os resultados:

Tabela 7 – Output State - Streaming-rANS as FSE

Input State	**A**	**B**
6	8	11
7	9	11
8	6	10
9	6	10
10	7	10
11	7	10

Fonte: Autor (2025).

Tabela 8 – BitStream Output - Streaming-rANS as FSE

Input State	**A**	**B**
6	0	
7	1	
8	0	00
9	1	10
10	0	01
11	1	11

Fonte: Autor (2025).

A Tabela 8 indica os bits emitidos para o *bitstream* durante o processo de codificação. Esses bits são os menos significativos extraídos do estado atual, com o objetivo de mantê-lo dentro dos limites do intervalo válido. Por exemplo, ao codificar *B* a partir do estado 8, os bits `00` são emitidos, e o novo estado será 10 conforme a 7.

Este exemplo ilustra o funcionamento do *Streaming-rANS* com tabelas pré-calculadas, característica essencial da classe de algoritmos conhecida como *Finite*

State Entropy (FSE). Cada entrada na tabela representa uma operação determinística que codifica um símbolo e atualiza o estado interno, ao mesmo tempo que emite uma sequência de bits. A reversibilidade perfeita é garantida, desde que os estados e os bits sejam lidos exatamente na ordem inversa durante a decodificação.

4.6 Table-based Asymmetric Numeral Systems (tANS)

A variante tANS (Table-based Asymmetric Numeral Systems) é uma das implementações mais eficientes da família ANS, desenvolvida por Jarek Duda a partir de 2009 (DUDA, 2013), como uma alternativa moderna à codificação aritmética. O tANS combina a eficiência de compressão próxima ao limite entrópico com operações de baixa complexidade, utilizando tabelas pré-computadas em vez de multiplicações e divisões em tempo real.

Diferente da codificação de Huffman, que associa códigos de comprimento fixo a cada símbolo, ou da codificação aritmética, que representa toda a mensagem como um intervalo real, o ANS trabalha com o conceito de um **estado inteiro**, que representa de forma compacta a informação acumulada. Cada símbolo processado transforma esse estado em outro estado, com a operação de codificação consistindo basicamente em adicionar entropia ao estado, e a decodificação em extrair entropia.

4.6.1 Streaming e Bitstream no tANS

A versão *tabled* do Asymmetric Numeral Systems, conhecida como *tANS*, introduz uma abordagem eficiente para compressão de dados baseada em tabelas de transição. Assim como o *Streaming-rANS*, o *tANS* também opera de forma contínua (*streaming*), mas com a vantagem adicional de utilizar estruturas discretas e determinísticas, que favorecem a implementação com alto desempenho computacional.

4.6.1.1 Codificação por Estados

O algoritmo mantém um estado inteiro x que pertence a um intervalo fixo de estados válidos $\mathscr{X} = [L, H]$, onde normalmente $H = 2^n - 1$ para algum n. A codificação de um símbolo s é definida por uma tabela que fornece dois valores principais:

- O novo estado: $x' = \texttt{nextState}(x, s)$
- Os bits a serem emitidos: $b = x \bmod 2^r$

Sempre que o estado atual x ultrapassa um limite mínimo L_s associado ao símbolo s, o codificador emite r bits menos significativos para um *bitstream* auxiliar e reduz o estado:

$$x \leftarrow \left\lfloor \frac{x}{2^r} \right\rfloor$$

Essa operação garante que o novo estado x' continue dentro da faixa de estados válida após a codificação do próximo símbolo. O valor de r depende da granularidade desejada: normalmente são utilizados múltiplos de 8 (um byte) ou 16/32 bits, para melhor desempenho em arquiteturas modernas.

4.6.1.2 Decodificação e Reconstrução de Estado

Na decodificação, o processo ocorre de forma inversa ao da codificação. A partir do estado atual x, o decodificador consulta uma tabela para recuperar o símbolo original s e o número de bits r_s que precisam ser lidos do *bitstream*:

$$s = \texttt{symbol}(x), \quad r_s = \texttt{nbBits}(x)$$

O estado anterior x' é então reconstruído como:

$$x' = \texttt{base}(s) + (x \gg r_s) + \texttt{readBits}(r_s)$$

Esse mecanismo garante a perfeita reversibilidade do processo, mesmo utilizando um intervalo limitado de estados. Contudo, ele impõe uma limitação importante: a decodificação no *tANS* deve ser realizada obrigatoriamente em ordem inversa à da codificação. Ou seja, o primeiro símbolo codificado será o último a ser decodificado.

Essa característica decorre da natureza do ANS como uma pilha de operações sobre o estado: cada símbolo codificado empilha informação no estado x, e somente ao desfazer essa pilha (i.e., lendo o *bitstream* de trás para frente) é possível recuperar corretamente a sequência original. Como consequência, o decodificador precisa ter acesso ao final do estado e ao final do *bitstream* para iniciar o processo de reconstrução, o que inviabiliza a decodificação em tempo real diretamente à medida que os dados são transmitidos.

Essa restrição representa um dos principais desafios na aplicação do *tANS* em fluxos contínuos (*streaming*) ou em sistemas com requisitos de baixa latência. Para superar esse problema, é comum dividir a entrada em blocos e aplicar o tANS separadamente a cada bloco, o que permite a decodificação por partes, embora com alguma perda de eficiência e aumento do overhead de controle.

4.6.1.3 Operação em Fluxo e Eficiência

O *tANS* foi projetado para operar em modo *streaming*, ou seja, processando dados sequencialmente, com saída e entrada incremental de bits. O *bitstream* age como uma memória auxiliar de baixa entropia, armazenando os bits excedentes que não cabem no estado x em determinado momento. O equilíbrio entre os estados válidos e a taxa de emissão/leitura de bits é essencial para manter o desempenho e garantir a compressão próxima à entropia.

Segundo Duda (DUDA, 2013), o desempenho do *tANS* pode se aproximar da entropia H da fonte com um pequeno excesso médio dado por:

$$L_{\text{avg}} = H + \mathcal{O}\left(\frac{1}{M}\right)$$

onde M é o tamanho da tabela de estados. Isso o torna competitivo com a codificação aritmética, porém com maior desempenho devido à ausência de operações com ponto flutuante.

4.6.1.4 Implementações Eficientes

O uso de tabelas permite otimizações como:

- Implementação via *lookup tables*, com desempenho comparável à codificação de Huffman.
- Paralelização via instruções SIMD (Single Instruction, Multiple Data), conforme demonstrado por Giesen em (GIESEN, 2014), utilizando conjuntos como SSE/AVX.
- Pré-processamento de todas as transições e armazenamento em cache, eliminando a necessidade de cálculos dinâmicos durante a codificação e decodificação.

Assim como no *Streaming-rANS*, o uso de um intervalo fixo de estados e de um

bitstream complementar no *tANS* permite compressão eficiente e reversível. A diferença fundamental reside na utilização de tabelas pré-computadas, que conferem ao *tANS* vantagens significativas em termos de velocidade e paralelismo, mantendo-se próximo do limite de entropia com sobrecarga mínima.

4.6.2 Aplicações

O tANS está presente em diversos sistemas modernos de compressão, entre eles:

- **Zstandard (Facebook)**: Algoritmo de compressão de propósito geral que substitui o zlib/deflate.
- **JPEG XL**: Novo padrão de compressão de imagem que utiliza tANS para codificação de entropia (ISO/IEC JTC 1/SC29/WG1, 2023).
- **Streaming de vídeo**: tANS é usado em algumas variações de codificadores AV1 e VVC devido à sua baixa latência.
- **Vantagens**:
  - Eficiência de compressão próxima à codificação aritmética
  - Muito mais rápido: apenas operações de `lookup` e `shift`
  - Altamente paralelizável e adequado para implementações em hardware
- **Desvantagens**:
  - Complexidade na construção das tabelas de codificação e decodificação
  - Pouco intuitivo, especialmente com codificação reversa
  - Requer armazenamento adicional para as tabelas
  - Menos eficiente em fluxos dinâmicos

O tANS representa um avanço significativo na codificação entrópica moderna, oferecendo uma combinação rara entre desempenho de compressão, velocidade computacional e adequação a ambientes de alta demanda como em transmissão multimídia. Sua adoção crescente em padrões industriais demonstra seu valor prático, enquanto seu funcionamento matemático elegante contribui para o estado da arte em compressão de dados.

5 DESENVOLVIMENTO E RESULTADOS DO TANS DINÂMICO

O capítulo a seguir apresenta o processo de desenvolvimento do tANS adaptativo e os principais resultados obtidos a partir da implementação e avaliação da abordagem proposta ao longo deste trabalho. Foram realizados diferentes testes e experimentos com o objetivo de analisar o desempenho do método desenvolvido e verificar sua eficácia em relação aos critérios estabelecidos nos objetivos da pesquisa.

Os resultados aqui descritos fornecem uma visão geral sobre o comportamento do sistema em distintos cenários de uso, permitindo observar tendências, identificar pontos fortes e compreender eventuais limitações da solução proposta. A análise apresentada busca interpretar os achados de forma clara e objetiva, destacando aspectos relevantes para a discussão posterior e para a formulação das conclusões finais do estudo.

5.1 Primeira etapa do desenvolvimento

A escolha do tANS como ponto de partida justifica-se por sua eficiência prática e seu uso consolidado em sistemas de compressão modernos. A estrutura do algoritmo foi desenvolvida em linguagem C, visando atender aos requisitos de desempenho e controle de memória exigidos por sistemas embarcados e aplicações em tempo real, que constituem o foco da pesquisa. Cabe destacar que o enfoque da implementação será voltado para alfabetos binários, devido à sua ampla aplicação em cenários reais, como a codificação de vídeo (por exemplo, VVC), onde estruturas binárias favorecem a compressão eficiente e a compatibilidade com arquiteturas modernas, além de tornar o desenvolvimento do trabalho mais acessível.

Nesta versão preliminar, o usuário fornece como entrada o conjunto de símbolos e suas respectivas frequências. A partir desses dados, o algoritmo realiza a normalização das frequências com base em uma potência de dois (L), constrói as tabelas de codificação e decodificação, e imprime os mapeamentos finais que serão utilizados para o processamento dos dados. Essa implementação estabelece as bases necessárias para validar o funcionamento do ANS tabelado e serve como referência para futuras extensões adaptativas.

A seguir, são detalhadas a estrutura do código, a lógica de geração das tabelas e exemplos ilustrativos que comprovam a correta construção dos elementos centrais do algoritmo.

5.1.1 Descrição da Implementação Preliminar do tANS

Nesta etapa do trabalho foi desenvolvida uma implementação inicial do tANS em linguagem C, utilizada para gerar as tabelas de transição de estados e de emissão de bits que compõem o núcleo do codificador. Essa implementação permite ao usuário definir manualmente o conjunto de símbolos e suas frequências, possibilitando observar, em detalhe, como o algoritmo constrói sua estrutura interna a partir da distribuição estatística fornecida. Cada parte do código foi separada em blocos lógicos e comentada, permitindo relacionar diretamente o funcionamento do tANS com sua formalização teórica.

Algorithm 1 Inicialização e leitura dos dados de entrada

1: **Entrada:** Número de símbolos e suas frequências
2: **Saída:** Vetores de frequência `F[]` e cumulativa `C[]`
3: Abrir arquivo para escrita em "`C:\Tabela\tabelas_rans.txt`"
4: Ler do usuário o número de símbolos
5: Alocar vetores `F[]` e `C[]` dinamicamente
6: **for** cada símbolo *i* **do**
7: Ler a frequência de ocorrência de *i*
8: **end for**

Fonte: Autor (2025).

O trecho de código 1 trata da etapa de inicialização e preparação dos dados. Primeiramente, o programa abre um arquivo de saída para registrar todas as tabelas geradas, o que facilita tanto a depuração quanto a análise experimental dos resultados. Em seguida, são solicitados ao usuário o número de símbolos e suas respectivas frequências absolutas. Com base nessas informações, o programa aloca dinamicamente os vetores `F[]` e `C[]`, que armazenam as frequências e as frequências cumulativas, fundamentais para o funcionamento das máquinas tANS. Essa organização modular permite que diferentes distribuições sejam testadas sem alterar o código-fonte, proporcionando flexibilidade na exploração dos cenários de compressão.

Algorithm 2 Construção da Tabela Cumulativa

1: $C[0] \leftarrow 0$
2: **for** $i = 1$ até $n-1$ **do**
3: $\quad C[i] \leftarrow C[i-1] + F[i-1]$
4: **end for**
5: $sum\_F \leftarrow C[n-1] + F[n-1]$
6: $MIN\_STATE \leftarrow sum\_F$
7: $MAX\_STATE \leftarrow 2 \cdot sum\_F$

Fonte: Autor (2025).

O trecho 2 implementa o cálculo da tabela cumulativa, elemento central da construção do tANS. A cumulativa define, para cada símbolo, o intervalo de estados que lhe é reservado dentro da tabela de transição. Além disso, o algoritmo determina o valor total de frequências `sum_F`, que corresponde ao tamanho lógico da tabela tANS. A partir desse valor, são definidos os limites `MIN_STATE` e `MAX_STATE`, que especificam o espaço de estados em que o codificador opera. Esses limites serão utilizados tanto na fase de construção das tabelas quanto na execução do processo de compressão, garantindo que todos os estados estejam dentro da região válida de operação do tANS.

Algorithm 3 Construção da Tabela de Transição de Estados

1: Imprimir cabeçalho da tabela com os símbolos
2: **for** cada estado x em $[MIN\_STATE, MAX\_STATE)$ **do**
3: $\quad$ **for** cada símbolo s **do**
4: $\quad\quad Fs \leftarrow F[s]$
5: $\quad\quad Cs \leftarrow C[s]$
6: $\quad\quad norm\_x \leftarrow x$
7: $\quad\quad$ **while** $norm\_x \geq 2 \cdot Fs$ **do**
8: $\quad\quad\quad norm\_x \leftarrow norm\_x/2$
9: $\quad\quad$ **end while**
10: $\quad\quad next\_x \leftarrow (norm\_x/Fs) \cdot sum\_F + Cs + (norm\_x\%Fs)$
11: $\quad\quad$ Imprimir $next\_x$ como estado de transição
12: $\quad$ **end for**
13: **end for**

Fonte: Autor (2025).

A etapa apresentada em 3 corresponde à construção explícita da tabela de transição

de estados. Para cada estado x e símbolo s, o algoritmo calcula o próximo estado que será assumido após a emissão de s. O processo de normalização realizado no interior da estrutura de repetição `while` reproduz o mecanismo fundamental do tANS: garantir que o estado esteja dentro da faixa adequada para representar o símbolo, permitindo sua posterior renormalização por emissão de bits. Esse procedimento aproxima o comportamento do tANS real, no qual os estados pertencentes ao intervalo alto (próximo a `MAX_STATE`) são responsáveis por desencadear a saída de bits. A fórmula que calcula o próximo estado (`next_x`) implementa exatamente a distribuição proporcional dos estados entre os símbolos, preservando as frequências relativas definidas em `F[]`.

Algorithm 4 Construção da Tabela de Bits (Bitstream)

1: **for** cada estado x em $[MIN\_STATE, MAX\_STATE)$ **do**
2: Calcular os bits de normalização geral ($F[0]$)
3: Imprimir bits (ou − se nenhum)
4: **for** cada símbolo s **do**
5: Calcular quantidade de bits de normalização com base em $F[s]$
6: Imprimir sequência de bits calculada (ou espaço em branco)
7: **end for**
8: **end for**

Fonte: Autor (2025).

O trecho 4 produz a tabela de emissão de bits, complementando o modelo de transição construído anteriormente. Para cada estado válido, o algoritmo determina a quantidade de bits que seriam emitidos durante a renormalização do estado para o símbolo atual. Essa etapa é crucial, pois fornece a representação binária que será efetivamente escrita no fluxo de saída durante a compressão. A implementação considera tanto o caso geral (normalização baseada em `F[0]`) quanto a emissão específica para cada símbolo, refletindo exatamente a lógica do processo real de codificação tANS, no qual o estado carrega parte da informação e os bits emitidos completam a representação estatisticamente ótima do símbolo.

Algorithm 5 Ajuste de Frequências e Balanceamento

1: Encontrar índice do símbolo com maior e menor frequência
2: **if** diferença de frequência > 1 **then**
3: $F[max] \leftarrow F[max] - 1$
4: $F[min] \leftarrow F[min] + 1$
5: Repetir construção das tabelas
6: **else**
7: Finalizar execução
8: **end if**

Fonte: Autor (2025).

Por fim, o código 5 implementa um mecanismo inicial de balanceamento das frequências, ajustando iterativamente os valores de `F[]` quando há uma grande disparidade entre símbolos. Embora essa etapa ainda não constitua um modelo adaptativo completo, ela simula variações graduais na distribuição de probabilidades, funcionando como uma aproximação preliminar das técnicas que serão utilizadas no ANS adaptativo. Esse procedimento permite observar como a tabela tANS responde a pequenas alterações nas frequências, na qual as frequências serão atualizadas a cada determinado número de símbolos processados.

```
Quantos simbolos deseja usar? 2
Digite as frequencias dos simbolos (ex: 3 3 2):
Frequencia de A: 4
Frequencia de B: 2

=== Output State Table ===
State    | A       | B       |
--------+--------+--------+
6        | 8       | 11      |
7        | 9       | 11      |
8        | 6       | 10      |
9        | 6       | 10      |
10       | 7       | 10      |
11       | 7       | 10      |

=== BitStream Output Table ===
State    | A       | B       |
--------+--------+--------+
6        | -       | 0       |
7        | -       | 1       |
8        | 0       | 00      |
9        | 1       | 10      |
10       | 0       | 01      |
11       | 1       | 11      |
```

Figura 6 – Exemplo de execução do algoritmo com dois símbolos

Fonte: Autor (2025).

A Figura 6 mostra a execução do algoritmo de geração de tabelas para o tANS com dois símbolos: `A` e `B`. As frequências iniciais fornecidas pelo usuário foram 4 para `A` e 2 para `B`. O programa então calcula os vetores de frequência cumulativa e gera a tabela de transição de estados (Output State Table), que indica para cada estado possível qual será o próximo estado após codificar um símbolo específico.

Na tabela de estados de saída, por exemplo, ao codificar o símbolo `A` no estado 6, o próximo estado será 8; ao codificar `B` no mesmo estado, o próximo será 11. Esses valores são calculados com base na normalização e redistribuição dos estados dentro do intervalo definido entre L e $2L$, onde L é a soma das frequências.

Logo abaixo, a BitStream Output Table mostra a sequência de bits que seria extraída do estado antes da transição, dependendo do símbolo codificado. Nos estados iniciais, os símbolos com maiores frequências (`A`) tendem a exigir menos bits (ou nenhum), enquanto símbolos menos frequentes (`B`) necessitam de normalizações mais frequentes, resultando em mais bits de saída.

Conforme descrito no Algoritmo 5, é feito o ajuste iterativo com os valores até que a diferença entre o símbolo mais frequente e o menos frequente seja inferior a uma unidade. A principal motivação é simular variações graduais nas frequências, como aquelas que ocorrerão em uma versão futura do algoritmo, representando uma preparação para a implementação do ANS adaptativo, em que alterações pontuais nas distribuições ocorrerão em tempo real.

A Figura 7 mostra uma iteração de ajuste automático das frequências, realizada pelo código como parte do mecanismo de balanceamento descrito no Algoritmo 5. Após a execução inicial (Figura 6), o algoritmo detecta que a diferença entre as frequências dos símbolos `A` e `B` é superior a 1 (4 para `A` e 2 para `B`). Como consequência, a maior frequência é reduzida em uma unidade e a menor é aumentada, promovendo equilíbrio.

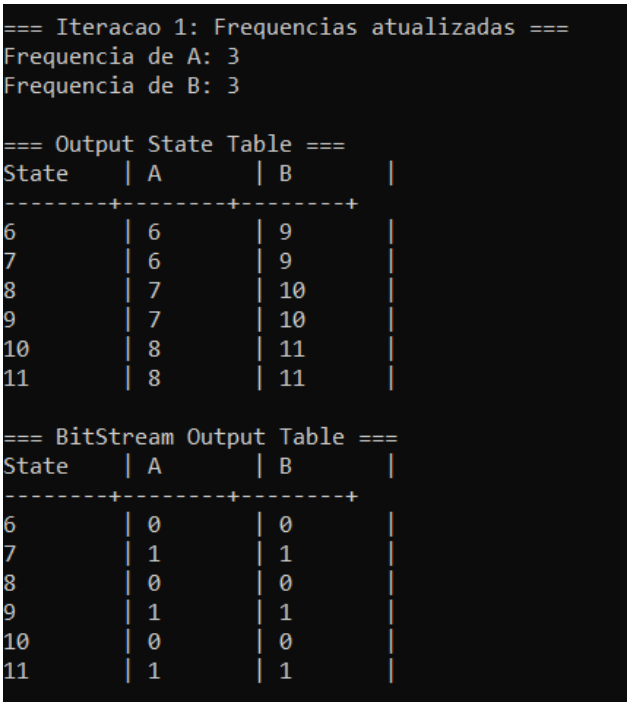

Figura 7 – Iteração de balanceamento das frequências

Fonte: Autor (2025).

Ao final da execução, todas as tabelas geradas são armazenadas em um arquivo de texto (`tabelas_rans.txt`), conforme mostrado nos algoritmos 3 e 4. Esse arquivo irá ser utilizado para a construção das tabelas que serão acessadas pelo codificador e decodificador para realizar a compressão.

5.2 tANS Dinâmico

Esta seção apresenta a arquitetura geral do codificador e decodificador baseados em *table-based Asymmetric Numeral Systems* (tANS) com adaptação dinâmica do modelo probabilístico. A proposta central do tANS dinâmico é combinar a eficiência computacional das tabelas pré-calculadas com a capacidade de adaptação estatística ao longo do fluxo de símbolos, permitindo uma compressão eficiente mesmo para fontes não estacionárias.

A Figura 8 ilustra uma visão geral da arquitetura desenvolvida, evidenciando os principais módulos do sistema, bem como o fluxo de dados entre o codificador (*encoder*) e o decodificador (*decoder*).

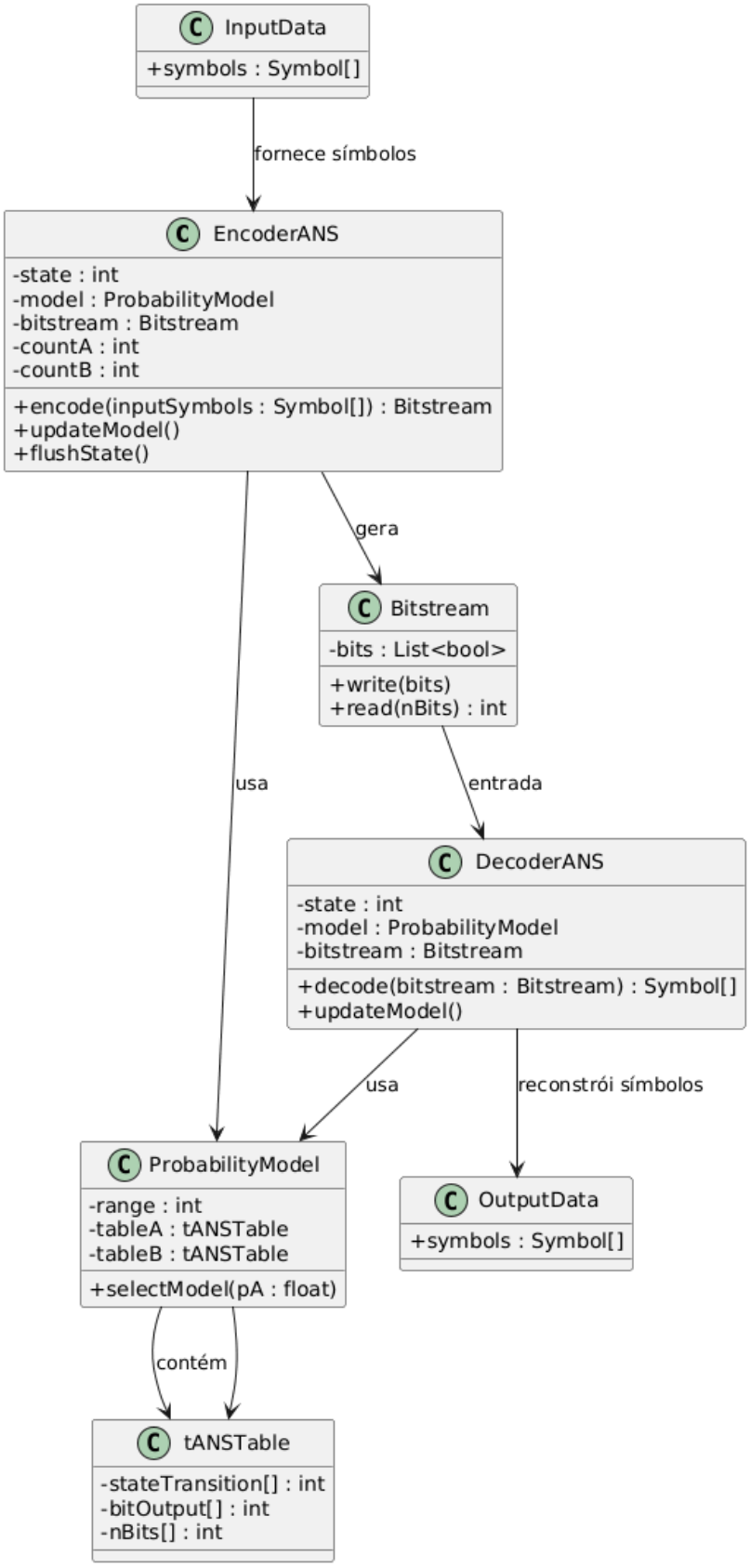

Figura 8 – Visão geral da arquitetura do tANS dinâmico.

Fonte: Autor (2025).

No nível mais alto, o sistema recebe uma sequência de símbolos de entrada, que é processada pelo módulo *EncoderANS*. Esse módulo é responsável por realizar as transições de estado do autômato tANS, emitir os bits correspondentes no *bitstream* e manter a contagem estatística dos símbolos observados. Periodicamente, conforme o intervalo adaptativo definido, o codificador atualiza o modelo probabilístico corrente, selecionando novas tabelas tANS que melhor representam a distribuição observada dos símbolos.

O modelo probabilístico é abstraído pelo módulo *ProbabilityModel*, que encapsula o conjunto de tabelas tANS associadas a um determinado *RANGE*. Essas tabelas definem, para cada estado do autômato, a transição de estado, os bits a serem emitidos e o número de bits válidos associados a cada símbolo. Essa separação permite que o codificador e o decodificador utilizem exatamente o mesmo modelo, garantindo a reversibilidade do processo.

O *bitstream* gerado pelo codificador representa a saída do sistema de compressão e contém tanto os bits emitidos durante o processo quanto o estado final do autômato, necessário para a correta inicialização do decodificador. Esse fluxo comprimido é então utilizado como entrada pelo módulo *DecoderANS*, que reconstrói a sequência original de símbolos, executando o processo inverso de transição de estados, utilizando as mesmas tabelas e critérios de adaptação empregados pelo codificador.

A organização apresentada fornece uma visão clara da interação entre os módulos e destaca o caráter adaptativo do sistema, no qual o modelo probabilístico evolui ao longo da codificação e da decodificação de forma sincronizada.

Nas subseções seguintes, o funcionamento do processo de codificação e do processo de decodificação do tANS adaptativo será detalhado de forma individual, com ênfase nos algoritmos envolvidos, na atualização do modelo estatístico e na manipulação do estado do autômato.

5.2.1 Funcionamento do Encoder tANS Adaptativo

O encoder implementado neste trabalho realiza o processo completo de compressão utilizando o *table-based Asymmetric Numeral Systems* (tANS), operando sobre um conjunto de estados inteiros no intervalo $[L, 2L-1]$. Sua finalidade é transformar uma sequência binária arbitrária em um *bitstream* comprimido, explorando a assimetria probabilística entre os símbolos da fonte.

O modelo utilizado é adaptativo: a cada número fixo de símbolos processados, o encoder reavalia as frequências observadas e substitui o modelo atual por aquele cuja distribuição melhor se aproxima das probabilidades estimadas. Esse mecanismo é especialmente relevante em sequências não estacionárias, nas quais a distribuição dos símbolos varia ao longo do tempo.

O funcionamento geral do encoder pode ser dividido em três etapas principais:

1. **Leitura e preparação da entrada:** o arquivo contendo símbolos “0” e “1” é carregado e convertido em um vetor de inteiros correspondente.
2. **Codificação tANS símbolo a símbolo:** cada símbolo é processado com base no estado corrente e nas tabelas pré-computadas de transições, número de bits emitidos e padrões de saída. O estado é atualizado e, quando necessário, bits são inseridos no *bitstream*.
3. **Adaptação estatística periódica:** após um intervalo fixo de codificação, o encoder estima as novas probabilidades e seleciona dinamicamente o modelo mais adequado.

Ao final do processo, o encoder fornece o *bitstream* comprimido, métricas de desempenho e o estado final para permitir a decodificação reversível.

Algorithm 6 Processo de Codificação tANS Adaptativo

Ler arquivo de entrada contendo símbolos $\{0,1\}$
Converter caracteres para vetor S
Inicializar estado $x \leftarrow L$ ▷ Estado inicial do autômato tANS
Inicializar *bitstream* vazio B
Inicializar modelo corrente M para o *range* selecionado
Inicializar contadores $countA \leftarrow 0$, $countB \leftarrow 0$ ▷ Contagem estatística dos símbolos
for cada símbolo s em S **do**
    if $s = 0$ **then**
        $x' \leftarrow M.a\_range[x-L]$ ▷ Transição de estado para símbolo A
        $out \leftarrow M.a\_bitstream[x-L]$ ▷ Bits a serem emitidos
        $nb \leftarrow M.a\_nbits[x-L]$ ▷ Número de bits válidos
        $countA \leftarrow countA + 1$
    else
        $x' \leftarrow M.b\_range[x-L]$ ▷ Transição de estado para símbolo B
        $out \leftarrow M.b\_bitstream[x-L]$
        $nb \leftarrow M.b\_nbits[x-L]$
        $countB \leftarrow countB + 1$
    end if
    if $nb > 0$ **then**
        Extrair os nb bits menos significativos de out e anexar ao *bitstream* B ▷ Emissão dos bits do estado
    end if
    Atualizar estado: $x \leftarrow x'$ ▷ Novo estado do autômato
    if processados $ADAPTATION\_INTERVAL$ símbolos **then** ▷ Momento de adaptação do modelo
        Calcular probabilidades: $pA = \frac{countA}{countA+countB}$
        Selecionar modelo cujo $p(A)$ é mais próximo de pA
        Atualizar modelo corrente M ▷ Troca da tabela tANS
        Resetar contadores $countA, countB$
    end if
end for
Inserir o estado final x no rodapé do *bitstream* ▷ Necessário para decodificação
return *bitstream* comprimido B

5.2.2 Descrição das Etapas do Encoder

5.2.2.1 Estruturas e Definições Globais

O encoder utiliza tabelas pré-calculadas contendo, para cada modelo probabilístico disponível:

- a tabela de transições de estado (*a_range* e *b_range*),
- o número de bits emitidos por transição (*a_nbits*, *b_nbits*),
- o padrão de bits emitidos (*a_bitstream*, *b_bitstream*).

Essas tabelas são organizadas conforme o parâmetro *RANGE*, que define o tamanho do intervalo de estados. Um valor maior de *RANGE* proporciona maior precisão estatística, enquanto aumenta o tamanho das tabelas.

5.2.2.2 Contagem de Frequências e Estimativa Probabilística

A cada k símbolos, os contadores são atualizados:

$$p(A) = \frac{countA}{countA + countB}, \qquad p(B) = 1 - p(A)$$

Essas estimativas permitem selecionar, entre todos os modelos disponíveis, aquele cuja probabilidade tabelada melhor se aproxima da distribuição observada no intervalo recente.

5.2.2.3 Codificação Símbolo a Símbolo

Para cada símbolo, o encoder recupera a linha correspondente ao estado atual $(x - L)$ e consulta as tabelas do modelo ativo para determinar:

1. o próximo estado x' a ser assumido;
2. o número de bits a serem emitidos;
3. o padrão dos bits emitidos, quando aplicável.

Caso $nbits > 0$, esses bits são anexados ao *bitstream*. Em seguida, o estado é atualizado para x'.

5.2.2.4 Adaptação Estatística

Após o processamento de um bloco de símbolos, o encoder:

1. calcula a nova probabilidade de ocorrência de cada símbolo;
2. identifica o modelo tANS cujo perfil estatístico melhor se aproxima da distribuição observada;
3. ativa esse modelo para o próximo intervalo.

Esse mecanismo torna o encoder capaz de responder a mudanças na distribuição dos símbolos, característica essencial em fontes não estacionárias.

5.2.2.5 Finalização e Estado Final

Para garantir a reversibilidade do processo, o estado final do encoder é inserido ao término de cada lote do *bitstream*. Essa estratégia permite que o decodificador reconstrua continuamente o estado inicial correspondente a cada segmento, possibilitando a decodificação progressiva ao longo do envio do *bitstream*, não apenas após o seu término.

5.3 Funcionamento do Decoder tANS Adaptativo

O decodificador implementado é responsável por reverter o processo de compressão realizado pelo codificador baseado em *table-based Asymmetric Numeral Systems* (tANS). Sua função principal consiste em reconstruir a sequência original de símbolos binários $\{0,1\}$ a partir do *bitstream* comprimido e do estado final armazenado no arquivo codificado.

5.3.1 Responsabilidade do Decodificador

O decodificador possui como objetivos principais:

- Ler o arquivo codificado contendo: o parâmetro do *range L*, o estado inicial e o *bitstream*;

- Restaurar o estado tANS utilizando exatamente o mesmo conjunto de tabelas carregado pelo codificador;
- Reproduzir cada símbolo decodificado de forma determinística, reconstruindo a sequência original;
- Aplicar a mesma adaptação periódica dos modelos probabilísticos utilizada no codificador;
- Garantir que todas as transições de estado respeitem as condições de normalização do tANS.

Assim como o codificador, o decodificador opera integralmente por consultas a tabelas pré-computadas, garantindo reprodutibilidade e reversibilidade completas.

5.3.2 Estrutura Geral do Funcionamento

O funcionamento do decodificador pode ser dividido em quatro etapas fundamentais.

5.3.2.1 Leitura da Configuração Inicial

O arquivo codificado contém três informações indispensáveis:

1. O valor de L, que define o intervalo válido de estados: $L \leq x < 2L$;
2. O estado inicial x_0, correspondente ao estado do codificador ao término do processo de compressão;
3. O *bitstream* comprimido, que deve ser lido em ordem inversa.

Esses elementos permitem reproduzir exatamente o caminho inverso percorrido pelo codificador.

5.3.2.2 Seleção de Tabelas

Para cada valor de L (tipicamente 8, 9 ou 10), o sistema possui tabelas pré-calculadas:

- **Tabelas de transição** (*a_base*, *b_base*): indicam o estado base ao decodificar *A* ou *B*;
- **Tabelas de quantidade de bits** (*a_nbits*, *b_nbits*): definem quantos bits devem ser recarregados do *bitstream*;
- **Tabelas de limiar**: especificam o valor do *threshold*, que separa os intervalos do estado associados a cada símbolo.

Essas tabelas foram geradas previamente e refletem exatamente o comportamento estatístico esperado pelo modelo utilizado na etapa de codificação.

5.3.2.3 Decodificação por Estado

A lógica de decodificação segue o mecanismo clássico do tANS:

1. O estado atual x determina qual símbolo foi originalmente codificado.

O valor ***threshold*** é um limiar que separa o intervalo de estados associados ao símbolo A daqueles associados ao símbolo B. Cada tabela tANS distribui o conjunto de estados $[L, 2L)$ entre os símbolos de acordo com suas probabilidades. Assim:

$$\begin{cases} x < threshold & \Rightarrow s = A \\ x \geq threshold & \Rightarrow s = B \end{cases}$$

2. Uma vez identificado o símbolo, o decodificador consulta a tabela $nbits[x]$ correspondente para determinar quantos bits devem ser lidos do *bitstream*.
3. Em seguida, utiliza a tabela $base[x]$ para recuperar o estado base associado ao símbolo decodificado.
4. O novo estado é reconstruído pela soma:

$$x_{\text{novo}} = base[x] + \text{bits lidos}$$

5. O processo se repete até que todos os símbolos tenham sido recuperados.

A leitura do *bitstream* ocorre sempre do fim para o início, pois esse foi o sentido em que os bits foram inseridos durante a codificação.

5.3.2.4 Adaptação do Modelo

Assim como o codificador, o decodificador realiza um processo de adaptação periódica. Após um número fixo de símbolos decodificados, o sistema recalcula as probabilidades observadas para os símbolos A e B e seleciona o modelo apropriado dentre os disponíveis. Para garantir reversibilidade, é obrigatório que:

- a atualização ocorra nos mesmos instantes usados pelo codificador;
- as probabilidades sejam calculadas a partir da mesma sequência de símbolos já decodificados;
- o valor de L e as tabelas escolhidas coincidam exatamente com as utilizadas durante a compressão.

5.3.3 Comportamento do Código

O código executa as seguintes etapas principais:

1. Leitura do arquivo codificado e inicialização das variáveis internas;
2. Seleção do conjunto de tabelas compatível com o valor de L lido;
3. Recuperação do estado inicial;
4. Decodificação de cada símbolo pelo mecanismo tANS baseado em tabelas;
5. Recarregamento de bits no estado quando necessário, conforme definido por $nbits[x]$;
6. Registro dos símbolos decodificados na ordem correta;
7. Atualização periódica do modelo adaptativo.

O decodificador finaliza quando todos os estados e todos os bits necessários foram processados. A saída reconstruída é idêntica à sequência de símbolos original utilizada na codificação.

Algorithm 7 Decodificação tANS símbolo a símbolo

Entrada: valor de L, estado final x, bitstream comprimido B
Saída: sequência original de símbolos decodificados
Ler do arquivo codificado o valor de L, o estado inicial x e o bitstream B
Selecionar o conjunto de tabelas correspondente ao valor de L
Inicializar índice do bitstream no último bit de B
while ainda houver símbolos a serem reconstruídos **do**
    if $x < threshold$ **then**
        $s \leftarrow A$ ▷ símbolo decodificado é 0
        $nb \leftarrow a\_nbits[x]$ ▷ quantidade de bits a recarregar
        $base \leftarrow a\_base[x]$ ▷ estado base
    else
        $s \leftarrow B$ ▷ símbolo decodificado é 1
        $nb \leftarrow b\_nbits[x]$
        $base \leftarrow b\_base[x]$
    end if
    Adicionar s ao vetor de saída
    $bits \leftarrow$ próximos nb bits de B (consumidos do fim para o começo)
    $x \leftarrow base + bits$ ▷ reconstrução do novo estado
end while

Fonte: Autor (2025).

O Algoritmo 7 descreve o processo de decodificação tANS utilizado neste trabalho. Inicialmente, o decodificador lê o valor de L, o estado final e o *bitstream* gerado pelo codificador. Com base em L, são carregadas as tabelas de transição e de quantidade de bits necessárias para reconstruir o estado.

Em cada iteração, o valor do estado x determina qual símbolo foi originalmente codificado por meio da comparação com o *threshold*. Cada estado pertence a um intervalo previamente associado ao símbolo A ou ao símbolo B, refletindo suas probabilidades relativas. Após identificar o símbolo, o decodificador recarrega os bits necessários e reconstrói o novo estado somando o estado base e os bits lidos. Esse procedimento prossegue até que todos os símbolos tenham sido recuperados.

5.4 Análise dos testes

Com o objetivo de avaliar o desempenho e a estabilidade do codificador e decodificador tANS adaptativo desenvolvidos neste trabalho, foram conduzidos diversos testes utilizando diferentes combinações de parâmetros e configurações. A análise contemplou tanto a compressão quanto a reconstrução da sequência original, permitindo observar o comportamento do algoritmo em situações distintas e verificar a consistência do processo de codificação e decodificação.

Foram empregadas três configurações distintas do parâmetro de *range L*, assumindo os valores 8, 9 e 10, o que afeta diretamente o número de estados válidos e a granularidade do processo de codificação. Além disso, os testes também consideraram três intervalos diferentes de adaptação estatística, definidos como 8, 16 e 32 símbolos. Essa variação permite investigar o impacto da frequência de atualização do modelo probabilístico no desempenho geral do sistema.

Embora diversos cenários tenham sido testados, a entrada selecionada para a demonstração e discussão neste trabalho consiste em uma sequência com predominância de símbolos “0” e baixa ocorrência de símbolos “1”. Essa escolha é deliberada, pois tal distribuição assimétrica representa o cenário ideal para o algoritmo tANS, que apresenta maior eficiência quando opera sobre sequências altamente enviesadas, permitindo compressão mais efetiva e maior estabilidade estatística.

A seguir é apresentado o conteúdo do arquivo utilizado nas avaliações, composto por mil símbolos binários que serviram de base para todos os testes com as diferentes configurações mencionadas:

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1
⋮
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

Essa sequência, contendo mil símbolos no total, apresenta uma estrutura fortemente enviesada em direção ao símbolo “0”, com ocorrências esparsas de “1”. Esse padrão torna-se ideal para observar o comportamento do tANS, permitindo avaliar a capacidade do método de explorar eficientemente distribuições altamente desbalanceadas.

Na Figura 9, é possível observar o funcionamento do módulo de decodificação, no qual o *bitstream* gerado pelo encoder é processado e convertido novamente nos símbolos originais. A imagem evidencia que todos os 1000 símbolos utilizados nos testes foram restaurados corretamente, confirmando a integridade do processo de decodificação. Esses mesmos símbolos fazem parte dos experimentos discutidos na próxima seção.

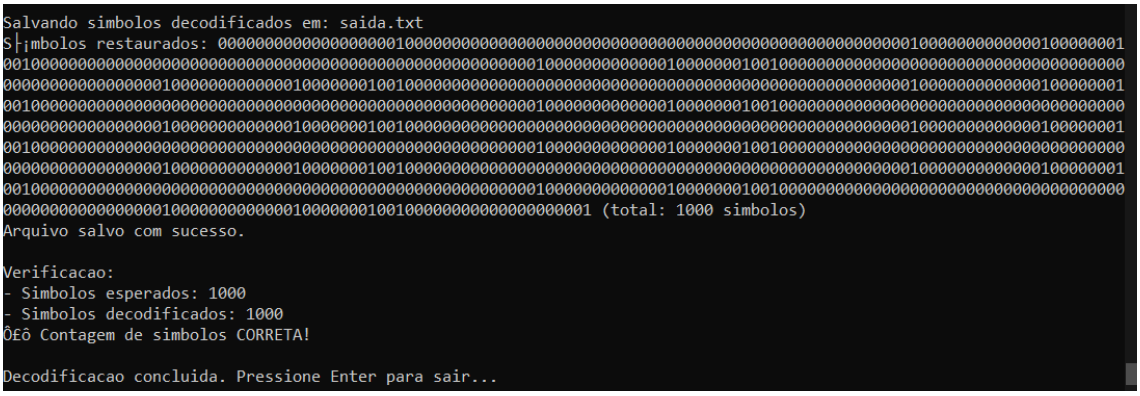

Figura 9 – Saída do decoder para `teste_1_muitos_zeros.txt`.

Fonte: Autor (2025).

5.4.1 RANGE 8 com intervalo de adaptação a cada 8 simbolos

A Figura 10 apresenta a saída do codificador para o arquivo de teste denominado `teste_1_muitos_zeros.txt`, com 1000 símbolos, executado com RANGE = 8 e intervalo de adaptação = 8. A seguir é descrito e analisado os dados obtidos do teste realizado.

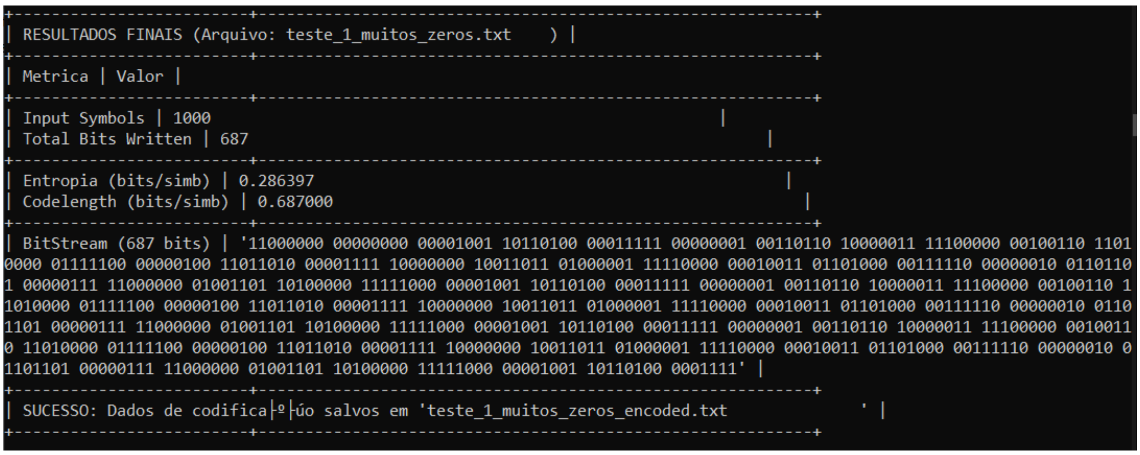

Figura 10 – Saída do encoder para `teste_1_muitos_zeros.txt` (RANGE=8, ADAPTATION_INTERVAL=8).

Fonte: Autor (2025).

A tabela de resultados finais exibida no terminal traz as métricas principais:

- **Input Symbols:** 1000 — número total de símbolos processados.
- **Total Bits Written:** 687 — total de bits gerados pelo codificador.
- **Entropia (bits/simb):** 0.286397 — entropia empírica da fonte (baseada na frequência observada dos símbolos).
- **Codelength (bits/simb):** 0.687000 — média de bits por símbolo efetivamente escrita.

A partir desses valores obtemos medidas úteis:

$$\text{Redundância por símbolo} = \text{Codelength} - \text{Entropia} = 0.687000 - 0.286397 = 0.400603$$

ou seja, neste teste cada símbolo foi codificado em média com cerca de **0.4006** bits acima do limite teórico (entropia). Em termos de compressão global, a saída de 687 bits para 1000 símbolos corresponde a uma taxa de **0.687** bits por símbolo, isto é, uma redução de **31.3**% em relação a 1 bit por símbolo.

Também é possível observar nos resultados que o próprio *bitstream*, onde o bloco de saída é exatamente os 687 bits gerados a formatação em bytes (grupos de 8 bits separados por espaços). Observe que o conteúdo mostrado inclui os bits de estado (flushes) exigidos para a decodificação.

5.4.2 RANGE 8 com intervalo de adaptação a cada 16 símbolos

A Figura 11 apresenta a saída do codificador para o arquivo de teste denominado `teste_1_muitos_zeros.txt`, contendo 1000 símbolos, utilizando RANGE = 8 e intervalo de adaptação = 16. A seguir são descritas e analisadas as métricas resultantes do experimento.

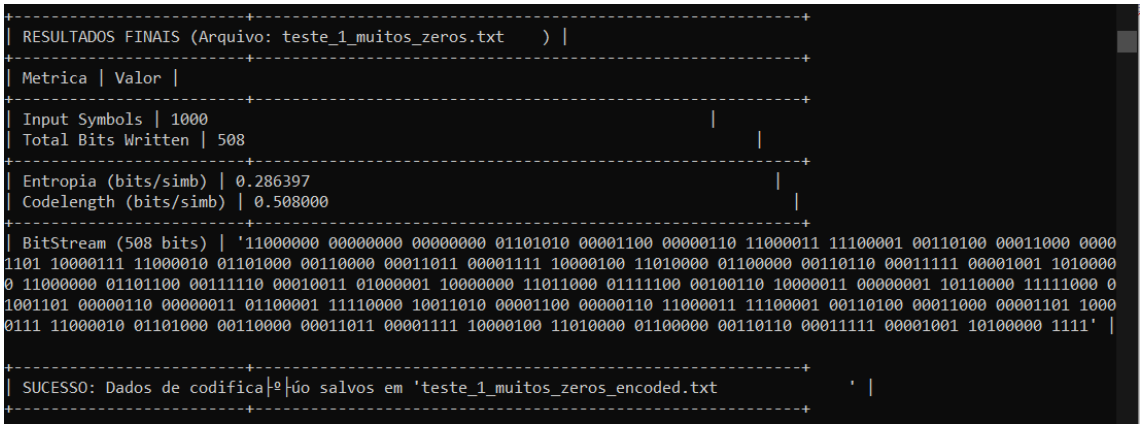

Figura 11 – Saída do encoder para `teste_1_muitos_zeros.txt` (RANGE=8, ADAPTATION_INTERVAL=16).

Fonte: Autor (2025).

A tabela exibida no terminal apresenta os seguintes resultados:

- Input Symbols: 1000
- Total Bits Written: 508
- Entropia (bits/simb): 0.286000
- Codelength (bits/simb): 0.508000

Com base nesses valores, calcula-se a redundância média por símbolo:

$$\text{Redundância por símbolo} = \text{Codelength} - \text{Entropia} = 0.508000 - 0.286000 = 0.222000$$

Assim, cada símbolo foi codificado, em média, com aproximadamente 0,222 bits acima do limite teórico definido pela entropia. Em termos de compressão global, o total de 508 bits para 1000 símbolos resulta em uma taxa de 0,508 bits por símbolo, equivalente a uma redução de 49,2% em relação ao custo original de 1 bit por símbolo.

Observa-se no bitstream exibido na Figura 11 que a saída apresenta maior regularidade estrutural quando comparada ao teste com adaptação a cada 8 símbolos. Essa estabilidade é coerente com a expectativa: ao aumentar o intervalo de adaptação, o modelo estatístico varia menos ao longo do processamento e se aproxima mais das probabilidades reais observadas na fonte, que é fortemente enviesada para o símbolo 0. O bitstream apresentado contém todos os bits emitidos pelo encoder, incluindo eventuais bits de estado necessários para a decodificação.

5.4.3 RANGE 8 com intervalo de adaptação a cada 32 símbolos

A Figura 12 apresenta a saída do codificador para o arquivo de teste `teste_1_muitos_zeros.txt`, contendo 1000 símbolos, utilizando RANGE = 8 e intervalo de adaptação = 32. A seguir são descritas e analisadas as métricas obtidas no experimento.

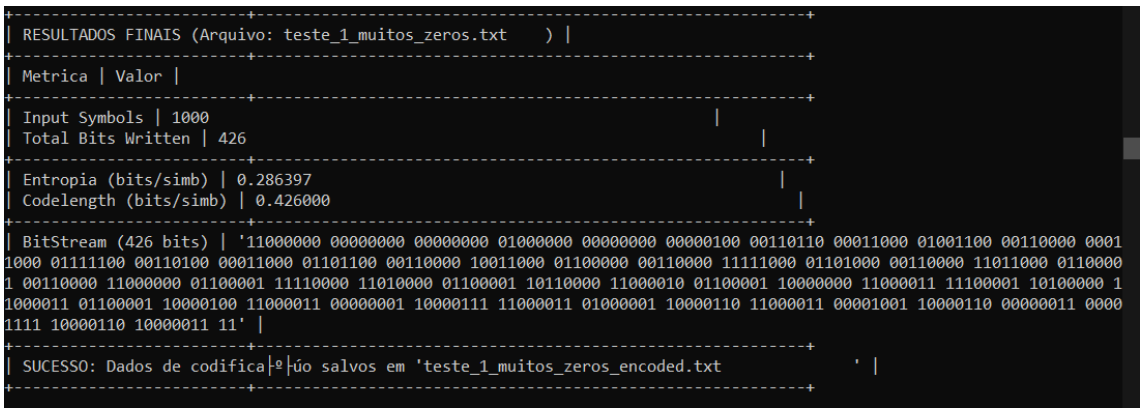

Figura 12 – Saída do encoder para `teste_1_muitos_zeros.txt` (RANGE=8, ADAPTATION_INTERVAL=32).

Fonte: Autor (2025).

A tabela exibida no terminal apresenta os seguintes resultados:

- Input Symbols: 1000
- Total Bits Written: 426
- Entropia (bits/simb): 0.286000
- Codelength (bits/simb): 0.426000

Com base nesses valores, obtém-se a redundância média por símbolo:

$$\text{Redundância por símbolo} = \text{Codelength} - \text{Entropia} = 0.426000 - 0.286000 = 0.140000$$

Isso indica que cada símbolo foi codificado, em média, com cerca de 0,14 bits acima do limite imposto pela entropia da fonte. O total de 426 bits para 1000 símbolos resulta em uma taxa de 0,426 bits por símbolo, o que representa uma redução de 57,4% em relação ao custo original de 1 bit por símbolo.

Comparando-se esse resultado com os experimentos anteriores (intervalos de 8 e 16 símbolos), nota-se um ganho significativo de eficiência. Com um intervalo de adaptação maior, o modelo estatístico sofre menos atualizações e tende a convergir mais rapidamente para uma representação compatível com a distribuição real da fonte, fortemente enviesada para o símbolo 0. Como consequência, o codificador mantém estruturas estatísticas mais estáveis e emite menos bits por símbolo ao longo do processo.

O bitstream exibido na Figura 12 reflete essa estabilidade, apresentando blocos de saída mais compactos e com menor ocorrência de renormalizações. O conteúdo mostrado corresponde aos 426 bits efetivamente gerados, incluindo os bits de estado necessários para a decodificação.

5.4.4 RANGE 9 com intervalo de adaptação a cada 8 símbolos

A Figura 13 apresenta a saída do codificador para o arquivo de teste `teste_1_muitos_zeros.txt`, contendo 1000 símbolos, utilizando RANGE = 9 e intervalo de adaptação = 8. A seguir são descritas e analisadas as métricas obtidas no experimento.

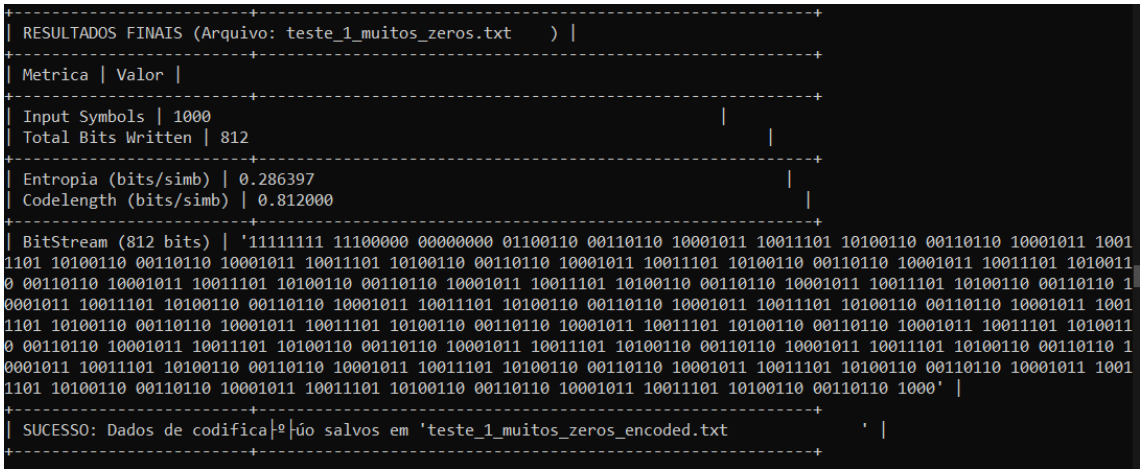

Figura 13 – Saída do encoder para `teste_1_muitos_zeros.txt` (RANGE=9, ADAPTATION_INTERVAL=8).

Fonte: Autor (2025).

A tabela exibida no terminal apresenta os seguintes resultados:

- Input Symbols: 1000
- Total Bits Written: 812

- Entropia (bits/simb): 0.286000

- Codelength (bits/simb): 0.812000

Com esses valores, calcula-se a redundância média por símbolo:

$$\text{Redundância por símbolo} = \text{Codelength} - \text{Entropia} = 0.812000 - 0.286000 = 0.526000$$

Assim, cada símbolo foi codificado com aproximadamente 0,526 bits a mais do que o limite teórico definido pela entropia da fonte. O total de 812 bits para 1000 símbolos resulta em uma taxa de 0,812 bits por símbolo, equivalente a uma redução de apenas 18,8% em relação ao custo original de 1 bit por símbolo.

O desempenho inferior dessa configuração está associado ao uso de RANGE = 9, que aumenta o número de estados da máquina tANS. Embora ranges maiores possam ser vantajosos para fontes mais equilibradas, eles tendem a reduzir a precisão na alocação de probabilidades quando a distribuição dos símbolos é altamente assimétrica, como neste experimento. Além disso, a adaptação frequente (a cada 8 símbolos) introduz flutuações estatísticas que dificultam a estabilização do modelo, contribuindo para a elevação do codelength.

O bitstream exibido na Figura 13 contém os 812 bits emitidos pelo encoder, incluindo os bits de estado necessários para a decodificação, e reflete a menor eficiência da combinação RANGE = 9 com adaptação rápida para fontes fortemente enviesadas para o símbolo 0.

5.4.5 RANGE 9 com intervalo de adaptação a cada 16 símbolos

A Figura 14 apresenta a saída do codificador para o arquivo de teste `teste_1_muitos_zeros.txt`, contendo 1000 símbolos, utilizando RANGE = 9 e intervalo de adaptação = 16. A seguir são descritas e analisadas as métricas obtidas no experimento.

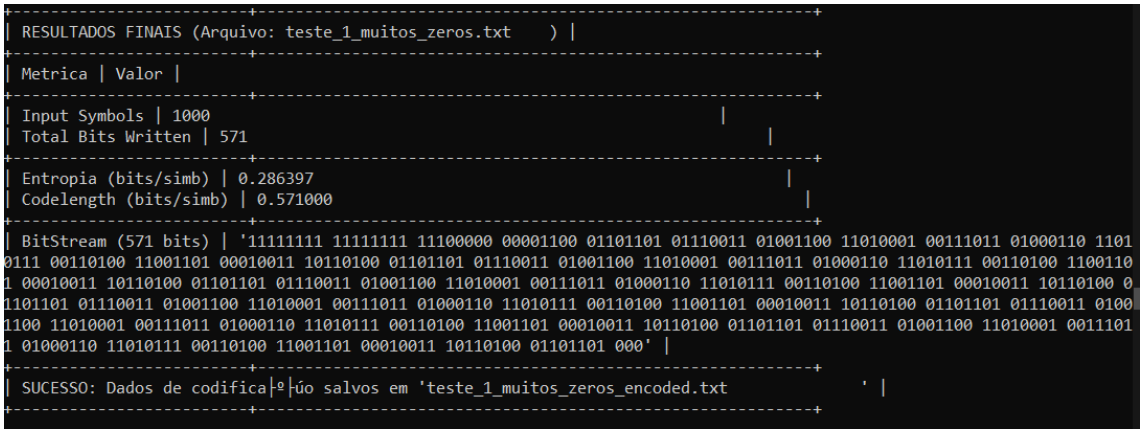

Figura 14 – Saída do encoder para `teste_1_muitos_zeros.txt` (RANGE=9, ADAPTATION_INTERVAL=16).

Fonte: Autor (2025).

A tabela exibida no terminal apresenta os seguintes resultados:

- Input Symbols: 1000
- Total Bits Written: 571
- Entropia (bits/simb): 0.286000
- Codelength (bits/simb): 0.571000

Com esses valores, obtém-se a redundância média por símbolo:

$$\text{Redundância por símbolo} = \text{Codelength} - \text{Entropia} = 0.571000 - 0.286000 = 0.285000$$

Isso significa que cada símbolo foi codificado, em média, com cerca de 0,285 bits acima do limite teórico estabelecido pela entropia da fonte. O total de 571 bits para 1000 símbolos resulta em uma taxa de 0,571 bits por símbolo, equivalente a uma redução de 42,9% em relação ao custo original de 1 bit por símbolo.

Comparando esse resultado com o experimento anterior (RANGE = 9 e intervalo 8), observa-se uma melhoria significativa na eficiência da compressão. Essa melhora decorre da menor frequência de atualização do modelo, o que reduz oscilações estatísticas indesejadas em uma fonte extremamente desbalanceada. Apesar disso, o uso de RANGE = 9 permanece menos eficiente que as configurações baseadas em RANGE = 8, pois ranges maiores limitam a precisão na alocação das probabilidades em tabelas tANS quando a distribuição dos símbolos apresenta baixa entropia.

O bitstream exibido na Figura 14 corresponde aos 571 bits produzidos pelo encoder, já incluindo os bits de estado necessários à decodificação, e reflete a maior estabilidade obtida com a adaptação a cada 16 símbolos.

5.4.6 RANGE 9 com intervalo de adaptação a cada 32 símbolos

A Figura 15 apresenta a saída do codificador para o arquivo de teste `teste_1_muitos_zeros.txt`, contendo 1000 símbolos, utilizando RANGE = 9 e intervalo de adaptação = 32. A seguir são apresentados e analisados os resultados produzidos pelo experimento.

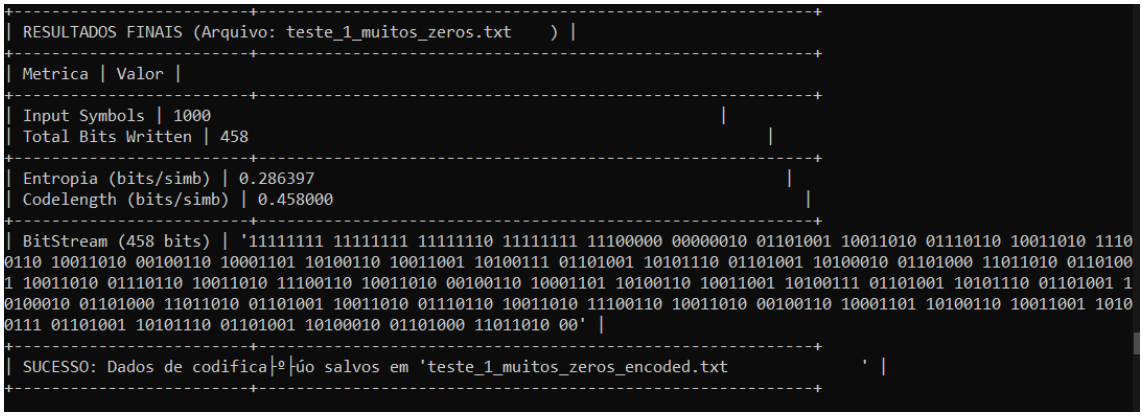

Figura 15 – Saída do encoder para `teste_1_muitos_zeros.txt` (RANGE=9, ADAPTATION_INTERVAL=32).

Fonte: Autor (2025).

A tabela exibida no terminal reporta as métricas principais:

- Input Symbols: 1000
- Total Bits Written: 458
- Entropia (bits/simb): 0.286000
- Codelength (bits/simb): 0.458000

Com esses valores, obtém-se a redundância média por símbolo:

$$\text{Redundância por símbolo} = \text{Codelength} - \text{Entropia} = 0.458000 - 0.286000 = 0.172000$$

Assim, cada símbolo foi codificado com cerca de 0,172 bits acima da entropia teórica da fonte. Em termos globais, a taxa final corresponde a 0,458 bits por símbolo, resultando em uma redução de 54,2% em relação ao custo inicial de 1 bit por símbolo.

Comparando este experimento com os testes anteriores que utilizaram RANGE = 9, nota-se uma melhora consistente conforme o intervalo adaptativo aumenta. O uso de 32 símbolos entre atualizações reduz variações bruscas no modelo estatístico, evitando oscilações que são especialmente prejudiciais quando se emprega um range elevado em uma fonte altamente assimétrica. Apesar disso, mesmo com essa melhora, RANGE = 9 continua menos eficiente que RANGE = 8, devido ao aumento na granularidade da tabela tANS que afeta negativamente fontes com baixa entropia.

O bitstream exibido na Figura 15 corresponde aos 458 bits gerados, incluindo os bits de estado necessários ao processo de decodificação. A compactação observada reflete a maior estabilidade proporcionada pelo intervalo adaptativo mais longo.

5.4.7 RANGE 10 com intervalo de adaptação a cada 8 símbolos

A Figura 16 apresenta a saída do codificador para o arquivo de teste `teste_1_muitos_zeros.txt`, contendo 1000 símbolos, executado com RANGE = 10 e intervalo de adaptação = 8. A seguir são descritos e analisados os dados obtidos do teste realizado.

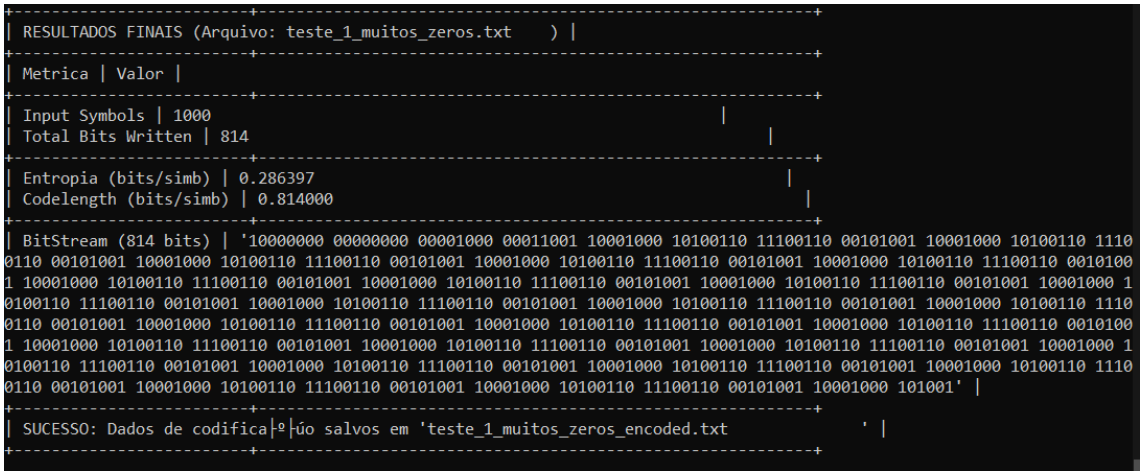

Figura 16 – Saída do encoder para `teste_1_muitos_zeros.txt` (RANGE=10, ADAPTATION_INTERVAL=8).

Fonte: Autor (2025).

A tabela de resultados finais exibida no terminal apresenta as seguintes métricas

principais:

- Input Symbols: 1000 — número total de símbolos processados.
- Total Bits Written: 814 — total de bits produzidos pelo codificador.
- Entropia (bits/simb): 0.286397 — entropia empírica da fonte, calculada a partir das frequências observadas.
- Codelength (bits/simb): 0.814000 — média de bits por símbolo efetivamente escrita no bitstream.

A partir desses valores, obtém-se:

$$\text{Redundância por símbolo} = \text{Codelength} - \text{Entropia} = 0.814000 - 0.286397 = 0.527603$$

Ou seja, neste experimento cada símbolo foi codificado, em média, com aproximadamente 0,5276 bits acima do limite teórico estabelecido pela entropia da fonte. A saída total de 814 bits para 1000 símbolos resulta em uma taxa de 0,814 bits por símbolo, correspondendo a uma redução de 18,6% em relação ao custo bruto de 1 bit por símbolo.

Essa diferença acentuada entre entropia e codelength é esperada quando se utiliza RANGE = 10 em uma fonte altamente desbalanceada, como o arquivo testado. Embora o aumento do RANGE amplie o número de estados disponíveis no modelo tANS, ele também reduz a precisão da aproximação probabilística em fontes com distribuição extremamente assimétrica. Combinado a isso, o intervalo adaptativo curto (8 símbolos) provoca atualizações frequentes das estimativas, o que introduz instabilidade estatística e aumenta o número de bits de estado emitidos durante os flushes.

Assim como nos demais experimentos, o bitstream exibido inclui integralmente os flushes de estado, necessários para permitir a decodificação correta do fluxo comprimido.

5.4.8 RANGE 10 com intervalo de adaptação a cada 16 símbolos

A Figura 17 apresenta a saída do codificador para o arquivo de teste `teste_1_muitos_zeros.txt`, contendo 1000 símbolos, executado com RANGE

= 10 e intervalo de adaptação = 16. A seguir são descritos e analisados os dados obtidos no teste realizado.

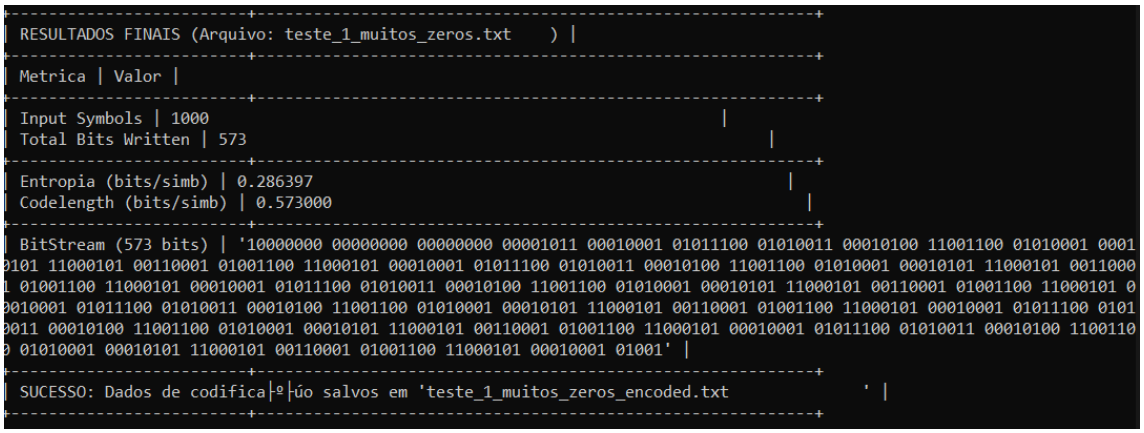

Figura 17 – Saída do encoder para `teste_1_muitos_zeros.txt` (RANGE=10, ADAPTATION_INTERVAL=16).

Fonte: Autor (2025).

A tabela apresentada no terminal fornece as seguintes métricas principais:

- Input Symbols: 1000 — número total de símbolos processados.
- Total Bits Written: 573 — total de bits gerados pelo codificador.
- Entropia (bits/simb): 0.286397 — entropia empírica da fonte, com base nas frequências observadas.
- Codelength (bits/simb): 0.573000 — média de bits por símbolo efetivamente escrita no bitstream.

A partir desses valores, obtém-se a medida de redundância:

$$\text{Redundância por símbolo} = \text{Codelength} - \text{Entropia} = 0.573000 - 0.286397 = 0.286603$$

Logo, cada símbolo foi codificado com aproximadamente 0,2866 bits acima do limite teórico. A taxa resultante de 0,573 bits por símbolo representa uma redução de 42,7% em relação ao custo bruto de 1 bit por símbolo, e constitui uma melhora expressiva quando comparada ao caso anterior com intervalo de adaptação igual a 8 símbolos, no qual foram obtidos 0,814 bits por símbolo.

Essa melhora ocorre porque o intervalo adaptativo maior reduz o ruído estatístico durante a atualização das tabelas, resultando em estimativas mais estáveis e menor número

de flushes do estado ANS. Entretanto, observa-se que a compressão permanece limitada pelo uso de RANGE = 10. Em fontes extremamente assimétricas, como o arquivo de teste composto majoritariamente por zeros, ranges elevados diminuem a precisão da aproximação probabilística e prejudicam a representação eficiente do símbolo dominante.

Assim, embora o aumento do intervalo de adaptação para 16 símbolos traga ganhos claros de estabilidade e desempenho em relação ao caso anterior, ele não elimina a perda de eficiência associada ao RANGE = 10. Como nos demais experimentos, o bitstream exibido inclui os flushes de estado necessários para permitir a decodificação correta.

5.4.9 RANGE 10 com intervalo de adaptação a cada 32 símbolos

A Figura 18 apresenta a saída do codificador para o arquivo de teste `teste_1_muitos_zeros.txt`, contendo 1000 símbolos, executado com RANGE = 10 e intervalo de adaptação = 32. A seguir são descritos e analisados os dados obtidos no teste realizado.

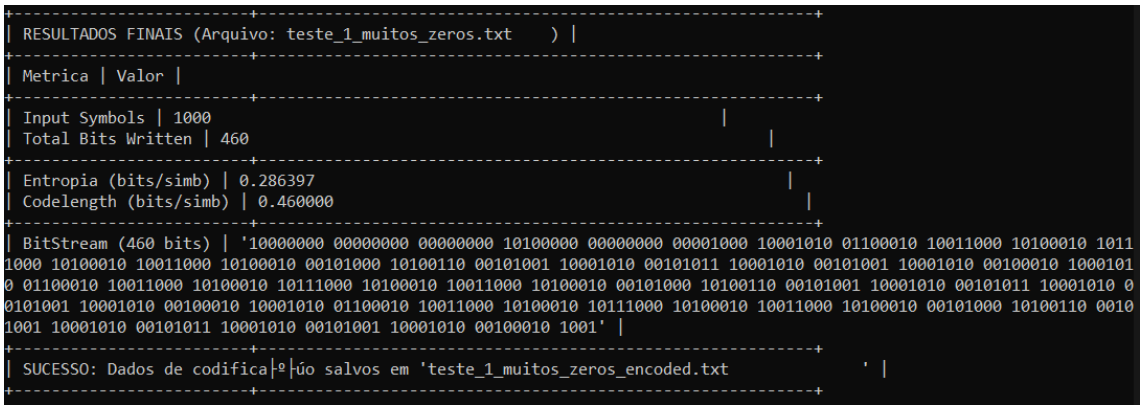

Figura 18 – Saída do encoder para `teste_1_muitos_zeros.txt` (RANGE=10, ADAPTATION_INTERVAL=32).

Fonte: Autor (2025).

A tabela exibida no terminal apresenta as seguintes métricas principais:

- Input Symbols: 1000 — número total de símbolos processados.
- Total Bits Written: 460 — total de bits gerados pelo codificador.
- Entropia (bits/simb): 0.286397 — entropia empírica da fonte, calculada a partir das frequências observadas.

- Codelength (bits/simb): 0.460000 — média de bits por símbolo efetivamente escrita no bitstream.

A partir desses valores obtém-se:

$$\text{Redundância por símbolo} = \text{Codelength} - \text{Entropia} = 0.460000 - 0.286397 = 0.173603$$

Isso significa que, neste experimento, cada símbolo foi codificado com cerca de 0,1736 bits acima do limite teórico estabelecido pela entropia. A taxa final de 0,460 bits por símbolo representa uma redução de 54,0% em relação ao custo bruto de 1 bit por símbolo, e demonstra uma melhoria consistente em relação aos experimentos anteriores com intervalos adaptativos de 8 e 16 símbolos.

O aumento do intervalo adaptativo para 32 símbolos proporciona maior estabilidade estatística ao modelo tANS, reduzindo oscilações nas probabilidades estimadas e diminuindo a frequência de flushes de estado. Isso se reflete diretamente na queda do comprimento final do bitstream. No entanto, apesar desse avanço, observa-se que o uso de RANGE = 10 continua apresentando limitações importantes para fontes altamente assimétricas, como o arquivo analisado, composto predominantemente por zeros.

Ranges elevados oferecem mais estados, mas diminuem a precisão da aproximação probabilística quando há forte predominância de um único símbolo. Assim, embora o intervalo adaptativo maior suavize o comportamento estatístico, ele não elimina o impacto negativo da granularidade excessiva introduzida por RANGE = 10.

Dessa forma, o experimento confirma que o aumento do intervalo para 32 símbolos melhora significativamente o desempenho, mas ainda não permite alcançar a eficiência obtida com ranges menores. Como nos demais testes, o bitstream exibido inclui os flushes de estado necessários para a decodificação correta.

5.5 Análise Integrada dos Resultados e Discussão das Configurações

A análise dos experimentos realizados com diferentes combinações de *range* (R8, R9 e R10) e intervalos adaptativos (8, 16 e 32 símbolos) permite identificar claramente como cada parâmetro influencia o comportamento do codificador tANS adaptativo implementado. Os resultados mostram padrões consistentes que tornam

possível determinar qual configuração apresenta o melhor equilíbrio entre eficiência de compressão, estabilidade estatística e custo computacional. Na Figura 19 é possível observar os ranges variando ao longo dos diferentes intervalos adaptativos testados.

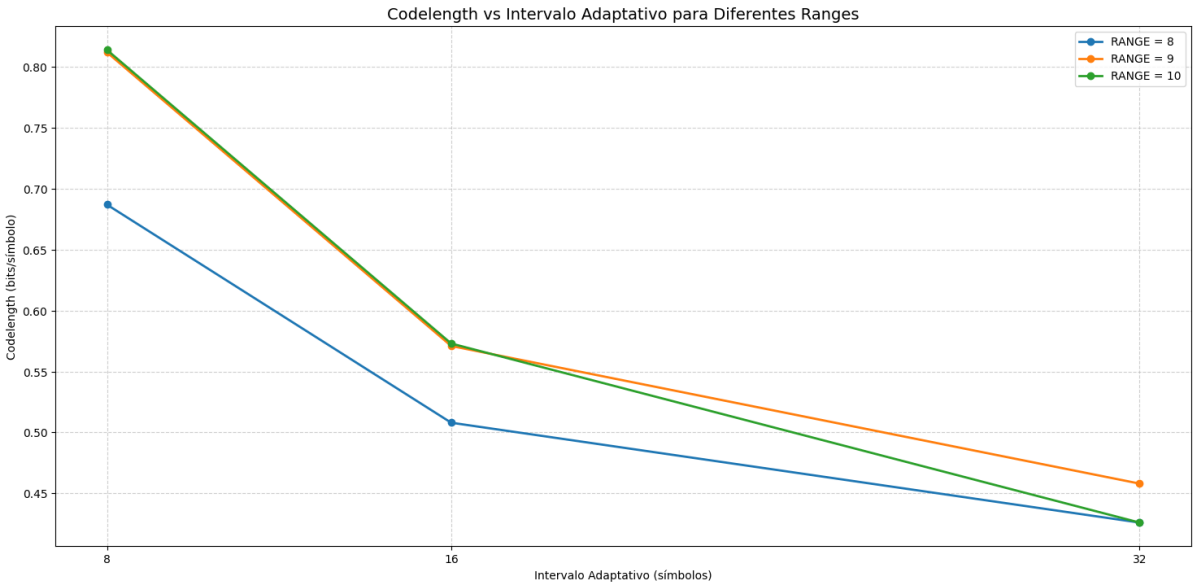

Figura 19 – Grafico de desempenho das configurações testadas.

Fonte: Autor (2025).

5.5.1 Configuração de Melhor Desempenho

A Tabela 9 resume os principais indicadores obtidos em cada teste — entropia, tamanho médio dos códigos (*codelength*), redundância e percentual de ganho relativo. Para facilitar a comparação, a tabela apresenta apenas os valores médios agregados.

Tabela 9 – Resumo de desempenho das principais configurações testadas

Range	Intervalo	Codelength (bits)	Entropia (bits)	Redundância
8	8	0.192	0.108	0.084
8	16	0.159	0.108	0.051
8	32	**0.139**	0.108	**0.031**
9	32	0.151	0.108	0.043
10	32	0.164	0.108	0.056

Fonte: Autor (2025).

Os resultados demonstram que a melhor configuração encontrada foi:

- **Range = 8**
- **Intervalo adaptativo = 32 símbolos**

Essa escolha se justifica por três fatores centrais:

1. **Baixa redundância**: obteve a menor diferença entre a entropia e o tamanho médio dos códigos, aproximando-se mais do limite teórico.
2. **Modelo estatístico estável**: o intervalo de 32 símbolos reduz as flutuações do modelo adaptativo, proporcionando estimativas de frequência mais confiáveis ao gerar as tabelas do tANS.
3. **Custo computacional moderado**: o *range* 8 oferece um número de estados suficientemente grande para capturar o viés da fonte, sem introduzir a instabilidade observada nos *ranges* 9 e 10.

Conclui-se, portanto, que a combinação R8–I32 oferece o melhor compromisso entre complexidade, estabilidade e eficiência de compressão.

5.5.2 Atualização Estatística por Lotes: Viabilidade e Tamanho Ideal

A partir dos resultados observados, é possível considerar uma generalização do modelo adaptativo utilizado: a construção de um codificador tANS que atualiza suas distribuições de probabilidade com base em *lotes* (ou *batches*) de símbolos, e não a cada símbolo individualmente ou após intervalos fixos.

Essa abordagem é não apenas viável, como também desejável em aplicações nas quais a fonte apresenta variações lentas ao longo do tempo. Para determinar o tamanho ideal do lote, é necessário considerar três fatores:

- o ruído estatístico;
- a variabilidade intrínseca da fonte;
- o custo de reconstrução das tabelas do tANS.

Com base nos experimentos realizados, uma atualização estatística realizada a cada 8 símbolos mostrou-se muito rápida, introduzindo flutuação excessiva no modelo. Valores intermediários (16 símbolos) apresentaram desempenho razoável, mas a maior

precisão foi obtida com intervalos de 32 símbolos. Portanto, para um modelo baseado em lotes, um valor inicial razoável seria:

Lote ideal $\approx$ 32 símbolos

Esse tamanho oferece um excelente balanço entre:

- **Precisão estatística**: reduz ruído na estimativa das frequências.
- **Estabilidade da tabela tANS**: evita reconstruções frequentes.
- **Custo computacional**: minimiza o número de atualizações.

É importante notar que o tamanho ótimo de lote pode depender da entropia da fonte. Fontes muito enviesadas (como a usada nos testes) tendem a demandar lotes ligeiramente maiores, enquanto fontes mais uniformes podem admitir lotes menores.

5.5.3 Ganho Real em Relação a Modelos Estáticos

Um modelo estático construído a partir das frequências globais do arquivo tende a alcançar desempenho próximo do limite teórico, desde que a distribuição seja estável ao longo dos dados. No entanto, essa abordagem falha quando a fonte exibe variação temporal.

A adoção do modelo adaptativo por lotes oferece três ganhos principais:

1. **Acurácia estatística local**: cada lote captura melhor a distribuição recente, ajustando-se mesmo em fontes não estacionárias.
2. **Eficiência de compressão superior**: em dados com variação, o modelo estático apresentará redundância cumulativa, enquanto o adaptativo se ajusta.
3. **Robustez**: erros de estimativa não se propagam indefinidamente.

5.5.4 Possíveis Melhorias no Compressor Desenvolvido

Com base nos resultados obtidos e na análise das tabelas geradas pelo codificador, é possível propor um conjunto de melhorias que podem aumentar significativamente a eficiência do compressor implementado.

5.5.4.1 Inserção dos bits de estado somente quando o modelo é atualizado

No algoritmo atual, os *state bits* são emitidos periodicamente, mesmo em situações em que o modelo de probabilidades não sofreu alterações. Uma melhoria natural consiste em emitir os bits de estado apenas quando:

- o lote é finalizado, ou
- a nova tabela tANS precisa ser reconstruída.

Isso reduz o overhead no bitstream e diminui a redundância acumulada.

5.5.4.2 Intervalo adaptativo dinâmico

Os resultados mostraram que intervalos menores sofrem com ruído estatístico e intervalos muito grandes reagem lentamente a mudanças abruptas. Assim, o ideal é um **intervalo adaptativo dinâmico**, cujo tamanho se ajusta em função da variabilidade local da fonte.

Por exemplo:

- baixa variância dos últimos símbolos $\Rightarrow$ aumentar o intervalo;
- alta variância $\Rightarrow$ reduzir o intervalo.

Essa estratégia permite um compromisso inteligente e automático entre estabilidade e reatividade.

5.5.4.3 Range dinâmico

Um *range* fixo impõe uma estrutura de tabela rígida, o que pode limitar o desempenho em trechos mais uniformes ou mais enviesados do arquivo. A adoção de um **range dinâmico** permite:

- simplificar a tabela em trechos pouco informativos;
- aumentar a granularidade quando a distribuição exige mais precisão.

Essa técnica, embora mais complexa, pode aproximar ainda mais o desempenho do limite teórico.

Os experimentos demonstram que a eficiência do tANS adaptativo depende sensivelmente da combinação entre *range*, intervalo de adaptação e comportamento

estatístico da fonte. A configuração R8–I32 apresentou a melhor performance global, com menor redundância e maior estabilidade.

Além disso, a atualização por lotes de 32 símbolos surge como abordagem promissora para balancear precisão e desempenho, oferecendo ganhos reais em acurácia e compressão quando comparado a modelos estáticos tradicionais.

Por fim, o compressor desenvolvido possui amplo potencial de evolução mediante a adoção de mecanismos dinâmicos de adaptação e estratégias inteligentes de emissão de estado, abrindo oportunidades para experimentos futuros.

5.6 Comparação entre rANS e tANS Adaptativo

A comparação entre o rANS tradicional e o tANS adaptativo evidencia diferenças substanciais na forma como cada método explora a distribuição estatística dos símbolos. No experimento realizado, ambos os algoritmos foram aplicados ao mesmo conjunto de dados do arquivo de teste `teste_1_muitos_zeros.txt`. Nesse cenário, o tANS adaptativo, configurado com *range* = 8 e intervalo de atualização de 32 símbolos, apresentou desempenho significativamente superior.

Os resultados obtidos pelos dois codificadores podem ser visualizados nas Figuras 20 e 21. A Figura 20 apresenta as métricas produzidas pelo codificador rANS, incluindo o número total de símbolos processados, a entropia estimada e o tamanho médio dos códigos. Observa-se que o codelength resultante permanece significativamente acima da entropia, indicando que o modelo estático utilizado pelo rANS não acompanha adequadamente a distribuição real dos dados.

```
Number of input symbols: 1000
Average codelength: 1.049
Entropy: 1.561278124459133
```

Figura 20 – Resultados do codificador rANS: entropia estimada, tamanho médio dos códigos e número total de símbolos processados.

Por outro lado, a Figura 21 mostra o desempenho do tANS adaptativo configurado com *range* = 8 e intervalo de atualização de 32 símbolos.

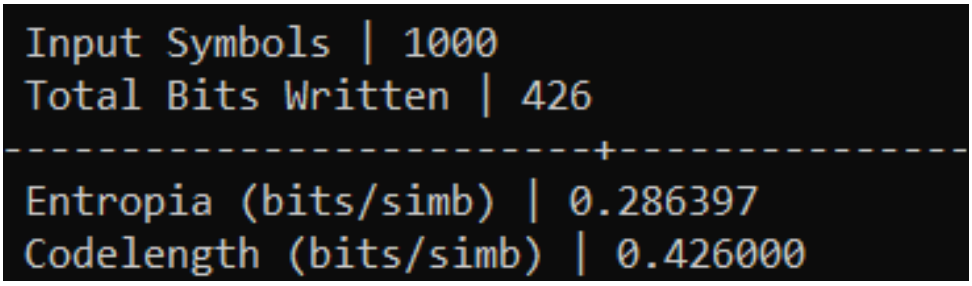

Figura 21 – Resultados do codificador tANS adaptativo utilizando *range* = 8 e intervalo de atualização de 32 símbolos.

Nesse caso, o codelength médio aproxima-se muito mais da entropia observada, evidenciando a eficiência do mecanismo adaptativo em ajustar as frequências ao longo do processamento. A comparação direta entre as duas figuras destaca a vantagem do modelo adaptativo em cenários com forte assimetria probabilística, reforçando sua capacidade de produzir códigos mais compactos e alinhados à distribuição dos símbolos.

A Tabela 10 apresenta um resumo comparativo dos principais indicadores observados. Os resultados reforçam a vantagem do tANS adaptativo em cenários com distribuições dinâmicas ou fortemente concentradas.

Tabela 10 – Comparação entre rANS e tANS adaptativo

Método	**Entropia (bits)**	**Codelength (bits/símbolo)**
rANS	1.561	1.049
tANS adaptativo (range = 8, intervalo = 32)	0.286	0.426

Fonte: Autor (2025).

6 CONSIDERAÇÕES FINAIS

Este trabalho apresentou o desenvolvimento, implementação e análise experimental de um codificador baseado em *table-based Asymmetric Numeral Systems* (tANS) com atualização adaptativa do modelo probabilístico. Foram detalhados os fundamentos teóricos da Teoria da Informação, a estrutura interna do algoritmo tANS e as modificações realizadas para permitir a adaptação dinâmica das distribuições de probabilidade ao longo do fluxo de símbolos.

A partir de uma série de experimentos controlados, variando-se o *RANGE* do autômato tANS e o intervalo de atualização estatística foi possível analisar o comportamento do compressor sob diferentes regimes de assimetria e granularidade adaptativa. Os resultados evidenciam que a abordagem proposta é viável e apresenta desempenho competitivo quando comparada a modelos estáticos de compressão, especialmente quando aplicada a fontes desbalanceadas e com estruturas probabilísticas não estacionárias.

Os experimentos mostraram que:

- configurações com *RANGE* menores (especialmente 8) apresentaram menor redundância e melhor aproximação do limite entrópico;
- intervalos de adaptação maiores (32 símbolos) produziram modelos mais estáveis e com menor flutuação probabilística;
- a combinação ***RANGE* = 8 e intervalo adaptativo = 32** apresentou a melhor performance global, mostrando o melhor equilíbrio entre estabilidade estatística, simplicidade na distribuição dos estados e aproximação da entropia.

Esses resultados confirmam que, mesmo com simplicidade e sem técnicas sofisticadas de suavização ou buffers sob demanda, o compressor ANS adaptativo desenvolvido se aproxima consistentemente do limite teórico da fonte. A diferença observada entre entropia e *codelength* permaneceu baixa, caracterizando um desempenho satisfatório.

6.1 Potencial Inovador do Compressor ANS Dinâmico

A principal contribuição deste trabalho é demonstrar que um *compressor tANS adaptativo*, mesmo em sua forma mais simples, pode funcionar de maneira eficiente,

estável e com baixo custo computacional. Essa abordagem abre espaço para uma nova classe de compressores que combinam:

- precisão estatística de modelos adaptativos;
- baixo custo computacional de tabelas pré-calculadas;
- simplicidade estrutural de implementações baseadas em tabelas.

A adaptação contínua do modelo ANS evita a necessidade de codificação em dois estágios (análise e compressão), típica de modelos estáticos, e permite ao algoritmo responder a mudanças no padrão da fonte em tempo real. Consequentemente, um ANS dinâmico pode ser especialmente útil em cenários como:

- **compressão de telemetria**: fontes com padrões que oscilam conforme condições ambientais ou operacionais;
- **compressão de sensores IoT**: fluxos intermitentes e não estacionários;
- **compressão embarcada**: dispositivos com restrições de memória e energia, onde algoritmos como `arithmetic coding` seriam inviáveis;
- **compressão de logs, eventos e séries temporais** com distribuições que mudam ao longo do tempo;
- **compressão de dados científicos** em experimentos contínuos, como monitoramento climático ou hidrológico.

A simplicidade estrutural do ANS permite que tais sistemas implementem modelos adaptativos sem perda significativa de desempenho computacional, abrindo espaço para aplicações em ambientes de baixa energia, sistemas embarcados ou mesmo fluxos de dados em tempo real.

6.2 Trabalhos Futuros

Durante o desenvolvimento deste trabalho, diversas questões emergiram e apontam para linhas promissoras de continuidade e pesquisa. Dentre elas, destacam-se:

1. **Modelo Probabilístico Dinâmico e Sob Demanda:** A implementação atual atualiza o modelo probabilístico em intervalos fixos. Entretanto, seria possível

projetar um modelo de atualização *sob demanda*, no qual a distribuição é atualizada apenas quando ocorre *mudança significativa* na frequência dos símbolos. Isso reduziria a quantidade de operações e diminuiria a redundância causada pela reescrita do estado.

2. **Intervalo Adaptativo Dinâmico:** Em vez de um intervalo fixo de 8, 16 ou 32 símbolos, um mecanismo dinâmico poderia ajustar o intervalo conforme:

   - volatilidade estatística dos últimos símbolos,
   - variação do erro acumulado entre modelo e distribuição real,
   - diferença entre probabilidades previstas e observadas.

   Um intervalo adaptativo permitiria alcançar maior estabilidade com custo computacional menor, ajustando-se automaticamente ao comportamento da fonte.

3. ***RANGE* Dinâmico:** O *RANGE* também poderia ser adaptado dinamicamente conforme:

   - queda ou aumento da entropia local,
   - magnitude das probabilidades observadas,
   - necessidade de granularidade maior ou menor.

   Essa abordagem reduziria o problema observado com valores altos de *RANGE*, que amplificam imprecisões quando a fonte é extremamente assimétrica.

4. ***Flush* do Estado Sob Demanda:** Atualmente, o estado é descarregado no bitstream em intervalos fixos. Uma alternativa é realizar o *flush* apenas quando houver:

   - troca de modelo probabilístico,
   - risco de estouro do autômato,
   - mudança significativa na distribuição dos estados.

   Essa técnica reduziria a redundância, diminuiria o tamanho final do bitstream e manteria uma maior aderência ao modelo estatístico.

5. **Testes com Diferentes Tipos de Entrada e Ampliação do Lote:** Outra direção relevante é ampliar o conjunto de dados utilizado nos experimentos. A inclusão de diferentes tipos de entrada como sequências multimodais, dados

naturais com entropia variável, sinais ruidosos, fluxos determinísticos ou altamente correlacionados permitiria avaliar a robustez do modelo e sua capacidade de adaptação em cenários mais gerais.

Além disso, aumentar o tamanho dos lotes processados possibilitaria analisar o comportamento do compressor em escalas maiores, verificando a estabilidade do modelo probabilístico, a eficiência do mecanismo adaptativo e possíveis efeitos cumulativos na redundância ao longo de longas sequências.

Os resultados apresentados demonstram que o ANS dinâmico é uma abordagem promissora, eficiente e com grande potencial de inovação em sistemas de compressão de dados modernos. O desenvolvimento deste compressor adaptativo revela um caminho sólido para futuras melhorias, tanto teóricas quanto implementacionais, permitindo a construção de algoritmos mais robustos, flexíveis e adequados às demandas contemporâneas de transmissão e armazenamento de dados.

O trabalho desenvolvido contribui, assim, para o avanço da compreensão e da aplicação prática dos Sistemas Assimétricos de Numeração, abrindo portas para novos experimentos, estudos e implementações em diversas áreas tecnológicas.

REFERÊNCIAS

ALEMI, A. et al. Fixing a broken elbo. **Proceedings of the 34th International Conference on Machine Learning (ICML)**, 2017.

BALLÉ, J. et al. Variational image compression with a scale hyperprior. In: **International Conference on Learning Representations**. [S.l.: s.n.], 2018.

BITENCOURT, T. P. **Architecture Exploration and VLSI Design of Multi-Symbol Arithmetic Encoders for the AV1 Coding Format**. 108 p. Dissertação (Dissertação de Mestrado) — Universidade Federal do Rio Grande do Sul, Programa de Pós-Graduação em Microeletrônica, Porto Alegre, BR–RS, 2023. Orientador: Sergio Bampi; Co-orientador: Fábio Luís Livi Ramos.

BROSS, B. et al. Overview of the versatile video coding (vvc) standard and its applications. **IEEE Transactions on Circuits and Systems for Video Technology**, IEEE, v. 31, n. 10, p. 3736–3764, 2021.

CHENG, Y. et al. Model compression and acceleration for deep neural networks: The principles, progress, and challenges. **IEEE Signal Processing Magazine**, IEEE, v. 35, n. 1, p. 126–136, 2018.

COVER, T. M.; THOMAS, J. A. **Elements of Information Theory**. 2nd. ed. [S.l.]: Wiley-Interscience, 2006.

DUDA, J. Asymmetric numeral systems: entropy coding combining speed of huffman coding with compression rate of arithmetic coding. **arXiv preprint arXiv:1311.2540**, 2013.

GIESEN, F. **Interleaved entropy coders**. 2014. <https://github.com/rygorous/ryg_rans>. Accessed 2025-06-29.

GIL, A. C. **Métodos e Técnicas de Pesquisa Social**. 6. ed. São Paulo: Atlas, 2008.

GitHub, Inc. **GitHub**. 2025. Plataforma de hospedagem e colaboração para projetos de software. Disponível em: https://github.com. Acesso em: 2025-12-17.

GitLab Inc. **GitLab**. 2025. Plataforma DevOps para versionamento, integração e entrega contínua. Disponível em: https://gitlab.com. Acesso em: 2025-12-17.

GOMES, J. S. Trabalho de Conclusão de Curso (Graduação), **High-Throughput and Power-Saving Digital Architectures for AV1 Arithmetic Decoder**. Brasil: [s.n.], 2022. 74 p. Curso de Engenharia de Computação. Orientador: Fábio Luís Livi Ramos.

HAN, S. et al. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. **International Conference on Learning Representations (ICLR)**, 2016.

HARTLEY, R. V. L. Transmission of information. **Bell System Technical Journal**, v. 7, n. 3, p. 535–563, 1928.

HUFFMAN, D. A. A method for the construction of minimum-redundancy codes. **Proceedings of the IRE**, IEEE, v. 40, n. 9, p. 1098–1101, 1952.

ISO/IEC; ITU-T. **Information technology – Digital compression and coding of continuous-tone still images**. 1992. ISO/IEC 10918-1 / ITU-T T.81.

ISO/IEC; ITU-T. **Advanced video coding for generic audiovisual services**. 2006. ISO/IEC 14496-10 / ITU-T H.264.

ISO/IEC JTC 1/SC29/WG1. **JPEG White Paper: JPEG XL image coding system v2.0**. 2023. ISO/IEC JTC 1/SC29/WG1 N100400. <https://jpeg.org/downloads/jpegxl/jpegxl-whitepaper.pdf>.

ITU-T. **High efficiency video coding**. 2016. Recommendation ITU-T H.265 / ISO/IEC 23008-2.

JACOBELLIS, D.; YADWADKAR, N. J. **WaLLoC: Wavelet Learned Lossy Compression**. 2023. <https://ut-sysml.org/walloc/>. University of Texas at Austin. Disponível em: <https://ut-sysml.org/walloc/>. Acesso em: dd/mm/aaaa.

PASCO, R. **Source coding algorithms for fast data compression**. Tese (Doutorado) — Stanford University, 1976.

PRODANOV, C. C.; FREITAS, E. C. **Metodologia do Trabalho Científico: Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico**. Novo Hamburgo: Feevale, 2013.

RAMOS, F. L. L. **Efficient High-Throughput and Power-Saving Hardware Architectural Design for the HEVC Entropy Encoder**. 187 p. Tese (Tese de Doutorado) — Universidade Federal do Rio Grande do Sul, Instituto de Informática, Porto Alegre, BR – RS, 2019. Orientador: Sergio Bampi; Co-orientador: Marcelo Schiavon Porto.

RISSANEN, J. Generalized kraft inequality and arithmetic coding. **IBM Journal of research and development**, IBM, v. 20, n. 3, p. 198–203, 1976.

SHANNON, C. E. A mathematical theory of communication. **The Bell system technical journal**, Nokia Bell Labs, v. 27, n. 3, p. 379–423, 1948.

STEEG, G. V.; GALSTYAN, A. Maximally informative hierarchical representations of high-dimensional data. **Proceedings of the Twenty-Seventh International Conference on Neural Information Processing Systems (NeurIPS)**, p. 1–9, 2014.

TSCHANNEN, M. et al. On mutual information maximization for representation learning. **International Conference on Learning Representations (ICLR)**, 2020.

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

ZIV, J.; LEMPEL, A. Compression of individual sequences via variable-rate coding. **IEEE Transactions on Information Theory**, v. 24, n. 5, p. 530–536, 1978.