Pontifícia Universidade Católica do Paraná
Programa de Pós-Graduação em Informática

Compressão de sequências genômicas baseado em modelo preditivo sem perda de informação

JULIANO VIEIRA MARTINS

Orientador

Edson Emílio Scalabrin

Co-orientador

Roberto Hirochi Herai

Pontifícia Universidade Católica do Paraná

Curitiba
2024

Pontifícia Universidade Católica do Paraná
Programa de Pós-Graduação em Informática

Compressão de sequências genômicas baseado em modelo preditivo sem perda de informação

JULIANO VIEIRA MARTINS

Tese apresentada ao Programa de Pós-Graduação em Informática como requisito parcial para obtenção do título de Doutor em Informática.

Campo de Concentração: Ciência da Computação

Orientador: Edson Emílio Scalabrin

Co-orientador: Roberto Hirochi Herai

Curitiba
2024

Dados da Catalogação na Publicação
Pontifícia Universidade Católica do Paraná
Sistema Integrado de Bibliotecas – SIBI/PUCPR
Biblioteca Central
Luci Eduarda Wielganczuk – CRB 9/1118

Martins, Juliano Vieira
M386c 2024 Compressão de sequências genômicas baseado em modelo preditivo sem perda de informação / Juliano Vieira Martins ; orientador: Edson Emílio Scalabrin ; co-orientador: Roberto Hirochi Herai. – 2024.
176 f. : il. ; 30 cm

Tese (doutorado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2024
Bibliografia: f. 131-146

1. Informática. 2. Compressão de dados (Computação). 3. Controle preditivo. 4. Protocolo de comparação. 5. Genoma. I. Scalabrin, Edson Emílio. II. Herai, Roberto Hirochi. II. Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática. III. Título.

CDD 20. ed. – 004

Pontifícia Universidade Católica do Paraná
Escola Politécnica
Programa de Pós-Graduação em Informática

Curitiba, 08 de novembro de 2024.

93-2024

DECLARAÇÃO

Declaro para os devidos fins, que **JULIANO VIEIRA MARTINS** defendeu a tese de Doutorado intitulada "**Compressão de sequências genômicas baseado em modelo preditivo sem perda de informação"**, na área de concentração Ciência da Computação no dia 10 de junho de 2024, no qual foi aprovado.

Declaro ainda, que foram feitas todas as alterações solicitadas pela Banca Examinadora, cumprindo todas as normas de formatação definidas pelo Programa.

Por ser verdade firmo a presente declaração.

Documento assinado digitalmente
gov.br
EMERSON CABRERA PARAISO
Data: 18/11/2024 22:09:14-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Emerson Cabrera Paraiso
Coordenador do Programa de Pós-Graduação em Informática

Dedico essa pesquisa à Deus que tem me sustentado até aqui, que foi o autor e o consumador da minha fé nessa caminhada.

"16 Porque Deus amou o mundo de tal maneira que deu o seu Filho unigênito, para que todo aquele que nele crê não pereça, mas tenha a vida eterna. 17 Porque Deus enviou o seu Filho ao mundo, não para que condenasse o mundo, mas para que o mundo fosse salvo por ele. 18 Quem crê nele não é condenado; mas quem não crê já está condenado, porquanto não crê no nome do unigênito Filho de Deus. 19 E a condenação é esta: Que a luz veio ao mundo, e os homens amaram mais as trevas do que a luz, porque as suas obras eram más. 20 Porque todo aquele que faz o mal odeia a luz, e não vem para a luz, para que as suas obras não sejam reprovadas. 21 Mas quem pratica a verdade vem para a luz, a fim de que as suas obras sejam manifestas, porque são feitas em Deus."

— João 3:16-21 (ACF)

Resumo

A recente redução no custo e no tempo necessários para sequenciar e montar genomas completos culminou em uma maior demanda por armazenamento de dados. Como consequência, várias estratégias para a compressão de dados biológicos sequenciados foram desenvolvidas, abrangendo desde métodos convencionais até abordagens inovadoras específicas para dados genômicos. Dentre essas abordagens, ferramentas de compressão vertical implementam estratégias que consideram o alto nível de similaridade entre múltiplas sequências genômicas montadas para melhorar os resultados de compressão. Neste trabalho, constituiu-se uma ferramenta de compressão vertical de sequências genômicas, sem perda de informação, baseada em um modelo de predição do próximo símbolo. Adicionalmente, para lidar com as informações inerentes aos acertos e erros do modelo, uma estratégia de representação binária denominada *Bitstring as Codebook (BAC)* foi definida e armazenada diretamente no Codebook. A técnica de codificação BAC permite diminuir o tamanho da representação binária de erros e acertos de predição. A ferramenta em questão foi avaliada experimentalmente usando uma rede neural convolucional em conjunto com a arquitetura em memória e duas variações desta. As variações são: a) uma rede de memória de curto e longo prazo (LSTM); b) uma rede neural bidirecional de memória de curto e longo prazo (biLSTM) com mecanismo de atenção com contexto (*Attention with Context*); c) uma abordagem de rede que incorpora duas camadas de biLSTM. Os resultados experimentais mostraram uma acurácia global de 99,75% para LSTM, 99,69% para a biLSTM com duas camadas e 99,55% para biLSTM (*Attention with Context*). Além disso, a *Economia Média de Espaço* alcançada foi de 97,51% para o experimento *CNNLstmBAC*, 97,49% para *CNNDoubleBiLstmBAC* e 97,25% para *CNNBiLstmAttBAC*, com desvios padrão de 0,37%, 0,37% e 0,35%, respectivamente. Os resultados mostraram que a combinação do modelo preditivo *CnnLstm* e a estratégia de codificação BAC são o conjunto mais promissor para compressão de sequências genômicas mitocondriais. Como proposta secundária, foi desenvolvido um protocolo padronizado para a comparação de desempenho das ferramentas especializadas de compressão de sequências genômicas. Este protocolo inclui métricas de avaliação fornecendo assim uma base de comparação justa e objetiva entre diferentes abordagens de compressão.

Palavras-chave: Compressão de dados, Genoma, Sem perda de informação, Vertical, Modelo preditivo, Protocolo de comparação

Abstract

The recent reduced cost and time required to sequence and assemble complete genomes has increased the demand for data storage. Consequently, various strategies for the compression of sequenced biological data have been developed, ranging from conventional methods to innovative approaches specific to genomic data. Among these approaches, vertical compression tools implement strategies that consider the high level of similarity between multiple assembled genomic sequences to improve compression outcomes. This work proposes a lossless vertical compression tool based on a next-symbol prediction model for genomic sequences. Additionally, a binary representation strategy named *Bits-tring as Codebook (BAC)* was defined and stored directly in the codebook to handle the inherent information of the model's hits and misses. The BAC encoding technique can reduce the binary representation size of prediction hits and misses. The proposed tool was experimentally evaluated using a convolutional neural network (CNN) in conjunction with the *Long Short Term Memory (LSTM)* architecture and two other variations thereof. The networks are a) a long short-term memory network (LSTM); b) a bidirectional long short-term memory network (biLSTM) with an *Attention with Context* mechanism; c) an approach incorporating two layers of biLSTM. The experimental results demonstrated an overall accuracy of 99.75% for LSTM, 99.69% for the two-layer biLSTM, and 99.55% for the biLSTM (*Attention with Context*). Furthermore, the *Average Space Saving* achieved was 97.51% for the *CNNLstmBAC* experiment, 97.49% for *CNNDoubleBiLstmBAC*, and 97.25% for *CNNBiLstmAttBAC*, with standard deviations of 0.37%, 0.37%, and 0.35%, respectively. The results demonstrated that the combination of the *CnnLstm* predictive model and the BAC encoding strategy proved to be the most promising set for the compression of genomic sequences. As a secondary proposal, a standardized protocol for the performance comparison of specialized tools for genomic sequence compression was developed. This protocol includes evaluation metrics, thus providing a fair and objective comparison basis among different compression approaches.

Key-words: Data compression, Genome, Lossless, Vertical, Predictive model, Comparison protocol

Agradecimentos

A Deus, por iluminar meu caminho nesta jornada e possibilitar minha chegada até aqui.

À minha esposa, Silvia, com quem amo compartilhar a vida. Sua presença tem sido um porto seguro. Agradeço pelo afeto, paciência e por sua incrível capacidade de acreditar em mim, trazendo paz aos desafios cotidianos.

À minha filha, Aline, e ao meu filho, Juliano, que com muito carinho e apoio, acreditaram na minha capacidade de alcançar esta etapa da minha vida.

À minha família, minha mãe e irmãos, sempre fontes de incentivo.

À Pontifícia Universidade Católica do Paraná (PUCPR) e ao Programa de Pós-Graduação em Informática Aplicada (PPGIa), pelo suporte acadêmico e estrutural oferecido durante minha jornada.

Ao meu orientador, Edson Emílio Scalabrin, por ser mais que um mestre, um verdadeiro amigo nesta caminhada. Além de acreditar em meu trabalho, proporcionou-me a liberdade necessária para pesquisar. Compartilhando expectativas e conduzindo-me a reflexões mais profundas, enriqueceu significativamente minha pesquisa. Expresso minha especial admiração e gratidão.

Ao professor Roberto Hirochi Herai, por dedicar seu valioso tempo para me orientar neste estudo. Meus sinceros agradecimentos.

Aos professores Jean Paul Barddal (Ciência de Dados), Júlio César Nievola (Mineração de Dados), Alceu de Souza Britto Jr. (Aprendizagem de Máquina), pelas disciplinas ministradas que contribuíram enormemente para meu desenvolvimento acadêmico durante o doutorado. Agradeço imensamente.

À equipe da Secretaria Acadêmica, suporte e recepção, pela eficácia, dedicação e cordialidade. Muito obrigado.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Lista de Acrônimos

NGS Next-Generation Sequencing

BAM Binary Alignment Map

SAM Sequence Alignment Map

VCF Variant Call Format

GWAS Genome-Wide Association Studies

FNN Feedforward Neural Network

CNN Convolutional Neural Network

RNN Recurrent Neural Network

LSTM Long Short-Term Memory

biLSTM Bi-directional Long Short-Term Memory

GRU Gated Recurrent Units

MTS Mapped Target Sequence

LPF Longest Previous Factor

SOM Second Order Mapping

LISS Longest Increasing Sub-Sequence

DNA Deoxyribonucleic Acid

RNA Ribonucleic Acid

ASCII American Standard Code for Information Interchange

HTS High-Throughput Sequencing

IUPAC International Union of Pure and Applied Chemistry

NCBI National Center for Biotechnology Information

RLE Run-Length Encoding

1

Introdução

O advento do Sequenciamento de Próxima Geração *Next-Generation Sequencing (NGS)* (SCHUSTER, 2007; REUTER; SPACEK; SNYDER, 2015) revolucionou a capacidade de sequenciar genomas de diversos organismos em uma escala sem precedentes. Esta capacidade está aumentando exponencialmente, com uma taxa de duplicação estimada a cada sete meses (STEPHENS et al., 2015). Projeções indicam que, em um futuro próximo, a capacidade global de sequenciamento poderá alcançar aproximadamente um bilhão de indivíduos anualmente (STEPHENS et al., 2015). A rápida evolução da biotecnologia, particularmente do NGS, tem transformado as ciências da vida em um negócio de "grandes volumes de dados", onde o desafio de gerenciar, armazenar e analisar dados se tornou um obstáculo significativo (PAPAGEORGIOU et al., 2018). Diante deste cenário, os custos associados ao armazenamento de dados genômicos emergem como um desafio significativo, muitas vezes ultrapassando os próprios custos de sequenciamento (DEOROWICZ; GRABOWSKI, 2013). Portanto, a ampliação do espaço de armazenamento, embora uma opção (WETTERSTRAND, 2022; KAHN, 2011), não é suficiente para abordar esta questão de forma eficaz (FRITZ et al., 2011; HAYDEN, 2015). Nesse contexto, estratégias de compressão de dados surgem como uma alternativa altamente eficiente para mitigar tais desafios de armazenamento.

> *"A compressão de dados é o processo de conversão de um fluxo de dados de entrada (o fluxo de origem ou os dados brutos originais) em outro fluxo de dados (a saída, o fluxo de bits ou o fluxo comprimido) que tem um*

tamanho menor. Um fluxo é um arquivo ou um buffer na memória."

— D. Salomon (SALOMON, 2007), tradução nossa.

A técnica de compressão de dados é uma abordagem de otimização fundamental que pode ser aplicada a múltiplos tipos de dados biológicos. Tais dados são frequentemente armazenados em variados formatos de arquivo, como é o caso do *FASTA/FASTQ* para leituras sequenciadas (COCK et al., 2009), Binary Alignment Map (BAM) e Sequence Alignment Map (SAM) para dados de alinhamento, e Variant Call Format (VCF) para informações de *Haplótipos* ou variações sequenciais (FAIRLEY et al., 2019). Importa salientar que as ferramentas de compressão especificamente desenvolvidas para a compressão de dados de Sequenciamento de Nova Geração (NGS) em formato *FASTQ*, bem como para o alinhamento de dados em arquivos BAM/SAM, foram objeto de uma avaliação por parte do grupo MPEG HTS (NUMANAGIć et al., 2016).

Arquivos VCF (DANECEK et al., 2011) ou estruturas baseadas em grafos são comumente usados para representar variações genéticas entre diferentes sequências (incluindo variações de haplótipos). Para essas variações, leituras sequenciadas são mapeadas para um genoma de referência, ou são usadas para uma montagem *De Novo* para criar uma sequência de consenso que é então mapeada para uma sequência de referência (WU et al., 2017). O uso mais comum de arquivos VCF são para estudos haplotípicos em Genome-Wide Association Studies (GWAS), como o projeto "1000 Genomes Project" (CONSORTIUM et al., 2010), UK10K (CONSORTIUM et al., 2015), GoNL (BOOMSMA et al., 2014) ou HRC (IGLESIAS et al., 2017). Dados de variação de haplótipos, armazenados em arquivos VCF, podem ser comprimidos por alguma ferramenta como GTC (DANEK; DEOROWICZ, 2017), BGT (LI, 2016), SeqArray (ZHENG et al., 2017) e PBWT (DURBIN, 2014) ou, quando armazenados em um formato baseado em grafo por ferramentas gPBWT (NOVAK; GARRISON; PATEN, 2016) e BFT (HOLLEY; WITTLER; STOYE, 2016).

Outro tipo de formato de arquivo é o *FASTA*, comumente usado para representar uma sequência parcialmente ou totalmente montada do genoma de um organismo (ZHU et al., 2015). Uma sequência do genoma pode ser obtida por *Resequenciamento* ou por uma montagem baseada no método *De Novo*. No processo de *Resequenciamento*, leituras sequenciadas são mapeadas para o genoma de referência do organismo correspondente. No processo de montagem baseado

no método *De Novo*, a reconstrução da sequência é uma leitura sequenciada com base na sobreposição, criando *Contig*s e *Scaffold*s (NAGARAJAN; POP, 2013). Um *Contig* representa um comprimento contíguo de sequência genômica na qual a ordem das bases nitrogenadas é conhecida com um alto nível de confiança. *Scaffold* representa a porção da sequência genômica composta pelos *Contig*s e *Gap*s. Nos arquivos baseados em formato *FASTA*, cada *Gap* é representado por símbolos *N*.

Um genoma montado baseado no método *De Novo*, em comparação com outro método baseado em *Resequenciamento*, torna possível identificar variações estruturais complexas ou grandes regiões repetitivas (NAGARAJAN; POP, 2013) para criar um mapa completo de variações genéticas (CHAISSON; WILSON; EICHLER, 2015). Por esta razão, abordagens baseadas no método *De Novo* estão sendo amplamente utilizadas para a identificação correta de grandes variações da sequência.

Considerando a importância das sequências montadas por *De Novo*, normalmente armazenadas no formato *FASTA*, várias ferramentas de compressão foram propostas. Conforme definido pelos autores do Biocompress (GRUMBACH; TAHI, 1993), a primeira ferramenta de compressão de sequência genômica, o processo de compressão poderia ser realizado por meio de duas abordagens, horizontal e vertical. No modo horizontal, cada sequência genômica é comprimida usando sua informação autocontida como espaço de busca (GRUMBACH; TAHI, 1993). No modo vertical, também conhecido como compressão *delta* ou *diferencial*, o conteúdo de uma ou mais sequências pode ser usado como um espaço de busca para detectar segmentos repetitivos compartilhados. Embora para alguns casos a nomenclatura utilizada para compressão de dados genômicos considere o modo vertical equivalente ao modo referencial, o segundo modo corresponde a um subtipo do primeiro (GIANCARLO; SCATURRO; UTRO, 2012; GIANCARLO; ROMBO; UTRO, 2014). As ferramentas de compressão vertical são projetadas para genomas de uma mesma espécie, pois possuem alta similaridade genômica, como entre os seres humanos que compartilham 99,5% de todo o seu material genético (LEVY et al., 2007). Assim, seria possível armazenar apenas as diferenças de 0,5% (GIANCARLO; ROMBO; UTRO, 2014).

Avaliações anteriores foram publicadas (cf. Tabela 1.1), mas cobriram diferentes tópicos, como áreas essenciais de bioinformática nas quais as técnicas de compressão poderiam ser aplicadas (GIANCARLO; SCATURRO; UTRO, 2012; GIANCARLO; SCATURRO; UTRO, 2009; NALBANTOGLU; RUSSELL;

SAYOOD, 2010) ou abordando a compressão de dados NGS (BHATTACHARYYA; BANDYOPADHYAY, 2012). Além destas, outras revisões exploram a compressão vertical de dados genômicos e avaliam ferramentas para este fim.

Tabela 1.1: Revisões da literatura sobre compressão de dados genômicos – adaptada de (KREDENS et al., 2020)

Ano de publicação	**Título**	**Jornal**
2009	*Textual data compression in computational biology: a synopsis*	*Bioinformatics*
2009	*Data Compression Concepts and Algorithms and Their Applications to Bioinformatics*	*Entropy*
2012	*Recent Directions in Compressing Next Generation Sequencing Data*	*Current Bioinformatics*
2012	*Textual data compression in computational biology: Algorithmic techniques*	*Elsevier Computer Science Review*
2013	*Data compression for sequencing data*	*Algorithms for Molecular Biology*
2013	*Compressive biological sequence analysis and archival in the era of high-throughput sequencing technologies*	*Briefings in Bioinformatics*
2013	*DNA Lossless Compression Algorithms: Review*	*American Journal of Bioinformatics Research*
2013	*High-throughput DNA sequence data compression*	*Briefings in Bioinformatics*
2014	*Trends in Genome Compression*	*Current Bioinformatics*
2016	*A Survey on Data Compression Methods for Biological Sequences*	*Information*
2019	*Application of signal processing for DNA sequence compression*	*IET Signal Processing*
2019	*Towards Context-Aware DNA Sequence Compression Algorithms*	*IEEE (ICECCT)*
2020	*Vertical lossless genomic data compression tools for assembled genomes: A systematic literature review*	*PLOS ONE Journal*
2021	*Compression Techniques for DNA Sequences: A Thematic Review*	*Journal of Computing Science and Engineering*

Devido à relevância inerente dos métodos de compressão e à considerável variabilidade das abordagens disponíveis, foi motivada uma revisão sistemática da literatura. O objetivo principal desta revisão é identificar ferramentas especializadas existentes para comprimir coleções de sequências genômicas montadas.

Portanto, para obter uma perspectiva abrangente sobre as características salientes das ferramentas de compressão computacional, duas questões importantes foram formuladas a serem abordadas:

- Quais são as ferramentas existentes para compressão vertical de dados genômicos e de que forma estas processam o conteúdo da sequência e como fornecem acesso aos dados comprimidos?
- Quais técnicas são usadas por estas ferramentas para detectar segmentos compartilhados e codificar o resultado?

No entanto, uma análise dos resultados veiculados por diversos autores na literatura revela taxas de compressão discrepantes entre as ferramentas especializadas. Tal variabilidade pode ser atribuída ao fato de que os dados foram gerados e avaliados sob condições distintas em cada estudo, tornando-os inadequados para comparações diretas entre as ferramentas de compressão vertical. Este obstáculo surge em decorrência da ausência de um protocolo padronizado para a homogeneização do conjunto de dados avaliados. Tal constatação encontra respaldo em outros trabalhos (GIANCARLO; ROMBO; UTRO, 2014; WANDELT; BUX; LESER, 2014), que também apontam para a falta de um protocolo unificado para a avaliação do desempenho das ferramentas de compressão vertical. Entre os estudos analisados, foram identificadas as seguintes discrepâncias:

a) a ausência de um *dataset* padronizado, implicando que os estudos consultados empregaram sequências genômicas variadas, que vão desde extensas sequências de animais e plantas até sequências mais diminutas de microrganismos;

b) a adoção de métodos divergentes para a coleta e apresentação de métricas destinadas à avaliação de desempenho das taxas de compressão; e

c) o recurso a um espectro limitado de ferramentas de compressão especializadas como *baseline*, isto é, em certos casos, restringiu-se a comparação ao uso de ferramentas de compressão genéricas, tais como ZIP ou GZIP.

Com relação às taxas de compressão apresentadas por cada ferramenta analisada, constatou-se um incremento substancial, da ordem de centenas ou até milhares de vezes, nas dimensões dos arquivos ou nos conjuntos de dados pós-compressão. Tal variabilidade na eficiência da compressão, conforme articulada pelos autores, é atribuída predominantemente à elevada similaridade entre as sequências genômicas em questão. No entanto, uma lacuna evidente nos estudos revisados é a ausência de uma análise que correlacione as características gramaticais intrínsecas às sequências biológicas e seus respectivos níveis de similaridade com a eficácia das taxas de compressão. Deste modo, estabelecer um conjunto de parâmetros que correspondam especificamente às características dos conjuntos de sequências genômicas poderá não somente aprimorar a taxa de compressão, mas também facilitar a seleção da abordagem metodológica mais adequada.

Por exemplo, nos resultados disponibilizados pelos autores da ferramenta especializada ABRC, constatou-se uma disparidade significativa nas taxas de compressão ao comparar genomas de espécies distintas. Utilizando um conjunto de dados de sequências do genoma de *Homo sapiens*, a referida ferramenta alcançou uma taxa de compressão de 397:1. Contudo, ao processar as sequências de *Saccharomyces Cerevisiae*, a razão decresceu para 61:1. A partir destes dados, emergem questões pertinentes que demandam investigação adicional:

1) Quais são as características biológicas ou gramaticais intrínsecas a esses dois conjuntos de dados que ocasionam tal disparidade nas taxas de compressão?

2) A frequência de repetições de subsequências ou a presença de outras regiões repetitivas poderiam ser os fatores responsáveis por essa variação nas taxas de compressão?

3) Distintos tipos de variações genéticas, tais como polimorfismos de nucleotídeo único (*SNP*), inserções e exclusões de nucleotídeos (*INDEL*), ou variações estruturais (*SV*), poderiam ser os catalisadores dessa divergência nas taxas de compressão?

Constatou-se que as ferramentas de compressão genômica examinadas não adotaram um critério uniforme para representar os tamanhos dos conjuntos de

dados de entrada, nem um método padronizado para apresentar os resultados da compressão, as métricas utilizadas e as estratégias adotadas para avaliação da eficiência compressiva. Adicionalmente, notou-se que diversas ferramentas empregaram a sequência do genoma humano identificada como KOREF_20090224 (AHN et al., 2009) como base para os testes de desempenho. Contudo, cada ferramenta especializada reportou dimensões distintas para a mesma sequência genômica (cf. Tabela 1.2). Enquanto múltiplos fatores podem explicar essas discrepâncias, tais como o emprego de diferentes sistemas de arquivos para o armazenamento de dados ou a inclusão de etapas de pré-processamento prévias à compressão, a ausência de homogeneidade na representação inicial dos conjuntos de dados reitera a imperativa necessidade de estabelecer um padrão normativo para tal finalidade.

Tabela 1.2: Discrepâncias nos Tamanhos Reportados Por Autores para o Genoma Humano KOREF_20090224 – adaptada de (KREDENS et al., 2020)

Ferramenta de Compressão	**Tamanho e Magnitude Reportado**
HiRGC	2.987 MB
FM-context	3.080.419.480 *bytes*
MLF	3.080.436.051 *bytes*
NRGC	2.938 MB
SLF	3.080.436.051 *bytes*
ERGC	2.938 MB
iDoComp	3.100 MB
DnaCompact	2.937,7 MB
Dai et al.	3.080.436.051 *bytes*
COOL	2.938 MB
GRS	2.986,8 MB
GReEn	3.131,78 MB
Tamanho Referencial[a]	3.131.776.827 *bytes*

Nota explicativa: Tamanhos do genoma KOREF_20090224, conforme reportados em publicações originais associadas a cada ferramenta de compressão. **Nota**[a]: Valor total, em *bytes*, obtido a partir da soma dos *bytes* dos arquivos no formato FASTA, disponíveis no endereço ftp://ftp.kobic.kr/pub/KOBIC-KoreanGenome/KOREF_20090224/fasta/, consultado em 30/01/2022.

Com base nas observações previamente delineadas, torna-se imperativo estabelecer um protocolo normativo para a avaliação de desempenho, de modo a assegurar que as ferramentas especializadas em compressão de sequências genômicas sejam comparáveis entre si (DEOROWICZ et al., 2016). Assim, poderiam ser contempladas as seguintes diretrizes normativas:

a) uniformização do ambiente computacional para a execução das ferramentas em análise;

b) elaboração de um conjunto abrangente de métricas de avaliação de desempenho, que podem incluir: tempo de compressão e descompressão, alocação de memória para compressão e descompressão, taxa de compressão e percentual de economia de espaço de armazenamento;

c) estabelecimento de pontos temporais padronizados para a coleta das métricas elencadas no item (b);

d) sistematização e padronização das informações divulgadas, de forma a garantir a reprodutibilidade dos experimentos realizados;

e) uniformização dos formatos de apresentação dos resultados obtidos;

f) padronização na seleção dos conjuntos de dados genômicos empregados nas avaliações de desempenho das ferramentas; e

g) equivalência na seleção das ferramentas de referência (*baseline*) para comparações de eficácia e eficiência.

Dado o cenário discutido até o momento, notou-se a evidente carência de uniformidade tanto na execução dos testes de desempenho quanto na divulgação dos resultados por ferramentas especializadas em compressão genômica. De particular interesse é a lacuna observada na literatura no âmbito da compressão genômica vertical sem perda de informação, onde o emprego de algoritmos de aprendizado de máquina, com um enfoque na aprendizagem profunda, é relativamente escasso.

Este panorama é especialmente intrigante porque as obras citadas frequentemente negligenciam o uso das características biológicas intrínsecas às sequências genômicas como dados de entrada para o algoritmo de compressão. Consequentemente, emergem questões acerca do desenvolvimento de um sistema de compressão que incorpore elementos de aprendizagem capazes de alcançar taxas de economia de espaço comparáveis às obtidas pelas ferramentas especializadas em compressão vertical. Uma abordagem que incorpora técnicas de aprendizado de máquina não apenas tem o potencial de melhorar as taxas de compressão, mas também confere ao sistema propriedades preditivas. Estas, por sua vez, podem ser relevantes para futuras investigações voltadas para a correção de erros em sequências genômicas montadas, por meio de mecanismos de predição.

Assim, vislumbra-se a necessidade de uma pesquisa adicional que explore essa direção, fornecendo tanto um *benchmark* de desempenho robusto quanto avanços teóricos e aplicados no campo da bioinformática e aprendizagem de máquina.

O emprego de técnicas de aprendizagem profunda, particularmente CNN (FUKUSHIMA, 1980) e LSTM (HOCHREITER; SCHMIDHUBER, 1997), na compressão de sequências genômicas, têm se mostrado uma estratégia promissora para a predição de características biológicas (QUANG; XIE, 2016; ZHOU; TROYANSKAYA, 2015). Ferramentas relacionadas a esse tema, que objetivam a compressão sem perda de informação das sequências genômicas, foram identificadas na literatura. Notavelmente, essas ferramentas tipicamente adotam um fluxo de trabalho bifásico, dividido em dois componentes fundamentais:

a) A geração de um modelo de predição, comumente fundamentado em arquiteturas de redes neurais, com a finalidade de estimar a probabilidade da ocorrência do próximo símbolo na sequência. Este modelo pode, por exemplo, levar em conta tanto características locais como globais da sequência, aprimorando assim a precisão da predição.

b) A aplicação subsequente de um algoritmo de compressão particular para transformar cada item de dado com base nas probabilidades resultantes do modelo de predição. Neste contexto, métodos de compressão aritmética são frequentemente empregados devido à sua eficácia na redução do tamanho do arquivo enquanto mantêm a integridade dos dados.

Portanto, essa abordagem bifásica não apenas possibilita a compressão eficiente das sequências genômicas, mas também abre espaço para a incorporação de elementos preditivos que podem enriquecer o contexto biológico dessas sequências. Esta integração entre a previsão do próximo símbolo e as técnicas de transformação e codificação de dados destaca o potencial das técnicas de aprendizagem profunda na área de compressão de sequências genômicas, sem perda de informação.

A ferramenta denominada *DeepDNA* (WANG et al., 2018) foi conceptualizada com inspiração na ferramenta *DeepZip* (GOYAL et al., 2019), a qual tem como foco a compressão de texto. Ambas as ferramentas, incluindo o compressor de texto *CMIX* (KNOLL, 2021), fundamentam-se em modelos de redes neurais recorrentes, especificamente as redes Long Short-Term Memory (LSTM) e Gated Recurrent Units (GRU). Estas estruturas neurais permitem a predição do símbolo subsequente por meio de uma única passagem pelos dados de entrada.

No caso do *DeepDNA*, a ferramenta foi configurada para operar com sequências genômicas mitocondriais humanas. No entanto, uma lacuna notável em sua arquitetura é a negligência de fatores locais e globais inerentes à sequência genômica, como palíndromos e repetições (tanto exatas quanto aproximadas), na estimativa da probabilidade de ocorrência do próximo símbolo. Essa limitação foi parcialmente abordada por outra investigação (CUI et al., 2020), na qual foi proposto um algoritmo de compressão que integra essas características locais e globais na arquitetura de rede neural. O estudo em questão empregou uma arquitetura de biLSTM, conhecida por sua capacidade de analisar dados sequenciais em direções opostas. Tal característica permite que a biLSTM capture de forma mais eficaz as dependências de longo prazo existentes em ambas as direções da sequência genômica.

O corpus científico até agora revela uma lacuna notável na aplicação de ferramentas de compressão baseadas em aprendizagem profunda para sequências genômicas de grande escala, particularmente cromossomos humanos e plantas. Embora o trabalho de (CUI et al., 2020) alude à possibilidade de comprimir conjuntos de dados de genoma humano, os experimentos realizados foram em grande medida circunscritos a genomas de dimensões substancialmente menores. Por exemplo, a maior sequência genômica analisada pertence ao organismo Peixe com Nadadeiras (*Ray-finned fishes*) da classe *Actinopterygii*, com aproximadamente 170 mil bases nitrogenadas, um tamanho que pálida em comparação

com cromossomos humanos, como o cromossomo 22, que conta com cerca de 50 milhões de bases nitrogenadas.

Este tamanho reduzido de genomas avaliados têm implicações diretas na eficiência e eficácia da compressão. Nomeadamente, modelos preditivos que são ajustados a genomas de pequeno porte tendem a ser menos complexos e, portanto, requerem menos espaço de armazenamento. Isso é especialmente relevante quando consideramos que esses modelos preditivos, uma vez construídos, devem ser armazenados para uso subsequente durante o processo de descompressão. Portanto, tais modelos funcionam essencialmente como uma forma de referência da sequência, que é imperativa tanto para a operação de compressão quanto para a de descompressão.

Dada a importância do armazenamento desses modelos preditivos para o funcionamento eficiente da compressão e descompressão, uma questão pertinente é a escalabilidade desses métodos quando aplicados a genomas de dimensões mais representativas da realidade biológica. Este aspecto é crucial não apenas para a viabilidade técnica da abordagem, mas também para a aplicabilidade prática em contextos de uso clínico que requerem a manipulação de genomas de grande escala.

Com base nas informações anteriormente delineadas, este estudo visa avaliar duas ferramentas especializadas em compressão de sequências genômicas, ambas fundamentadas em modelos preditivos de Aprendizado Profundo (*Deep Learning*). Para alcançar este objetivo, propõe-se a utilização de um conjunto de dados composto por sequências de mitocôndrias humanas, as quais serão submetidas a procedimentos de predição do próximo símbolo, compressão e descompressão. Além disso, o trabalho engloba, de forma exploratória, a avaliação de um conjunto de características biológicas e propriedades estatísticas que possam atuar como variáveis de entrada, com o intuito de otimizar a taxa de compressão e consequentemente potencializar os percentuais de economia de espaço. Por último, o estudo buscará padronizar os métodos de compressão e descompressão empregados, fornecendo uma análise comparativa que envolve ferramentas *baseline* e indicadores estatísticos, de acordo com as métricas propostas neste trabalho.

1.1 Objetivo

O objetivo primordial desta investigação científica é o de formular um método de compressão de sequências genômicas que adote uma abordagem vertical, garantindo a integridade informacional e fundamentando-se em modelos preditivos. Adicionalmente, aspira-se à proposição de um protocolo para a avaliação comparativa do desempenho das ferramentas de compressão em questão.

Para a efetiva realização do objetivo geral previamente enunciado, estabelecem-se os seguintes objetivos específicos, categoricamente enumerados:

1) Realização de uma revisão bibliográfica abrangente focada na compressão vertical de sequências genômicas e sem perda de informação, culminando na seleção de ferramentas pertinentes ao domínio de estudo.

2) Formulação de um protocolo para a avaliação comparativa do desempenho das ferramentas de compressão, visando à uniformidade dos procedimentos e à confiabilidade dos resultados obtidos.

3) Concepção de um método de compressão vertical de sequências genômicas, sem perda de informação, fundamentado em modelos preditivos.

4) Avaliação de duas ferramentas *baseline* por meio de execução de testes de compressão sob as diretrizes do protocolo proposto.

1.2 Limitações do escopo

O escopo deste trabalho está limitado a realizar a revisão da literatura sobre ferramentas especializadas de compressão vertical de sequências genômicas e sem perda de informação. Estas ferramentas devem seguir as regras estabelecidas da Seção 2. Além disso, está limitado a definir um protocolo mínimo que padronize a avaliação comparativa de desempenho entre as ferramentas de compressão, conforme descrito na Seção 3. Ainda, está limitado a avaliar duas

ferramentas de compressão de genoma baseada em modelo preditivo (*Deep Learning*), levando em conta características biológicas-estatísticas da sequência, de acordo com a Seção 3. Está também limitado a aplicar o protocolo sobre os testes comparativos realizados entre a ferramenta baseada em aprendizagem profunda proposta e as ferramentas *baseline* indicadas na Seção 3.

1.3 Organização do documento

Além do corrente Capítulo, este trabalho está organizado da seguinte forma: O Capítulo 2 oferece uma revisão sistemática e abrangente da literatura, focando em ferramentas especializadas na compressão vertical de dados genômicos sem comprometimento da integridade informacional. No Capítulo 3, é delineado o método empregado na criação de uma ferramenta de compressão que utiliza um modelo preditivo. Esse capítulo também introduz um conjunto padronizado de métricas para avaliação de desempenho. O Capítulo 4 documenta o processo de teste da ferramenta desenvolvida, com o intuito de mensurar sua eficácia em comparação com duas outras ferramentas de referência previamente selecionadas. Este capítulo aborda de forma sistemática os critérios, métodos e resultados dos testes realizados, proporcionando uma base sólida para a comparação do desempenho das ferramentas analisadas. O Capítulo 5 conclui o documento, sintetizando os resultados alcançados e delineando direções potenciais para a continuação da pesquisa. Tal organização visa proporcionar uma exposição clara e sistemática dos contributos deste trabalho.

2

Revisão Sistemática da Literatura

Este capítulo apresenta o estado da arte em ferramentas especializadas na compressão de dados de sequência genômica, com uma abordagem vertical e sem perda de informação, publicada na revista PLOS ONE (KREDENS et al., 2020). Neste contexto, delineia-se a estratégia adotada para a busca de artigos na literatura científica e os critérios empregados para a seleção dos artigos alvo. Adicionalmente, são elucidados os detalhes do fluxo de trabalho implementado por essas ferramentas, bem como as técnicas que elas aplicam em cada etapa do processo. Tais técnicas podem ser aplicadas a sequências genômicas armazenadas em arquivos no formato *FASTA*. Cada técnica pode atender a múltiplos objetivos, tais como detecção de segmentos, indexação da sequência, realização de mapeamentos de primeira ou segunda ordem, redução da entropia da informação, reorganização, agrupamento ou ordenação dos dados de uma sequência específica e, finalmente, a codificação. Para uma compreensão aprofundada dos fundamentos da compressão de dados, sugere-se ao leitor os seguintes livros: (SALOMON, 2007) e (MCANLIS; HAECKY, 2016).

2.1 Introdução

Antes de abordar o tema das ferramentas de compressão vertical, sem perda de informação, para sequências genômicas, faz-se necessário examinar a proveniência e a natureza dos dados que tais ferramentas se propõem a comprimir. Assim, até que um genoma existente na célula de um organismo vivo seja ma-

peado de seu estado biológico para uma representação digital, há um percurso a ser trilhado (cf. Figura 2.1).

A molécula de *Deoxyribonucleic Acid (DNA)* está configurada na forma de uma dupla hélice. As extremidades de uma fita são indicadas por 5′ (*cinco linha* ou *five prime*) e 3′ (*três linha* ou *three prime*), respectivamente. Conforme o modelo de replicação estabelecido por (MESELSON; STAHL, 1958), as duas fitas da hélice se orientam em direções opostas; isto é, a extremidade 5′ da fita molde é pareada com a extremidade 3′ da fita complementar. Deste modo, cada fita possui uma cadeia de nucleotídeos "flutuantes"que contêm as bases nitrogenadas Adenina, Citosina, Guanina e Timina, representadas respectivamente pelos símbolos A, C, G e T. As bases nitrogenadas A e T são complementares entre si, assim como as bases C e G.

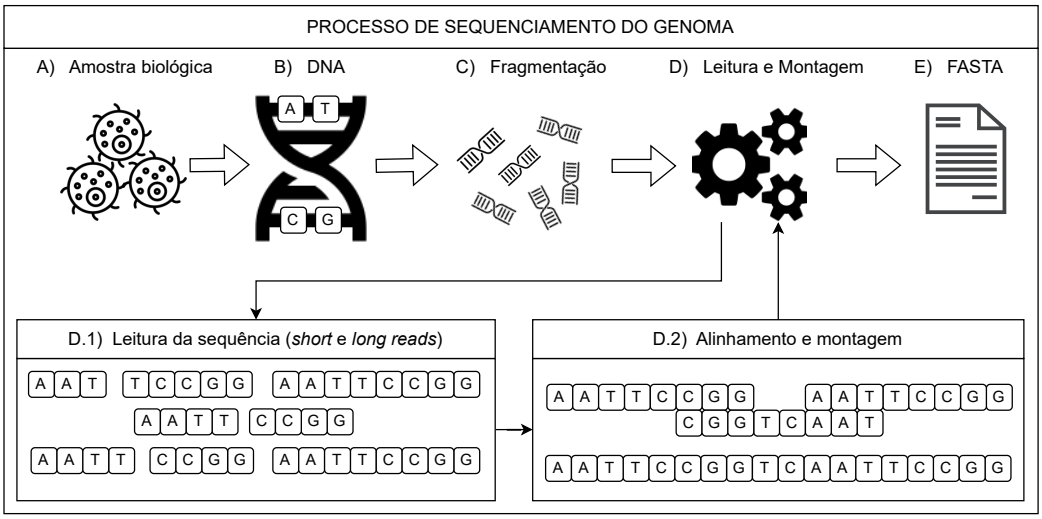

Figura 2.1: Fluxograma do processo de sequenciamento de genoma – o autor 2024

O processo de sequenciamento genético inicia-se com a coleta de uma amostra biológica de um organismo (cf. Figura 2.1A), que serve como insumo para o equipamento de sequenciamento NGS. Subsequentemente, o material genético é isolado, consistindo na identificação e extração das moléculas de DNA (cf. Figura 2.1B). Dependendo do objetivo almejado e da tecnologia utilizada (HEATHER; CHAIN, 2016; KRAFT; KURTH, 2019), as moléculas de DNA presentes na amostra são fragmentadas mediante processos químicos (cf. Figura 2.1C). Estes fragmentos variam em tamanho: para sequenciamento de leitura curta (*short-reads*), os tamanhos oscilam entre aproximadamente 35 e 1000 pa-

res de bases nitrogenadas; para sequenciamento de leitura longa (*long-reads*), os tamanhos vão de aproximadamente 5000 a 30000 pares (cf. Figura 2.1D1).

A sequência dos símbolos que representam estas bases nitrogenadas é registrada em uma ordem idêntica àquela em que são decodificadas pelo equipamento de sequenciamento. No entanto, inerentes limitações do *hardware* de leitura podem introduzir incertezas na identificação precisa das bases nitrogenadas. Em tais circunstâncias, símbolos adicionais, conforme estabelecido em referências pertinentes (IUBMB-IUPAC, 1984), podem ser utilizados para denotar ambiguidade ou imprecisão na decodificação da sequência. Estes símbolos adicionais são padronizados e descritos na Tabela 2.1.

Tabela 2.1: IUPAC - Símbolos atribuídos à representação de bases nitrogenadas em função da precisão da leitura – adaptada de (IUBMB-IUPAC, 1984)

Base	**Símbolo**	**Base**	**Símbolo**
Leitura precisa			
Adenina	A	Guanina	G
Citosina	C	Timina	T
Leitura ambígua			
A ou G	R	C ou G ou T	B
C ou T	Y	A ou G ou T	D
G ou C	S	A ou C ou T	H
A ou T	W	A ou C ou G	V
G ou T	K	A ou C ou G ou T	N
A ou C	M	Falha na leitura	. ou -

Finalmente, o alinhamento dos fragmentos *reads* é realizado, culminando na montagem da sequência genômica (cf. Figura 2.1D2). Após a conclusão do processo de sequenciamento, o equipamento fornece a sequência genômica em formatos digitais. Um desses formatos é o *FASTA* (ZHU et al., 2015), que constitui o foco deste estudo (cf. Figura 2.1E). Este formato é estruturado de tal forma que permite tanto a leitura quanto a interpretação pelos seres humanos.

O sequenciamento de um genoma humano completo, quando armazenado em um arquivo *FASTA*, pode alcançar um comprimento total de aproximadamente $3,2$ bilhões de símbolos, os quais representam as bases nitrogenadas. Isso equivale a um volume de dados de $3,2$ Gigabytes. Com a aceleração dos processos de sequenciamento genômico e a redução dos custos associados, uma quantidade crescente de dados genômicos está sendo gerada anualmente. Até

meados de 2022, o custo para sequenciar 1 Megabase (1 milhão de bases) de sequência genômica situava-se em torno de 0,01 dólares (cf. Figura 2.2), enquanto o custo para sequenciar um genoma humano completo era inferior a 1000 dólares (cf. Figura 2.3). Consoante à Lei de Moore, a qual postula que a capacidade de processamento computacional duplica bi-anualmente para um custo constante, observa-se que a queda nos custos de sequenciamento genômico tem superado o ritmo de avanço da capacidade computacional.

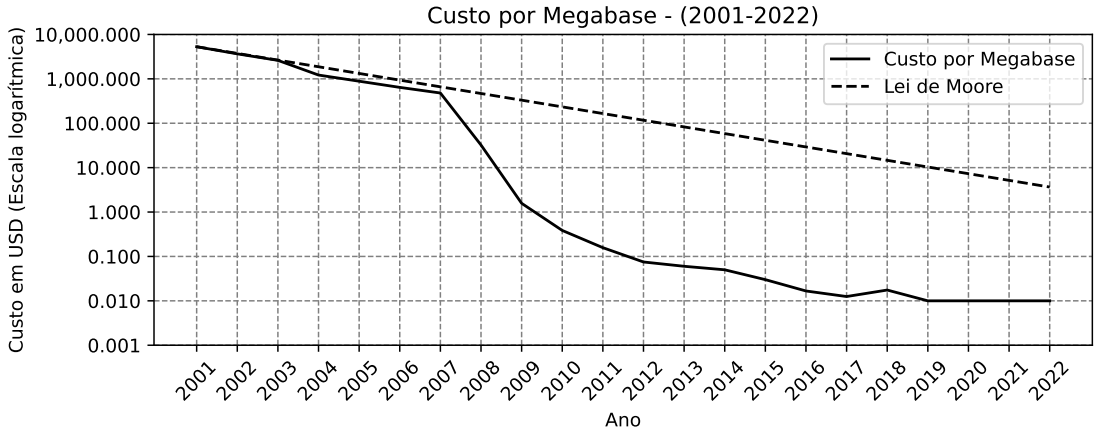

Figura 2.2: Custo do sequenciamento por Megabase do genoma – adaptada de (WETTERSTRAND, 2022)

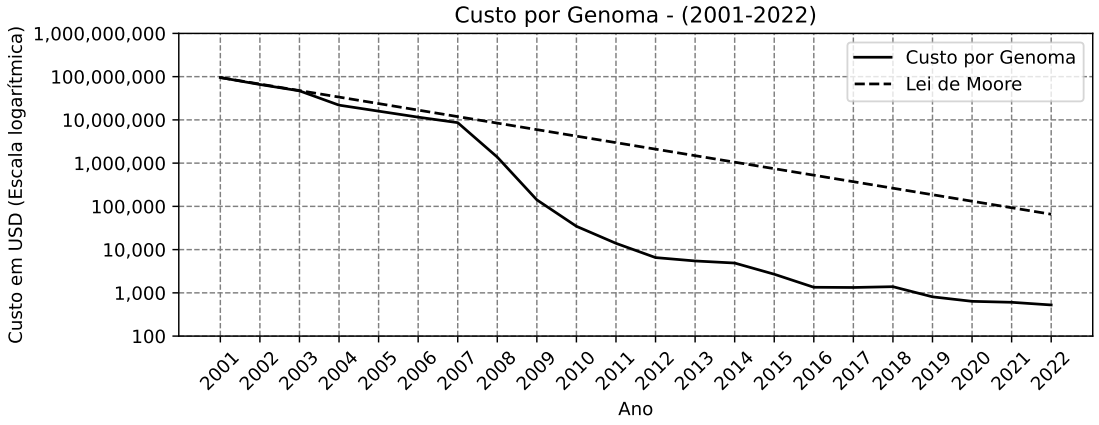

Figura 2.3: Custo do sequenciamento por genoma – adaptada de (WETTERSTRAND, 2022)

Em face do grande acervo de dados de sequenciamento genético já existente, bem como da trajetória ascendente que o campo do sequenciamento genômico está percorrendo, tornam-se imperativos esforços em pesquisa e desenvolvi-

mento para conceber métodos de armazenamento mais eficientes e economicamente viáveis. Segundo as projeções de (STEPHENS et al., 2015) acerca do volume total de genomas que serão sequenciados até 2025, um ganho de espaço de apenas 1% no volume sequenciado equivale à média de 3.125 bilhões de arquivos *FASTA* contendo sequências de genomas humanos completos.

O método adotado para a revisão sistemática da literatura neste estudo fundamenta-se no protocolo elaborado por (KITCHENHAM; CHARTERS, 2007). O principal objetivo desta revisão é identificar e sintetizar o estado da arte em ferramentas computacionais destinadas à compressão vertical de sequências genômicas armazenadas no formato *FASTA*. Tal compressão deve ser realizada sem perda de informação e sem a necessidade de conhecimentos prévios, como, por exemplo, mapas de mutações esperadas para um determinado organismo.

Aspectos relacionados à estratégia de busca na literatura, critérios de inclusão de artigos e estratégias para a extração de dados serão abordados em seções subsequentes.

2.2 Estratégia de busca

A estratégia de busca adotada foi delineada em duas fases distintas. A primeira fase envolveu um estudo exploratório, centrado na composição dos índices, na identificação das fontes de pesquisa e na delimitação de parâmetros temporais. A segunda fase foi dedicada à filtragem, análise e comparação das obras recuperadas, utilizando a estratégia de busca previamente estabelecida. Ambas as fases foram cruciais para sistematizar e orientar o alcance do objetivo final da revisão da literatura.

Uma *string* de busca foi elaborada, incorporando diversas palavras-chave e três conceitos distintos (cf. Tabela 2.2). O objetivo principal e as questões de pesquisa nortearam a seleção dessas palavras-chave, resultando na seguinte expressão de busca: *((compression) AND (genome OR genetic OR dna OR nucleotide OR "biological sequence"OR genomic) AND (relative OR referential OR reference OR model OR dictionary OR codebook OR collection OR multiple OR vertical OR content OR set))*. A validação dessa *string* de busca foi realizada por meio de um processo iterativo, que incluiu a execução de uma busca preliminar e a subsequente comparação dos resultados com outras revisões e trabalhos já estabelecidos na literatura. Durante esse processo, observou-se a ausência de termos como

genomic e *content*. Optou-se por não incluir sinônimos para a palavra-chave *compression*, uma vez que, com base nos resultados da fase exploratória, não se identificou outra palavra-chave que pudesse ser associada ao conceito de "*data compression*".

Tabela 2.2: Conceitos e Palavras-Chave Utilizadas na Construção da *String* de Pesquisa – adaptada de (KREDENS et al., 2020)

Conceito	**Palavras-Chave Associadas**
Compressão de Dados	*compression*
Dados Biológicos	*genome; DNA; genetic; nucleotide; biological sequence; genomic*
Compressão Vertical	*relative; referential; reference; model; dictionary; codebook; collection; multiple; vertical; content; set*

Nota: Conceito refere-se à abstração que se busca cobrir. **Palavras-Chave Associadas** são termos empregados para capturar a respectiva abstração.

Embora o foco primordial desta revisão seja a identificação de ferramentas computacionais para a compressão vertical de dados genômicos sem perda de informação, deliberou-se não incluir na *string* de pesquisa palavras-chave que especificam o atributo "sem perda" (*lossless*). Dessa forma, a *string* de busca foi concebida para permitir a recuperação de ferramentas que operam tanto com quanto sem perda de dados.

As fontes de dados selecionadas para esta revisão foram PubMed (PubMed Central (PMC), 2023; MEDICINE, 2023) e Scopus (ELSEVIER, 2004; ELSEVIER, 2023). De acordo com informações obtidas, PubMed indexa cerca de 30.000 periódicos (MEDICINE, 2023), conforme descrito no site oficial da *National Library of Medicine*. Por outro lado, Scopus, que agora é propriedade da Elsevier, indexa mais de 84 milhões de registros (ELSEVIER, 2023) (cf. Tabela 2.3).

A investigação foi conduzida mediante a utilização da interface de pesquisa fornecida pelas respectivas máquinas de busca. Foi optado pela pesquisa de texto completo, sem a imposição de restrições a áreas temáticas específicas, a fim de não limitar indevidamente o escopo da análise. Esse procedimento permitiu a configuração de um conjunto inicial de trabalhos acadêmicos, os quais foram identificados com base na aplicação de uma estratégia de busca bem definida.

Subsequentemente, foi realizada uma etapa de depuração dos dados coletados. Nesta fase, trabalhos duplicados foram identificados e removidos de forma

Tabela 2.3: Máquinas de busca pesquisadas – adaptada de (KREDENS et al., 2020)

Base	**URL**	**Conteúdo indexado**
PubMed[1]	<http://ncbi.nlm.nih.gov/pubmed>	Aprox. 30.000 periódicos
Scopus[2]	<http://scopus.com>	Aprox. 84.000.000 registros

[1] PubMed é uma base de dados de livre acesso à base de dados MEDLINE de citações e resumos de artigos de pesquisa biomédica.
[2] Scopus é uma base de dados bibliográfica de resumos e citações de artigos de revistas científicas.
Base: Nome da base pesquisada.
URL: Endereço eletrônico do mecanismo de busca utilizado.
Conteúdo indexado: Quantidade aproximada de trabalhos indexados.

automatizada, baseando-se em critérios tais como o título da obra, o ano de publicação e a lista de autores contribuintes. Após a execução desse procedimento, critérios de inclusão específicos foram aplicados para estabelecer o conjunto final de obras primárias submetidas à análise. Os pormenores desses critérios são delineados nas seções subsequentes.

2.3 Critérios de Inclusão

Com vistas a direcionar o foco analítico em direção a trabalhos de relevância intrínseca ao objetivo desta revisão sistemática, estabeleceram-se critérios de inclusão aplicados ao conjunto inicial de artigos identificados. Uma obra específica é admitida no conjunto de obras primárias se ela satisfaz as seguintes condições:

1) Propor uma abordagem para compressão vertical de dados genômicos que seja isenta de perda de informação;

2) Possibilitar a compressão de uma sequência genômica mediante informações advindas de sequências genômicas correlatas, satisfazendo assim a definição de compressão vertical;

3) Oferecer compatibilidade com arquivos no formato *FASTA*;

4) Operar independentemente de qualquer forma de informação *a priori*;

5) Disponibilizar uma codificação funcional do algoritmo proposto, excluindo-se, portanto, meras proposições teóricas.

A aplicação desses critérios de inclusão foi realizada em duas etapas sequenciais: inicialmente, os critérios 1 e 2 foram aplicados à análise dos títulos e resumos dos trabalhos; subsequentemente, os critérios de 3 a 5 foram aplicados ao corpo integral dos artigos.

Excluíram-se da presente revisão várias categorias de trabalhos que não atenderam aos critérios estabelecidos, a saber: (a) aqueles fundamentados em auto-índices como E2FM (MONTECUOLLO; SCHMID; TAGLIAFERRI, 2017), LZ-End (MäKINEN et al., 2010a), e RLZ 1.4 (PROCHáZKA; HOLUB, 2014); (b) trabalhos que utilizam sequências genômicas meramente como variáveis de entrada para avaliação de desempenho, mas que não se concentram na compressão de dados genômicos *per se*, como é o caso de Ray (CANNANE; WILLIAMS, 2001) e Cobald (PEEL; WIRTH; ZOBEL, 2011); (c) abordagens de compressão horizontal que facultam a compressão vertical, mas omitem detalhes algorítmicos, como Biocompress (GRUMBACH; TAHI, 1993), Biocompress-2 (GRUMBACH; TAHI, 1994), e Hex-LRE (SAADA; ZHANG, 2015); (d) métodos que comprimem múltiplas sequências de forma simultânea, mas sem utilizar informações de forma vertical como GtEncseg (STEINBISS; KURTZ, 2012); (e) algoritmos incompatíveis com o formato *FASTA*, tal como TGC (DEOROWICZ; DANEK; GRABOWSKI, 2013); (f) métodos que requerem informações *a priori*, tais como um banco de dados de variantes genéticas conhecidas, como DNAzip (CHRISTLEY et al., 2009) e GenomeZIP (PAVLICHIN; WEISSMAN; YONA, 2013); e (g) adaptações de ferramentas pré-existentes para novas plataformas (DU et al., 2017).

2.4 Processo de Extração de Dados

O procedimento de extração de dados foi desenhado para assegurar a aquisição sistemática de informações relevantes aos questionamentos de pesquisa previamente definidos. A extração de dados foi efetuada mediante a leitura de cada artigo integrante do conjunto de trabalhos selecionados. Dados pertinentes foram então organizados em registros de dados armazenadas em ambiente de armazenamento em nuvem.

Esta etapa foi conduzida de forma independente por três pesquisadores especializados no domínio em questão. Quaisquer discordâncias ou ambiguidades emergentes durante este processo foram objeto de avaliação conjunta e resolução individualizada.

Os questionamentos de pesquisa foram formulados com base nos dados extraídos dos textos principais de cada obra selecionada. Adicionalmente, quando se fez necessário, recorreu-se a fontes complementares de dados, tais como *websites* oficiais, repositórios de código-fonte e documentação associada ao projeto de pesquisa em questão.

É digno de nota que em determinados casos, os autores propuseram múltiplas abordagens dentro de um único manuscrito. Nestas instâncias, os dados pertinentes a todas as abordagens foram sistematicamente extraídos e consolidados. Nas seções subsequentes, tais trabalhos são representados como entidades unitárias. Tal decisão metodológica foi tomada com vistas a oferecer uma visão coesa e integrada do cenário atual relativo às abordagens e ferramentas existentes, obviando a necessidade de segregação e apresentação de cada implementação como uma entidade distinta.

Na realização da revisão literária, cuja busca de títulos considerou publicações até o ano de 2022, as plataformas de busca *PubMed* e *Scopus* foram empregadas. Após a etapa de eliminação de duplicatas, restaram 46.982 artigos. Posteriormente, a aplicação dos critérios de inclusão resultou em um conjunto de 36 trabalhos aptos para avaliação, conforme ilustrado na Figura 2.4. É imperativo observar que critérios quantitativos como fator de impacto, número de citações ou periódicos em que os artigos foram publicados não foram adotados como métricas de exclusão. A partir deste momento, cada trabalho selecionado foi avaliado em relação às questões de pesquisa estabelecidas, mediante a leitura e análise do conteúdo integral do artigo.

2.5 Métodos de Compressão Vertical em Genômica

Nos trabalhos escolhidos para esta revisão, identificaram-se 36 distintas ferramentas voltadas para a compressão de dados genômicos, tal como apresentado na Tabela 2.5. De cada um desses artigos, foram extraídas informações como data de publicação, linguagem de programação utilizada, taxas máximas e mínimas de compressão (expressas como a relação entre o tamanho final e o tamanho

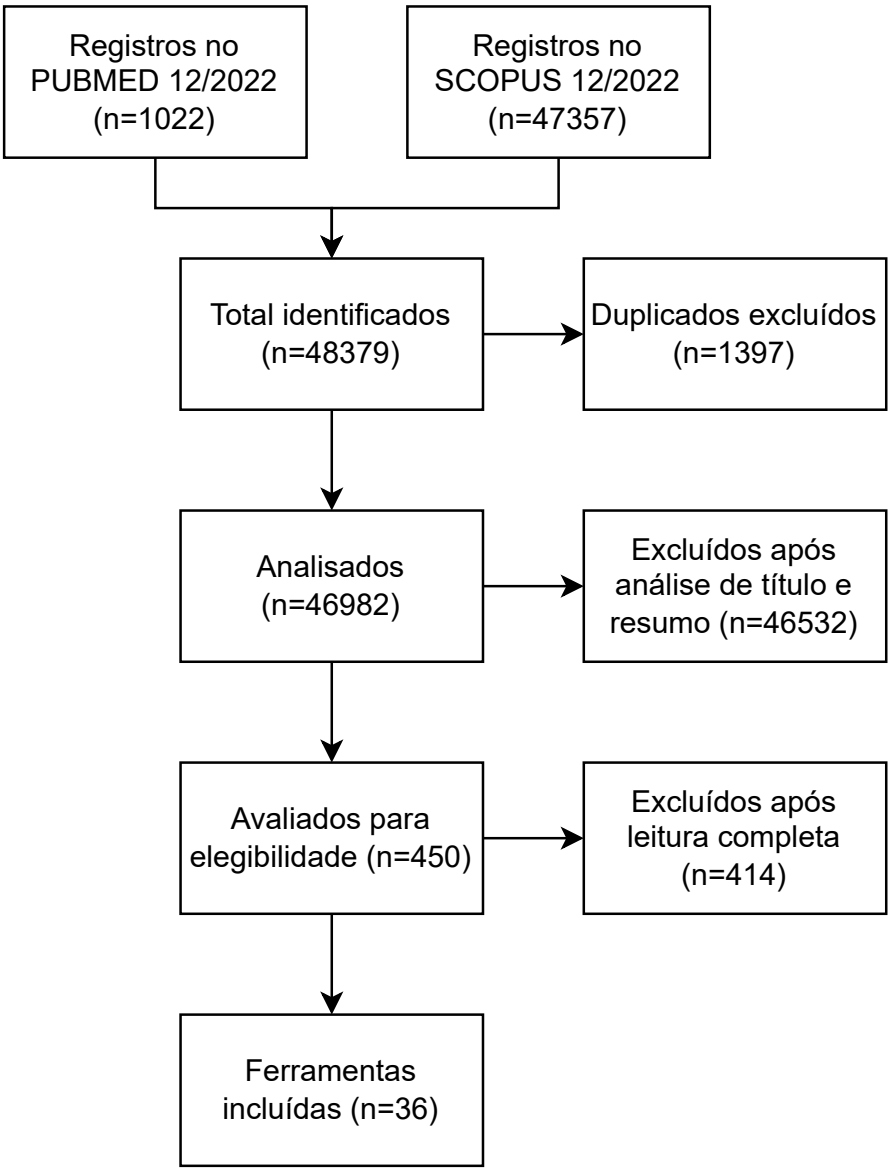

Figura 2.4: Esquema Ilustrativo do Procedimento de Identificação e Seleção de Estudos – adaptada de (KREDENS et al., 2020)

inicial da sequência genômica comprimida), esquema de compressão, alfabeto suportado, método de compressão do cabeçalho da sequência e utilização de memória externa. Além disso, foi observado o método disponível para acesso aos dados após a compressão. É relevante ressaltar que, embora o projeto de sequenciamento do genoma humano tenha sido concluído em 2001 (CONSORTIUM et al., 2001; J. Craig Venter, Mark D. Adams, Eugene W. Myers, Peter W. Li et al., 2001), todas as ferramentas dedicadas à compressão vertical de dados

genômicos foram publicadas somente após o ano de 2009, conforme mostrado na Figura 2.5A.

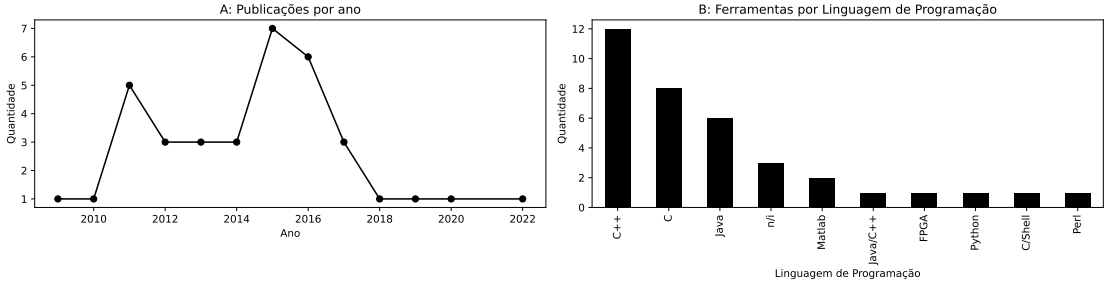

Figura 2.5: Análise Temporal e Linguística das Ferramentas de Compressão Vertical. (A) Distribuição anual das publicações referentes a ferramentas de compressão vertical; (B) Distribuição das ferramentas de compressão vertical por linguagem de programação. Nota-se que o número total de ferramentas ultrapassa 36, dado que a ferramenta HiRGC possui implementações nas linguagens de programação C++ e Java – adaptada de (KREDENS et al., 2020)

É digno de nota que aproximadamente 57,5% das ferramentas especializadas em compressão vertical de dados genômicos foram desenvolvidas utilizando as linguagens de programação *C* ou *C++*, conforme evidenciado na Figura 2.5B. Embora as razões específicas para a escolha destas linguagens de programação não sejam explicitamente delineadas pelos autores dos trabalhos analisados, é pertinente ressaltar que tanto *C* quanto *C++* são linguagens compiladas, o que frequentemente se traduz em tempos de execução mais eficientes em comparação com linguagens interpretadas (NEUMANN; LEIS, 2014).

Outro aspecto avaliado foi a taxa de compressão, conforme ilustrado na Tabela 2.4. Esta métrica é definida como a razão entre o volume de dados após a compressão e o volume original de dados não comprimidos, expressa pela fórmula:

$$\text{Taxa de Compressão} = \frac{\text{Tamanho Original}}{\text{Tamanho Comprimido}}$$

Por exemplo, uma sequência de dados de 40 MB que, após a compressão, tem seu tamanho reduzido para 8 MB resulta em uma taxa de compressão de $\frac{40}{8} = 5$, explicitamente representada pela proporção 5:1. Os dados referentes a esta métrica foram extraídos dos resultados apresentados em cada artigo selecionado, com priorização, sempre que possível, dos resultados associados às sequências genômicas de *Homo sapiens*. Contudo, é pertinente destacar que algumas ferramentas, como RCC, RLZAP, Mehta *et al.*, RLZ 1.4, RLZ RePair, SCCG, HRCM (YAO et al., 2019), memRGC (LIU; WONG; LI, 2020) e Project DNA Compression,

Tabela 2.4: Análise Comparativa de Desempenho Relativo às Taxas de Compressão entre Diversas Ferramentas – o autor.

Ferramenta	Taxa Máxima de Compressão	Taxa Mínima de Compressão
MBGC	91:1	87:1
memRGC	–a	–a
HRCM	79:1	–a
SCCG	–a	–a
RCC	**475:1**	**29:1**
HiRGC	218:1	83:1
FM-context	348:1	278:1
MLF	486:1	–a
RLZAP	**59:1**	**7:1**
NRGC	370:1	64:1
GeCo	25:1	9:1
RCSCS	588:1	369:1
Arram *et al.*	12772:1	10937:1
DNAComp	**207:1**	**17:1**
SLF	360:1	–b
JDNA	790:1	–a
ERGC	579:1	154:1
Mehta *et al.*	**9:1**	**8:1**
GDC 2.0	9557:1	183:1
CoGI	n/a c	244:1
iDoComp	3024:1	268:1
DNAC-K	91:1	–b
RLZ 1.4	**30:1**	–a
FRESCO	3057:1	–b
DNA-COMPACT	189:1	177:1
Dai *et al.*	408:1	244:1
ABRC	397:1	–b
COMRAD	5:1	–b
COOL	1920:1	288:1
GRS	159:1	–b
GReEn	171:1	99:1
GDC-0.3	1003:1	262:1
RLZ-Opt	17:1	–b
RLZ RePair	**75:1**	**21:1**
RLZ	16:1	–b
Project DNA Compression	**432:1**	**344:1**

–a) Ausência de dados adicionais fornecidos pelos autores. –b) Ausência de dados adicionais para o genoma de *Homo sapiens*. –c) Melhores resultados obtidos, porém representados apenas graficamente pelos autores.

não disponibilizaram dados específicos sobre a taxa de compressão alcançada para o conjunto de referência considerado. Além disso, alguns artigos, como o dos autores ARRAM et al. que apresentam as maiores taxas (máxima e mínima) de compressão na Tabela 2.4, não especificam se as proporções apresentadas são

expressas diretamente em termos de *bytes*, *kilobytes* ou *megabytes*. No entanto, baseados nos textos dos artigos analisados, parece ser comum expressar tais proporções como um fator de redução do tamanho original dos dados para o tamanho comprimido, sem especificar explicitamente a unidade, a menos que se refira aos tamanhos dos arquivos antes e depois da compressão.

Tabela 2.5: Ferramentas de compressão vertical de dados genômicos – adaptada de (KREDENS et al., 2020)

Ferramenta	Autor	Ano	Linguagem	Esquema	Alfabeto	Cabeçalho	Memória	Acesso
MBGC	GRABOWSKI; KOWALSKI	2022	C++	RF, CB, SR	ASCII	sim	não	n/i
memRGC	LIU; WONG; LI	2020	C++	RF, CB, SR	ASCII	sim	não	n/i
HRCM	YAO et al.	2019	C++	RF, PB, SR	ATCGN	sim	não	n/i
SCCG	SHI et al.	2018	Java	RF, CB, SR	ATCG	ignora	não	n/i
RCC	CHENG; LAW; SIU	2018	Matlab[a]	RF, CB, SR	ATCG	n/i[b]	não	n/i
HiRGC	LIU et al.	2017	Java/C++	RF, PB	ASCII	sim	não	n/i
FM-context	FAN et al.	2017	C++	RF, PB	ASCII	n/i	não	n/i
MLF	BEAL ALIYA FARHEEN	2016	n/i	RF, PB	ASCII	não	não	n/i
RLZAP	COX et al.	2016	C++	RF, CB, SR	ATCGN	n/i	não	RA, SR
NRGC	SAHA; RAJASEKARAN	2016	Java	RF, PB	ASCII	yes	yes	n/i
GeCo	PRATAS; PINHO; FERREIRA	2016	C	ST, PB	ATCG	não	sim	n/i
RCSCS	Wiselin Kiruba; RAMAR	2016	Java	RF, PB	n/i	n/i	não	n/i
Arram *et al.*	ARRAM et al.	2015	FPGA[c]	RF, PB	ATCG[d]	não	não	n/i
DNAComp	CHENG et al.	2015	Matlab[a]	RF, CB, MR[e]	ATCG	não	não	n/i
SLF	BEAL et al.	2015	C	RF, PB	IUPAC	não	não	n/i
JDNA	ALVES et al.	2015	Java	RF, PB	ATCGN	sim	sim	n/i
ERGC	SAHA; RAJASEKARAN	2015	Java	RF, PB	ASCII	sim	sim	não
Mehta *et al.*	MEHTA; GHRERA	2015	Java	DT, PB	n/i	n/i	não	n/i
GDC 2.0	DEOROWICZ; DANEK; NIEMIEC	2015	C++	RF, CB, MR[f]	ASCII	sim	não	SR
CoGI	XIE; ZHOU; GUAN	2015	n/i	RF, CB, SR	ATCGN[g]	ignora	não	n/i
iDoComp	OCHOA; HERNAEZ; WEISSMAN	2014	C	RF, PB	ASCII	sim	não	n/i
DNAC-K	TAN; SUN	2014	n/i	RF, CB, MR[h]	IUPAC	n/i	não	n/i
RLZ 1.4[i]	PROCHáZKA; HOLUB	2014	C++	RF, CB, SR	ATCGN	n/i	não	RA
FRESCO	WANDELT; LESER	2013	C++	RF, CB, MR[j]	ATCGN	n/i	não	n/i
DnaCompact	LI et al.	2013	C++	RF, PB	ASCII	sim	não	n/i
Dai *et al.*	DAI et al.	2013	C++	RF, PB	ASCII	n/i	não	n/i
ABRC	WANDELT; LESER	2012	C++	RF, PB	ASCII	ignora	sim	n/i
COMRAD	KURUPPU et al.	2012	C	DT, CB, MR	IUPAC	n/i	sim	RA
COOL	CHERN et al.	2012	Python	RF, PB	ASCII[k]	sim	não	n/i
GRS	WANG; ZHANG	2011	C/Shell	RF, PB	ASCII	sim	não	n/i
GReEn	PINHO; PRATAS; GARCIA	2011	C	ST, PB	ASCII	ignora	não	n/i
GDC-0.3	DEOROWICZ; GRABOWSKI	2011	C++	RF, CB, MR[l]	ASCII	yes	yes	RA[m]
RLZ-Opt	KURUPPU; PUGLISI; ZOBEL	2011	C	RF, CB, SR	ATCGN	n/i	não	n/i
RLZ RePair	KURUPPU; PUGLISI; ZOBEL	2011	C	RF, CB, SR	ATCGN	n/i	não	n/i
RLZ	KURUPPU; PUGLISI; ZOBEL	2010	C	RF, CB, SR	ATCGN	n/i	não	RA, SR
P. DNA Comp.	BRANDON; WALLACE; BALDI	2009	Perl	RF, CB, SR	ATCG	ignora	não	n/i

Esquema: RF=Referential, SR=Single-reference, MR=Multi-reference, CB=Collection-based, PB=Pair-based, ST=Statistical, DT=Dictionary. **Acesso:** RA=Random Access, SR=Single recovery.

Descrição das Colunas da Tabela 2.5:

- **Ferramenta** - Denota o nome oficial da ferramenta ou mecanismo de compressão utilizado. Se o nome não estiver disponível, a referência bibliográfica correspondente é fornecida.
- **Autor** - Declara o autor ou autores da ferramenta de compressão.
- **Ano** - Refere-se ao ano em que cada ferramenta foi publicada ou divulgada.

- **Linguagem** - Especifica a linguagem de programação predominante na qual a ferramenta foi desenvolvida.
- **Esquema** - Caracteriza o esquema de compressão adotado por cada ferramenta, proporcionando informações sobre o método de redução de dados.
- **Alfabeto** - Indica o conjunto mais amplo de símbolos distintos que cada ferramenta é capaz de suportar.
- **Cabeçalho** - Denota se o cabeçalho do arquivo *FASTA* é considerado no processo de compressão. Caso seja considerado, será restaurado ao seu estado original durante a descompressão.
- **Memória** - Destaca as ferramentas que possuem especificidades quanto ao uso de memória externa (como discos rígidos) durante os processos de compressão e descompressão.
- **Acesso** - Descreve as modalidades de acesso aos dados comprimidos sem necessidade de descompressão completa da sequência genômica.

Notas Complementares da Tabela 2.5:

a) Ferramenta codificada em MATLAB, mas faz uso de bibliotecas externas programadas em linguagem C.

b) Dado não disponibilizado pelos autores.

c) Para fins experimentais, uma versão alternativa foi desenvolvida em C++ pelos autores.

d) A representação alfabética das bases nitrogenadas é convertida uniformemente para letras minúsculas.

e) A despeito de sua capacidade multirreferencial, a ferramenta foi aplicada em uma configuração de referência única nos resultados apresentados para grandes volumes de dados.

f) A ferramenta opera exclusivamente com mais de uma sequência de referência no mapeamento de segunda ordem, sem restrições quanto ao número máximo de sequências de referência.

g) Todos os símbolos representando bases nitrogenadas são transmutados para minúsculas, e caracteres fora do alfabeto *ATCGN* são substituídos por *N*.

h) A ferramenta classifica sequências em grupos distintos e, para cada agrupamento, estabelece uma sequência de consenso, que é armazenada como uma representação resumida do conjunto.

i) A nomenclatura 'RLZ 1.4' foi adotada em razão da designação dada ao código-fonte pelos autores.

j) A ferramenta é limitada ao uso de múltiplas sequências de referência no mapeamento de segunda ordem, mas impõe uma restrição quanto ao número máximo dessas sequências.

k) Os símbolos representando bases nitrogenadas são uniformemente convertidos para letras maiúsculas.

l) O número máximo de sequências de referência admitido pela ferramenta é de 39.

m) O acesso aleatório aos dados comprimidos é inviabilizado quando se empregam múltiplas sequências de referência.

Nesta seção, a ênfase não recai sobre uma avaliação de desempenho comparativa das ferramentas de compressão genômica, conforme executado em estudos anteriores (NUMANAGIć et al., 2016; ZHU et al., 2015). Por exemplo, o trabalho apresentado em (NUMANAGIć et al., 2016) concentra-se exclusivamente em ferramentas de compressão de dados de Sequenciamento de Alto Desempenho (High-Throughput Sequencing (HTS)), enquanto o estudo em (ZHU et al., 2015)

limita-se a uma comparação entre duas ferramentas específicas de compressão vertical, nomeadamente GRS (WANG; ZHANG, 2011) e GReEn (PINHO; PRATAS; GARCIA, 2012). Ademais, o conjunto de dados de teste utilizado nestes estudos, embora diversificado em termos de espécies e dimensões genômicas, não fornece detalhamento suficiente acerca das características intrínsecas das sequências, como grau de similaridade, tipos de variação e a presença de polimorfismos de nucleotídeo único (*SNP*), inserções e deleções (*INDEL*) ou variações estruturais (*SV*).

Para além das distinções metodológicas nos estudos anteriores, foi também realizada uma análise para identificar quais ferramentas atualmente empregam esquemas específicos de compressão. O exame dos documentos culminou na identificação de três esquemas distintos de compressão, conforme detalhado na Tabela 2.6.

A compressão vertical fundamenta-se em uma estrutura bem-definida que consiste em um conjunto de sequências genômicas. Em contrapartida, a compressão referencial, também conhecida por compressão relativa, delta ou diferencial (GIANCARLO; ROMBO; UTRO, 2014), opera por meio da seleção de uma sequência genômica que atua como referência. As sequências remanescentes, referidas como sequências alvo, são comparadas com a sequência de referência a fim de localizar e mapear símbolos ou conjuntos de símbolos cujas coordenadas sejam dissonantes em relação à sequência de referência (ZHU et al., 2015). Estes símbolos e suas posições correspondentes são então substituídos, dentro das sequências alvo, por identificadores denominados correspondências, que fazem referência a segmentos específicos da sequência de referência (WANDELT; BUX; LESER, 2014). A Figura 2.6A exemplifica este tipo de compressão referencial. Este esquema de compressão é adotado por uma série de ferramentas, incluindo, mas não se limitando a, SCCG (SHI et al., 2018), HRCM, RCC, HiRGC, FM-context, MLF, RLZAP, NRGC, RCSCS, DNAComp, SLF, JDNA, Arram *et al.*, ERGC, GDC 2.0, CoGI, iDoComp, DNAC-K, RLZ 1.4, FRESCO, DnaCompact, Dai *et al.*, ABRC, COOL, GRS, GDC 0.3, RLZ-Opt, RLZ-RePair, RLZ e Project DNA Compression.

Na Figura 2.6A, com base em pares referenciais, a sequência alvo é comprimida usando informações extraídas da sequência de referência. O objetivo deste processo é encontrar segmentos compartilhados. O resultado é um mapa Mapped Target Sequence (MTS), em que as correspondências substituem os segmentos compartilhados. Além disso, os segmentos não compartilhados são

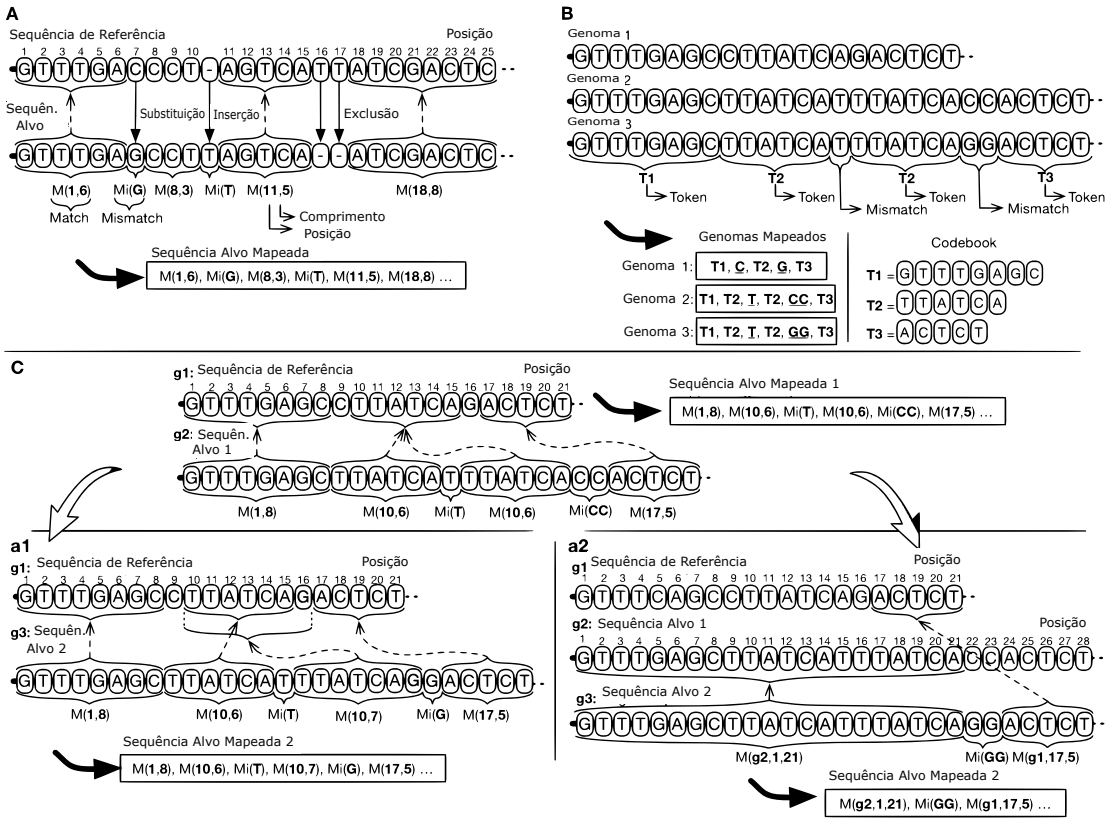

Figura 2.6: Esquemas de compressão. (A) Com base em pares referenciais, (B) Baseado em coleção de dicionários, e (C) Referência única/multi-referência baseada em coleção referencial – adaptada de (KREDENS et al., 2020)

gravados em formato bruto, como uma incompatibilidade. Ainda no exemplo (A), a correspondência é composta por dois inteiros, em que o primeiro representa a posição ou coordenada onde o segmento compartilhado começa na sequência de referência e o segundo representa o tamanho da correspondência. No exemplo da Figura 2.6B, com base em coleção de dicionários, três genomas são comprimidos ao mesmo tempo. No primeiro passo, o processo busca segmentos compartilhados e os armazena em uma estrutura chamada de *Codebook*. Em seguida, para obter os genomas mapeados, ele reescreve as sequências, substituindo segmentos por índices de *tokens* criados anteriormente. Índices de *token* e símbolos de incompatibilidade compõem o mapeamento final. Neste exemplo, a conversão de um segmento em um *token*, requer que o segmento compartilhado seja maior do que três símbolos —ou bases nitrogenadas. No exemplo da Figura 2.6C, com referência única/multi-referência baseada em coleção referencial, as sequências $g1$, $g2$ e $g3$ são comprimidas. Na primeira etapa, a sequência $g2$ ou

Sequência Alvo 1 (*Target Sequence 1*) é comprimida em relação a Sequência de Referência $g1$ e um mapeamento (*Mapped Target Sequence 1*) é gerado entre eles. A segunda etapa representa o comportamento ou referência única ($a1$) ou referência múltipla ($a2$). No caso de referência única ($a1$), o processo é semelhante ao passo anterior apresentado em C, em que a sequência $g3$ ou Sequência Alvo 2 (*Mapped Target Sequence 2*) é comprimida com base na sequência de referência $g1$. No caso de compressão com estratégia multi-referência ($c2$), para comprimir $g3$: Sequência Alvo 2 (*Target Sequence 2*), o processo de compressão tenta encontrar segmentos compartilhados usando tanto $g1$: Sequência de Referência (*Reference Sequence*) quanto aqueles já foram comprimidos na $g2$: Sequência Alvo 1 (*Target Sequence 1*). Assim, as correspondências podem estar em qualquer outra sequência da coleção.

A abordagem referencial apoia-se em uma estrutura de dicionário. A ideia básica deste esquema é encontrar porções de dados, ou seja, *substrings* repetidas ou recorrentes, removê-los do conteúdo principal e armazená-los como *tokens* indexados em um dicionário, chamado de *Codebook*. Essas partes de dados recorrentes são substituídas por seus índices (ou coordenadas) correspondentes e armazenadas no *Codebook* (ZIV; LEMPEL, 1978). A compressão se torna efetiva quando menos espaço é necessário para armazenar os índices do que o espaço dos *tokens* mapeados (LARSSON; MOFFAT, 2000). A Figura 2.6B ilustra como funciona um esquema de compressão baseado em dicionário. Neste esquema, a ferramenta percorre linearmente as três sequências genômicas de exemplo para detectar segmentos compartilhados entre elas. Os segmentos encontrados são então removidos da sequência e armazenados como *tokens* no *Codebook*. No exemplo, o *token* $T1$ é composto pelas bases $GTTTGAGC$ e é encontrado nas três sequências.

As ferramentas que implementam o conceito baseado em dicionário são COMRAD (KURUPPU et al., 2012) e Mehta et al (MEHTA; GHRERA, 2015). Embora Mehta e colaboradores tenham descrito sua ferramenta como um esquema baseado em referencial, deve-se notar que há uma etapa inicial para criar um conjunto de referências artificiais. Esse conjunto de referências é composto por blocos compartilhados extraídos de um conjunto de sequências de referência definidas aleatoriamente. Esses blocos, presentes em partes das sequências de referência, são então localizados nas sequências alvo para serem substituídos por itens correspondentes do mapeamento. Neste esquema baseado em dicionário,

o conjunto de referências é usado como *Codebook*, e os *tokens* correspondem aos índices;

Outra abordagem de compressão faz uso da probabilidade de um dado símbolo (base nitrogenada) ocorrer em uma sequência. Essa abordagem estatística, em vez de implementar uma fase de transformação dos dados, o esquema aplica diretamente a codificação sobre as sequências genômicas de entrada. Em seguida, ele pode determinar uma palavra-código única, que pode ser de tamanho variável (proporcional ao tamanho do alfabeto). Esta palavra-código faz parte do arquivo final comprimido (SALOMON; MOTTA, 2009). A ferramenta GReEn (PINHO; PRATAS; GARCIA, 2012) é baseada em um modelo de codificação aritmética (detalhado em (MCANLIS; HAECKY, 2016)), que usa a sequência de referência para criar um modelo probabilístico dinâmico. O modelo gera um intervalo numérico que é usado para comprimir as sequências de destino e para reconstruir a sequência original. A ferramenta GeCo (PRATAS; PINHO; FERREIRA, 2016a; PRATAS; HOSSEINI; PINHO, 2017) também usa um esquema de codificação aritmética. Seu conteúdo corresponde à probabilidade de ocorrência de cada símbolo, que é calculada por dois modelos, um baseado em Modelos de Contexto Finito (FCMs) e um segundo baseado em Modelos de Contexto Finito Estendido (XFCMs). Para as ferramentas GReEn e GeCo, os modelos são criados e atualizados usando informações extraídas da sequência de referência. Portanto, quanto melhor a capacidade do modelo predizer o próximo símbolo ocorrer na sequência, menor a quantidade de dados a serem armazenados (WANDELT; LESER, 2013).

Quando a abordagem de compressão é vertical, para que o processo de descompressão possa ser realizado, as ferramentas especializadas requerem a disponibilidade da sequência de referência ou do dicionário, no caso de ferramentas baseadas em dicionário. Assim, foi investigado neste trabalho o número de sequências genômicas envolvidas no processo de compressão, que pode ser baseado em dois métodos diferentes:

Baseado em pares: É um método de compressão que usa uma única sequência genômica alvo como dados de entrada a serem comprimidos (DEOROWICZ; DANEK; NIEMIEC, 2015). Quando um conjunto de sequências de destino está disponível, cada sequência é comprimida separadamente (DEOROWICZ; DANEK; NIEMIEC, 2015) usando uma sequência de referência que deve estar disponível para as outras sequências de des-

tino a serem comprimidas ou descomprimidas (GRUMBACH; TAHI, 1993; GRUMBACH; TAHI, 1994) (cf. Figura 2.6A);

Baseado em coleção: É um método de compressão que usa como dados de entrada uma coleção de sequências genômicas. O conjunto é analisado para encontrar segmentos compartilhados com apenas uma única sequência de referência (KURUPPU; PUGLISI; ZOBEL, 2010) (abordagem referencial), ou com uma coleção delas (abordagem multi-referência) (GIANCARLO; SCATURRO; UTRO, 2009) (cf. Figura 2.6C). Dependendo do número de sequências envolvidas, o método de compressão pode aplicar diferentes etapas para a compressão. Na primeira etapa (cf. Figura 2.6C), para ambas as abordagens, única e multi-referência, são realizados os mesmos procedimentos de busca por meio de segmentos compartilhados quando apenas há uma sequência de referência e uma sequência de destino (Sequência alvo 1). Na segunda etapa, o procedimento de busca por ocorrências compartilhadas torna-se diferente. Para uma compressão de referência única, uma segunda sequência alvo (cf. Figura 2.6Ca1, Sequência Alvo 2) é comprimida usando a mesma sequência de referência usada pela Sequência Alvo 1. A compressão multi-referência (cf. Figura 2.6Ca2, Sequência Alvo 2) usa as informações contidas em Sequência de Referência e Sequência Alvo 1. A Sequência Alvo 1 também é incorporada ao conjunto de sequências comprimidas e é usada como fonte de informação (espaço de busca). A possibilidade de utilizar mais sequências como referência, aumenta a probabilidade de encontrar mais segmentos compartilhados.

Entre as ferramentas avaliadas (cf. Tabela 2.5), apenas 6 delas, GDC 2.0, DNAComp, DNAC-K, FRESCO, COMRAD e GDC 0.3, implementam o método de compressão baseado em coleção usando a abordagem multi-referência. Resultados preliminares sugerem que esses 6 métodos são promissores, na medida em que eles mostraram resultados significativos quando comparados a outras técnicas (DEOROWICZ; DANEK; NIEMIEC, 2015; WANDELT; LESER, 2015). Um caso particular é a ferramenta RCC (CHENG; LAW; SIU, 2018), que agrupa as sequências alvo em grupos e, para cada grupo, gera uma sequência de referência artificial —ou sequência de consenso. Em seguida, cada sequência alvo é comprimida usando uma sequência de referência correspondente. Então, embora haja mais de uma sequência de referência, cada grupo usa apenas

uma única referência por vez, logo é considerada uma abordagem de referência única.

Outro ponto importante na compressão de dados genômicos está ligado à maneira como se processa e fornece acesso a uma representação comprimida de arquivo *FASTA*. A representação *FASTA* possui duas informações para cada sequência. A primeira informa a linha de dados de cabeçalho que identifica a sequência, começando com o símbolo > (maior que), e a segunda informação corresponde a linha do conteúdo, que são os símbolos que representam as bases nitrogenadas da sequência genômica propriamente dita. A representação *FASTA* não faz distinção entre maiúsculas e minúsculas, embora algumas ferramentas especializadas também tratem essas informações. Em um arquivo *FASTA* —com sequência genômica—, o cabeçalho tem um tamanho médio menor que 256 símbolos. Em outras palavras, o cabeçalho é extremamente pequeno em comparação ao conteúdo que representa a sequência de DNA. No entanto, o cabeçalho é necessário para recuperar as sequências comprimidas; deve-se notar que o cabeçalho da sequência pode ser armazenado separadamente sem a aplicação de um esquema de compressão particular.

Embora o cabeçalho seja parte essencial para a recuperação de dados comprimidos, apenas algumas ferramentas armazenam essa informação: HiRGC, NRGC, JDNA, ERGC, GDC 2.0, iDoComp, DNA_COMPACT, COOL, GDC 0.3 e GRS. Outras ferramentas investigadas descartam o cabeçalho: MLF, Arram *et al.* , GeCo, DNAComp, SLF, CoGI, ABRC, GReEn e Project DNA Compression. Outras ferramentas não mencionam como tratam o cabeçalho da sequência: RCC, FM-Context, DNAC-K, RLZ 1.4, FRESCO, Dai *et al.* , COMRAD, RLZ-opt, RLZ-RePair e RLZ. A heterogeneidade encontrada na forma como as ferramentas processam o cabeçalho da sequência torna um desafio avaliar com precisão o desempenho de uma ferramenta especializada. Por exemplo, nos testes descritos pela ferramenta iDoComp, os autores ignoraram o tamanho do cabeçalho da sequência para pequenos conjuntos de dados para tornar os resultados comparáveis com outras ferramentas.

Além do cabeçalho da sequência, outra característica essencial para a compressão de dados genéticos é o alfabeto usado para representar a sequência de DNA em um arquivo *FASTA*. Quatro símbolos compõem o menor alfabeto necessário para representar uma sequência de DNA: A, T, C e G. Dentre as 36 ferramentas avaliadas, 17 delas: MBGC (GRABOWSKI; KOWALSKI, 2022), HRCM, HiRGC, FM-context, MLF, NRGC (SAHA; RAJASEKARAN, 2016), SLF,

ERGC, GDC 2.0, iDoComp, DnaCompact, Dai *et al.* , ABRC, COOL, GRS, GReEn e GDC 0.3 não impõem nenhuma restrição alfabética, sendo, portanto, capaz de considerar qualquer símbolo da tabela ASCII e também fazer distinção entre maiúsculas e minúsculas. A única exceção ocorre para as ferramentas COOL e ABRC. Deve-se notar que COOL converte todos os símbolos para maiúsculas e a ferramenta ABRC não especificou se usa essas informações. Entre as outras ferramentas, 2 delas aceitam o alfabeto International Union of Pure and Applied Chemistry (IUPAC): COMRAD e DNAC-K. Outras 9 ferramentas consideram um alfabeto de tamanho reduzido e composto pelos símbolos ATCGN: RLZAP, GeCo, JDNA, CoGI, RLZ 1.4, FRESCO, RLZ-opt, RLZ-RePair e RLZ. Ainda, 5 ferramentas são restritas a apenas 4 símbolos correspondentes às bases nitrogenadas ATCG: RCC, Arram *et al.* , GeCo, DNAComp e Project DNA Compression.

Além da análise do tratamento do conteúdo do arquivo *FASTA*, também investigou-se as demandas computacionais de cada ferramenta especializada, tais como o uso de memória secundária (disco). Quando usada apropriadamente, a memória secundária fornece a ferramenta de compressão a capacidade de manipular grandes conjuntos de dados, sem restringir o trabalho de compressão/descompressão ao tamanho da memória principal (memória RAM). Em contrapartida, o uso desordenado pode forçar a ferramenta de compressão a usar a memória virtual do Sistema Operacional (swap) e reduzir o desempenho do processo de compressão/descompressão. Portanto, ao comprimir grandes conjuntos de dados em ambientes computacionais limitados, uma ferramenta de compressão deve autogerenciar o uso de recursos computacionais. Esta funcionalidade foi implementada em algumas das ferramentas avaliadas: GeCo, NRGC, JDNA, ERGC, ABRC e COMRAD. No GeCo, uma nova proposta foi implementada usando tabelas *hash*, chamadas de *cache-hash*, que não precisam ser armazenadas inteiramente na memória principal do sistema computacional. Em vez disso, a ferramenta mantém armazenadas na memória principal apenas as entradas mais recentes. NRGC, JDNA e ERGC implementam uma estratégia de carregamento das sequências genômicas de entrada sob demanda. Nesse método, o esquema de compressão mantém os dados restantes, que ainda não foram comprimidos, em um meio de armazenamento secundário. A ferramenta ABRC implementa uma abordagem ligeiramente diferente e carrega para o mapeamento na memória principal apenas as partes necessárias do índice. A ferramenta COMRAD usa uma abordagem semelhante; no entanto, ela processa cada sequência genômica de forma independente das outras durante o processo

de compressão. Essa estratégia reduz a quantidade total de memória principal necessária para compressão.

Analisou-se também como as ferramentas de compressão recuperam parte ou sequências inteiras de dados genômicos já comprimidos. A motivação pode ser recuperar uma sequência genômica proveniente de uma coleção de sequências em estado comprimido ou apenas um segmento de uma única sequência em estado comprimido. A recuperação pode exigir a descompressão de toda a coleção de sequências comprimidas —recuperação total— ou permitir que apenas a sequência ou trecho desejado da sequência sejam descomprimidos— recuperação única; o último caso é caracterizado por acesso não sequencial aos dados (GIANCARLO; ROMBO; UTRO, 2014). Entre as ferramentas de compressão baseadas em coleção, a funcionalidade de recuperação única está disponível em: GDC 2.0, COMRAD, RLZAP, RLZ 1.4, GDC 0.3 e RLZ.

A recuperação de uma só vez —*one-time recovery*—, ou seja, recuperar os dados de interesse utilizando apenas um passo, está presente na ferramenta GDC 2.0 quando apenas uma única sequência de referência foi usada na compressão; caso contrário, a ferramenta deve descomprimir mais de uma sequência. A ferramenta COMRAD também implementa a recuperação de uma única vez para extrair a sequência de interesse. No entanto, requer que todo o conteúdo do *Codebook*, além das sequências comprimidas, sejam armazenados na memória principal. Neste método, apenas os *tokens* necessários para descompressão da sequência terão seus símbolos correspondentes extraídos. As outras ferramentas como RLZAP, RLZ 1.4, GDC 0.3 e RLZ implementam extração de sequência parcial e total como forma de recuperação de uma única vez. Estruturas de dados baseadas em auto-índices é a abordagem mais amplamente utilizada para recuperação de uma única vez (PROCHáZKA; HOLUB, 2014; KREFT; NAVARRO, 2010; MäKINEN et al., 2010b).

2.6 Detecção de Segmentos Compartilhados e Codificação

As 36 ferramentas de compressão vertical de dados genômicos avaliadas são baseadas em três fases distintas (cf. Figura 2.7), cada uma composta por um número diferente de etapas (cf. Tabela 2.6). Essas ações são detalhadas nas seções a seguir.

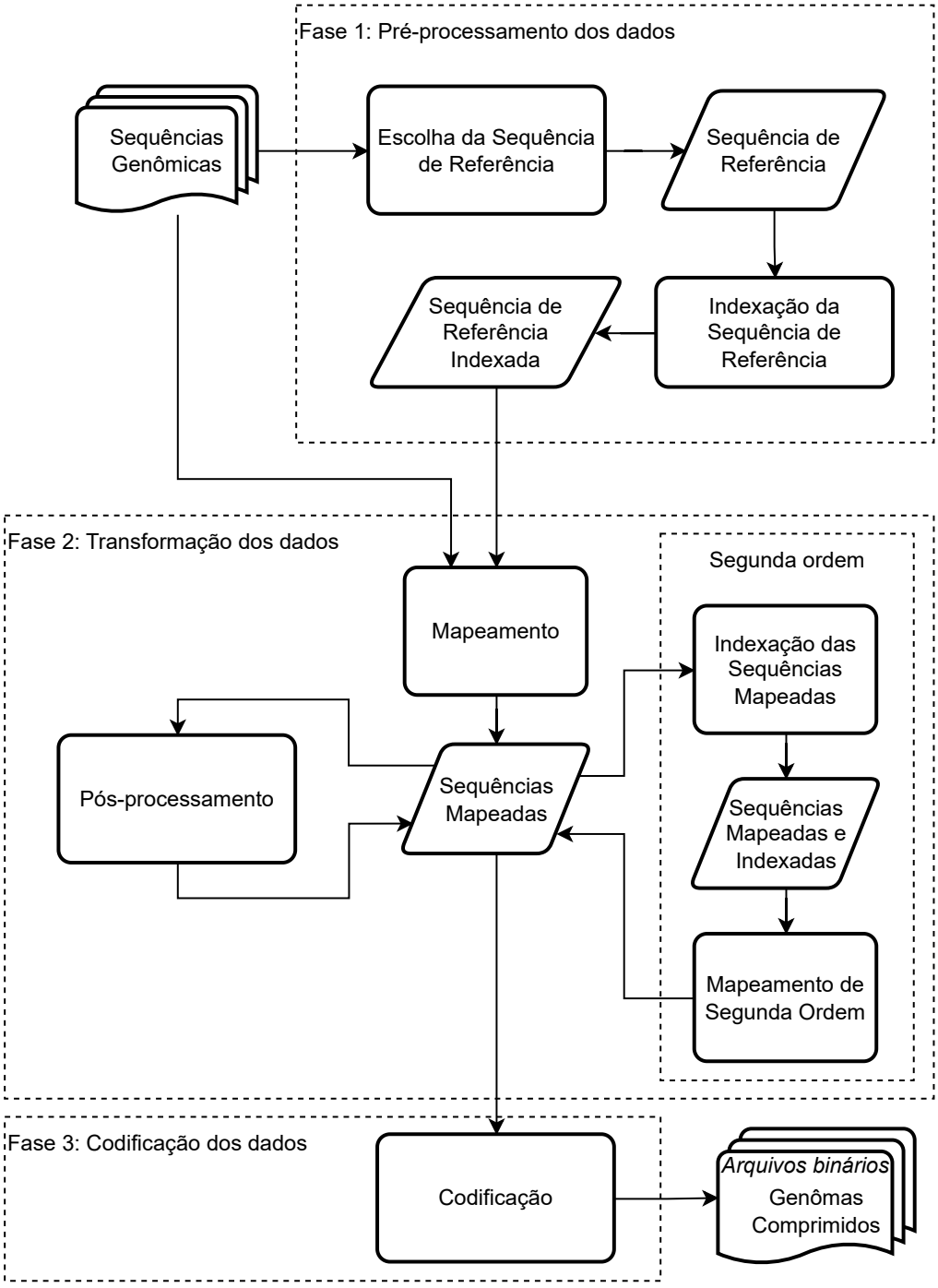

Figura 2.7: Fluxo conceitual de compressão – adaptada de (KREDENS et al., 2020)

A Figura 2.7 descreve o fluxo conceitual das atividades que podem ser executadas por uma ferramenta de compressão vertical para comprimir dados de sequências genômicas. São elas:

Fase de Pré-processamento: é a fase de seleção e indexação da sequência utilizada como referência. Algumas ferramentas não possuem um processo automático de seleção de referências. Neste caso, as ferramentas demandam que a referência seja informada manualmente.

Fase de transformação: é a fase em que as ferramentas exploram características dos dados inerentes às sequências genômicas que, por serem todas da mesma espécie ou correlatas, tendem a compartilhar parte significativa da informação. A primeira atividade é criar o mapeamento, que conterá correspondências, incompatibilidades ou editar operações entre uma determinada sequência alvo e a referência. Depois disso, as ferramentas podem executar o passo de **Pós-processamento** ou **Mapeamento de Segunda Ordem**.

Fase de Codificação: é a fase que recebe como entrada as sequências mapeadas para serem codificadas, ou seja, escritas de forma binária. O objetivo é reescrever a informação visando atingir a menor entropia possível. Nesta fase, algumas ferramentas agrupam os dados de acordo com a distribuição dos símbolos que a ferramenta irá codificar. Isso é feito porque alguns métodos de codificação de entropia têm sua eficiência diretamente relacionada ao número de símbolos distintos e à distribuição de probabilidade de cada símbolo. Por exemplo, as incompatibilidades podem ser codificadas separadamente das correspondências, que podem ter o valor referente à posição, codificado separadamente dos valores que se referem ao tamanho. O resultado final é um arquivo binário contendo uma ou mais sequências comprimidas.

Tabela 2.6: Fases executadas por ferramentas de compressão vertical – adaptada de (KREDENS et al., 2020)

TN	RSS	RSI	MP	MLS	SOM	PP	EC	RPR
MBGC	b	e	b	não	não	não	d[b]	a
memRGC	b	e	b	não	não	não	d[b]	a
HRCM	b	e	b	não	não	não	d[b]	a
SCCG	b	e	b	não	não	não	d[b]	a
RCC	a	n/i[a]	a	não	não	não	a,b	a
HiRGC	b	a ($k = 20$)	a	não	não	não	c,d[b]	a
FM-context	b	b	a	não	não	não	a,b	b
MLF	b	c	a	não	não	não	e,f[b],d[b]	b
RLZAP	b	d	a	sim	não	não	e,g[c]	a
NRGC	b	e ($k = 11, 12, 13$)	a	não	não	não	d[b]	a
GeCo	b	e ($cache - hash, k = n/a$)	não[d]	sim[e]	não	não	a	n/a[f]
RCSCS	b	n/i[v]	a	sim	não	não	h	n/i[v]
Arram *et al.*	b	b	a	não	não	sim	e[h]	b
DNAComp	b,c	n/i[a]	a	não	não	não	a,b,c	c
SLF	b	c	a	não	não	não	f[b]	b
JDNA	b	e ($k = n/a$)	a	sim	não	não	i[i]	a
ERGC	b	e ($k = 21, k = 9$)	a	sim	não	não	d[b]	a
Mehta *et al.*	a	e ($k = dynamic$)	c	não	não	não	n/i[v]	n/i[f]
GDC 2.0	b	e ($k = 15$)	a	sim	sim	não	j	d
CoGI	c[j]	e ($k = n/a$)	b	sim	não	não	k,l,m	a
iDoComp	b	f	a	não	não	sim	a	e
RLZ 1.4	b	b	a	não	não	não	e,g[c],c	f
FRESCO	a,c[n]	e ($k = 34$)	a	sim	sim	não	e	g
DnaCompact	b	não[o]	a	sim	não	não	a,b,n	h
Dai *et al.*	b	não	b[d]	sim	não	não	d[p],a	n/i[v]
ABRC	b	g	a	sim	não	não	l	i
COMRAD	n/a[m]	não	c	não	não	não	o	n/a[f]
COOL	b	não[q]	a	sim	não	sim	i,p	a
GRS	b	h[r]	b[v]	sim	não	não	i	a
GReEn	b	e ($k = 11$)	não[d]	sim[e]	não	não	a	n/a[f]
GDC 0.3	b,c	e ($k = len(\alpha\beta)\ k\ se\ k = 4069$)	a	sim	não	não	i,q	j
RLZ-opt	b	d,f	a	sim	não	não	p	k
RLZ-RePair	a	f	a	sim	não	não	p	k[u]
RLZ	b	f	a	não	não	não	r	b
P. DNA Comp.	a,b	não	n/i[v]	não	não	não	b[w],i,p	a

Notas complementares da Tabela 2.6:

a) Apesar da ferramenta não implementar uma estrutura de índice, ela utiliza uma ferramenta externa chamada PatterHunter para a busca dos segmentos compartilhados;

b) Usa a ferramenta de compressão genérica 7zip para comprimir o resultado;

c) Utiliza a Biblioteca de Estruturas de Dados Sucintas (SDSL) (GOG et al., 2014);

d) Envia cada símbolo da sequência diretamente para o codificador;

e) Apesar de não possuir fase de mapeamento, a ferramenta utiliza *Busca Local* sempre que houver uma nova tentativa de reiniciar o uso da probabilidade;

f) Não aplicável, não possui elemento de posição a ser armazenado;

g) Para a primeira correspondência, a posição absoluta é armazenada;

h) Usa inteiros, de comprimento variável, para codificar elementos de posição e tamanho de correspondências;

i) Usa a ferramenta de compressão de uso geral Gzip para comprimir o resultado final;

j) Por padrão, usa a sequência mais curta como referência e também possui duas heurísticas distintas para seleção automática de referência;

k) Usado para representar a posição dos SNPs;

l) Usado para representar o elemento de posição das correspondências;

m) Não aplicável, não utiliza sequência de referência;

n) Por padrão, utiliza a sequência mais longa e também possui uma heurística específica para seleção automática de referências;

o) Implementa uma janela de pesquisa bidirecional, portanto, não cria índice na sequência de referência;

p) Usa uma ferramenta de compressão genérica PPMDj para comprimir o resultado final;

q) Implementa uma janela dinâmica baseada na correspondência de *strings*, portanto, não cria índice na sequência de referência;

r) Usa o programa UNIX-diff modificado;

s) Embora não possua fase de mapeamento, a ferramenta cria índice para a sequência de referência;

t) O resultado de (posição da última correspondência + comp. da última correspondência + comp. dos fatores anteriores) é chamado de "posição esperada";

u) Quando isso acontece, significa que a posição pode ser derivada com base na última posição, para que o valor não seja armazenado. Um vetor de *bits* é usado;

v) Não informado ou não detalha como funciona;

w) Essas técnicas não foram combinadas, ao contrário, foram utilizadas separadamente para avaliar o desempenho de cada uma;

Explicação das colunas da Tabela 2.6:

Nome da Ferramenta (TN): Nome da ferramenta, ou cotação da mesma quando não há nome.

Seleção de Sequência de Referência (RSS): Indica como é feita a seleção da sequência de referência ou se deve ser informada manualmente pelo usuário:

a) Gerada;

b) Manual;

c) Automática.

Indexação de Sequência de Referência (RSI): Apresenta o método ou estrutura de dados usado por cada ferramenta para indexar a sequência de referência:

a) K-tuple Hash Table;

b) FM-index;

c) Longest Previous Factor;

d) Matching Statistics (LCP array);

e) k-mer Hash Table;

f) Suffix Array;

g) Compressed Suffix Tree;

h) Matrix Graph.

Mapeamento (MP): Lista qual esquema é implementado para mapear sequências de destino:

a) Fatoração;

b) Alinhamento da sequência por: c:consenso, d:diferença, v:variação;

c) Baseado em gramática.

Mapeamento de *Busca Local* (MLS): Indica quais ferramentas implementam estratégias de pesquisa local.

Mapeamento de segunda ordem (SOM): Indica quais ferramentas executam uma fase de Mapeamento de Segunda Ordem.

Pós-processamento (PP): Indica quais ferramentas executam a fase de pós-processamento.

Codificação (EC): Apresenta quais técnicas de codificação são utilizadas na fase de codificação:

a) Arithmetic Coding

b) Adaptive Elias Gamma Coding

c) Run Length Encoding

d) PPMD

e) AD-HOC Binary Coding

f) LZMA2

g) Compressed bit-vectors

h) Integer arithmetic coding (with encryption capability)

i) Huffman Coding

j) Range Coding

k) Rectangular Partition

l) AD-HOC Binary Coding (static entropy)

m) Variable length for the reference sequence

n) Log-Skewed Coding

o) Canonical Huffman Coding

p) Golomb Coding

q) Variable-length byte Coding

r) Compressed integer Set

Representações de posição relativa (RPR): Apresenta, quando aplicável, como cada ferramenta calcula os valores da posição dos símbolos nas correspondências relativas:

a) RPR = (posição da correspondência atual - posição da última correspondência).

b) Posição absoluta;

c) RPR = (posição da correspondência atual - tamanho do alvo);

d) RPR = (posição da última correspondência + comp. da última correspondência + comp. dos literais anteriores) - posição correspondência atual;

e) RPR = (posição da correspondência atual - posição da última correspondência) | posição da correspondência atual - (posição da última correspondência + comp. da última correspondência);

f) RPR = (início da posição atual na seq. alvo - posição correspondente atual na seq. de referência);

g) RPR = posição da correspondência atual - (posição da última correspondência + comp. da última correspondência + 1);

h) RPR = (posição da correspondência atual - posição inicial da janela de busca);

i) RPR = Posição absoluta juntamente com o respectivo número de bloco;

j) RPR = (Início da posição atual no alvo - posição correspondência atual na referência);

k) RPR = (Se t == p, então a p (posição) é descartada, se não, p = (posição atual - posição esperada), sendo t = (posição da última correspondência + comp. da última correspondência + comp. dos fatores anteriores) e p = (posição da correspondência atual)).

2.7 Sequência de referência

Deve-se notar que a sequência de referência pode ser selecionada automaticamente ou criada. Essa etapa de seleção ou criação visa determinar pelo menos

uma sequência que será usada como referência para a detecção de segmentos compartilhados nas sequências alvo da coleção. A seleção de referência pode ser feita por escolha automática de uma das sequências da coleção (cf. Figura 2.8B). As ferramentas que implementam esta estratégia são: DNAComp, CoGI, FRESCO, GDC 0.3. Outro método para determinar a referência é baseado na geração de uma sequência de referência artificial —uma referência de consenso baseada nas sequências alvo da coleção (cf. Figura 2.8A)— e é implementado pelas ferramentas: RCC, CoGI, Mehta *et al.* (MEHTA; GHRERA, 2015), FRESCO e Project DNA Compression. Para ambos os casos, os melhores resultados de compressão são alcançados quando maiores semelhanças são encontradas entre as sequências de referência e o restante das sequências alvo da coleção.

Levando em consideração a forma como as ferramentas especializadas de compressão selecionam a sequência de referência, identificou-se três maneiras distintas:

Manual: a sequência de referência é fornecida previamente pelo usuário antes do início da compressão;

Automático: a sequência de referência é selecionada automaticamente pela ferramenta especializada antes do início da compressão;

Gerado: a sequência de referência é criada automaticamente pela ferramenta especializada, ou seja, uma sequência artificial é gerada a partir das sequências de uma coleção. Este processo é executado antes ou durante a compressão.

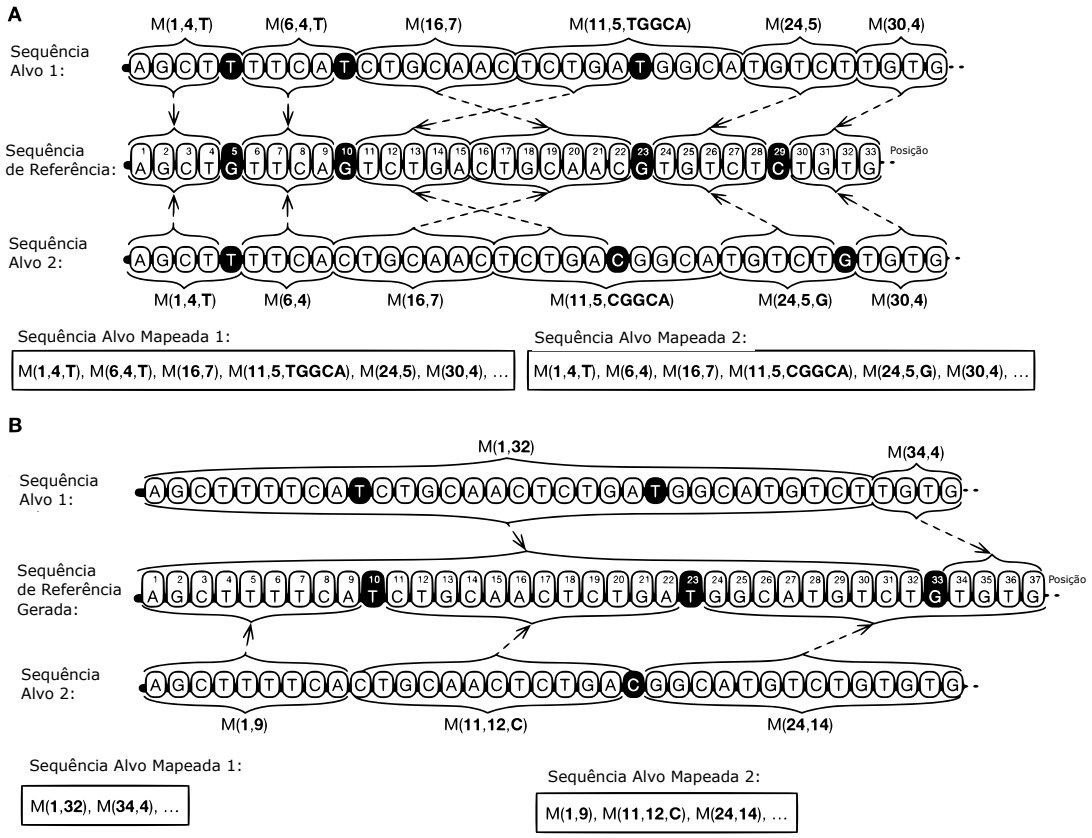

Figura 2.8: Sequência de referência (artificial vs. biológica) – adaptada de (KREDENS et al., 2020)

A Figura 2.8 ilustra as diferenças quanto ao uso de dados de referência com base em uma sequência genômica natural (cf. Figura 2.8B), em comparação com uma sequência gerada artificialmente (cf. Figura 2.8A). A grande vantagem de usar uma sequência artificial é sua alta similaridade com as sequências a serem comprimidas, na medida em que a sua criação é baseada nos segmentos mais frequentes encontrados dentro da coleção de sequências a serem comprimidas. No entanto, os usuários deste método devem considerar o alto custo computacional para criar e armazenar arranjos artificiais antes de aplicar tal abordagem.

Embora a seleção de uma referência tenha impacto direto e significativo no resultado do tamanho final das sequências comprimidas (WANDELT; LESER, 2013; BRANDON; WALLACE; BALDI, 2009; KURUPPU; PUGLISI; ZOBEL, 2011b), a maioria das ferramentas especializadas utiliza a seleção manual, conforme relatado por HiRGC, FM-context, MLF, RLZAP, NRGC, RCSCS, GeCo, SLF, Arram *et al.* , JDNA, ERGC, GDC 2.0, iDoComp, RLZ 1.4, DnaCompact, Dai *et al.* , ABRC, COOL, GRS, GReEn, RLZ-Opt, SCCG e RLZ. No entanto, a

seleção eficiente da sequência de referência permite minimizar a variação entre a sequência de referência e a sequência alvo, produzindo melhores resultados no tamanho total do arquivo de saída (BRANDON; WALLACE; BALDI, 2009); a seleção de referência de forma automática pode trazer benefícios significativos na taxa de compressão final.

2.8 Indexação de Sequência de Referência

Durante a fase de mapeamento (detecção de segmentos compartilhados entre sequências genômicas detalhadas na Seção 2.9), algumas ferramentas especializadas usam uma estratégia específica para acessar o conteúdo da sequência de referência. Uma dessas estratégias baseia-se na utilização de métodos de indexação para reduzir o custo computacional. Os métodos baseados em indexação usam estruturas de dados para acelerar a busca sobre o conteúdo armazenado. Estes métodos são baseados principalmente em matrizes de sufixos (*suffix arrays*) que permitem localizar um segmento de comprimento m em uma sequência de comprimento n com complexidade $O(m\ log\ n)$ em comparação com $O(n)$ sem seu uso. Em geral, quanto menor for o custo computacional para acessar um índice, maior será o custo em sua geração e armazenamento (CHENG; LAW; SIU, 2018).

Nesse contexto, MLF (BEAL ALIYA FARHEEN, 2016) e SLF (BEAL et al., 2016) usam matrizes de sufixo (*suffix arrays*) para criar duas listas de dados para busca de segmentos compartilhados. Uma lista encerra o fator anterior mais longo Longest Previous Factor (LPF) e a outra encerra a posição POS (*Position*). Em testes com o uso da ferramenta SLF, tem-se relatado que a mesma demorou 2376 segundos para construir o índice do genoma do *Homo sapiens* (versão KOREF_20090131 (AHN et al., 2009)). Para criar as listas LPF e POS foram necessários 399 segundos, resultando em 2775 segundos. Em contraste, a utilização dessas listas reduziram o tempo de compressão dos maiores cromossomos humanos para menos de um segundo. Por outro lado, a ferramenta MLF, que usa uma versão melhorada das matrizes de sufixo, demorou menos tempo do que a SLF, ou seja, 889 segundos para criar o índice do mesmo genoma, mas demorou cerca de 39 segundos para realizar a compressão da mesma sequência.

Este cenário exemplifica a correlação direta entre o tempo gasto para a etapa de indexação da referência com o desempenho da ferramenta de compressão

de dados. Os autores da ferramenta FRESCO descreveram uma análise semelhante. Eles identificaram que, usando a estrutura de índice baseada em uma tabela *hash k-mer*, o consumo de memória foi em média entre 8 a 10 vezes o tamanho da sequência de referência. Usando uma estrutura de índice baseada em árvores de sufixos comprimida (*compressed suffix trees*), a demanda de memória foi reduzida em até 5 vezes, mas com um tempo de processamento foi 30% maior do que quando foi usada uma estrutura de tabela *hash k-mer*. Uma vez mais, tal observação torna evidente que o uso de uma estrutura de índice pode aumentar substancialmente o desempenho final da ferramenta especializada de compressão.

Com base nesses problemas de desempenho, algumas ferramentas também implementam heurísticas específicas para contornar o aumento do custo computacional na criação do índice. A ferramenta JDNA implementa uma estratégia que cria um índice sob demanda, o que evita criar um índice de toda a sequência (cf. Tabela 2.6). JDNA só precisa indexar em média 2,5% das sequências, o que reduz a memória necessária para o índice. Da mesma forma, outras ferramentas como ABRC e ERGC implementam uma estratégia que divide a sequência de referência em blocos e então inicia o processo de indexação, bloco por bloco, que são excluídos após o uso.

Entre todas as abordagens analisadas, as ferramentas COOL e DnaCompact são as únicas que não requerem a criação de índice. Elas operam sobre a sequência de referência limitando o espaço de busca a uma janela de tamanho fixo em substituição do índice.

A ferramenta HiRGC (LIU et al., 2017) apresenta uma nova abordagem em que o processo de mapeamento recebe como entrada sequências baseadas em *2-bits* inteiros. Com isso, ela não precisa lidar com grandes alfabetos. Para conseguir isso, todos os símbolos do *Alfabeto Genômico diferente de {A, T, C, G}* são removidos antes de indexar a sequência de referência e, em seguida, a sequência é codificada em uma sequência de *2-bits* inteiros ($A = 0,\ C = 1,\ G = 2$ *e* $T = 3$). Em seguida, uma tabela *hash* de *k-tupla* (*k-tuple hash table*) é criada com base nos valores das tuplas, de tamanho k, da sequência de referência. O valor numérico de uma *k-tupla* é calculado com base em uma fórmula simples criada pelos autores. Um processo semelhante é aplicado à sequência alvo. Por fim, todos os símbolos do *Alfabeto Genômico diferente de {A, T, C, G}* removidos pelas ferramentas são armazenados junto com suas respectivas posições. Também são mantidas as coordenadas dos caracteres minúsculos. Essas informações são

usadas no processo de descompressão de modo que o arquivo descomprimido final tenha exatamente o mesmo conteúdo que tinha antes da compressão.

Conforme discutido até aqui, diferentes abordagens foram usadas para lidar com a indexação de sequência de referência para reduzir os problemas de memória. Nas ferramentas avaliadas, foram identificados 6 tipos de estruturas de índices: tabela *hash k-mer* (*k-mer hash table*), tabela *hash k-tupla* (*k-tuple hash table*), matriz de sufixo (*suffix array*) (GUSFIELD, 1997), índice FM (*FM-index*) (FERRAGINA et al., 2004), árvore de sufixo comprimida (*compressed suffix tree*) (OHLEBUSCH; FISCHER; GOG, 2010), grafo de matriz (*Matrix graph*) (MYERS, 1986) e fator anterior mais longo (*Longest Previous Factor*) (CROCHEMORE; ILIE, 2008).

A escolha do tipo de estrutura de índice impacta diretamente no tempo de processamento e no espaço de memória necessário para o processamento dos dados. A estrutura da tabela *hash k-mer* apresentou os melhores resultados em comparação com as outras formas de índice (DEOROWICZ; GRABOWSKI, 2011). Outros arranjos para indexação, como aqueles baseados no índice FM, também mostraram vantagens em comparação com as tabelas *hash k-mer*. Por exemplo, o desempenho do mapeamento não é influenciado pelo tamanho da sequência de referência. Dado que a referência raramente é alterada, pode-se manter armazenado o índice para comprimir outras sequências alvo (ARRAM et al., 2015). Apesar da importância do índice na qualidade final da compressão, deve-se considerar sua demanda em termos de custo de armazenamento. Em alguns casos, como na ferramenta em Arram *et al.* (ARRAM et al., 2015), o tamanho do índice gerado para o genoma humano é de 17 *Gibibytes*.

Dentre as estratégias de geração de índices analisadas, implementadas pelas ferramentas de compressão especializadas, o uso de tabela *hash k-mer* (ou k-tupla) é o tipo mais frequente de estrutura de dados utilizada. Essa estrutura é relatada por 11 das 36 ferramentas avaliadas (HiRGC, NRGC, GeCo, JDNA, ERGC, Mehta *et al.* , GDC 2.0 (DEOROWICZ; DANEK; NIEMIEC, 2015), CoGI, FRESCO (WANDELT; LESER, 2013), GReEn e GDC 0.3). Uma tabela *hash* armazena chaves e valores (KNUTH, 1998). Quando uma tabela *hash k-mer* é usada como uma estrutura de índice, é atribuído para a variável k o valor correspondente ao tamanho do segmento na sequência alvo (cf. Tabela 2.6). Em relação à importância do k, sabe-se que quanto menor o valor, maior o tempo necessário para a geração do índice e seu tamanho correspondente, sem causar qualquer diferença na taxa de compressão final (WANDELT; LESER, 2013). Apesar do alto

custo computacional envolvido na geração do índice da sequência de referência, o índice pode melhorar o desempenho final das ferramentas especializadas de compressão de dados genômicos.

2.9 Mapeamento de Primeira Ordem

É uma etapa de transformação dos dados responsável por mais de 70% do tempo de processamento (ARRAM et al., 2015). Na compressão de dados, a fase de transformação visa reduzir a redundância de informações e o número de elementos exclusivos que precisam ser armazenados. Com isso, a desordem dos dados é reduzida, diminuindo sua entropia, o que permite aumentar sua compressibilidade na fase de codificação. É por meio da transformação que se reduz o espaço necessário para armazenamento em uma magnitude menor do que o previsto pela Teoria da Informação (MCANLIS; HAECKY, 2016). Essa ideia foi explorada inicialmente por transformações LZ (ZIV; LEMPEL, 1978; ZIV; LEMPEL, 1977). Antes, a fase de codificação era alimentada com o conteúdo original dos dados a serem comprimidos, limitando-se à distribuição de cada elemento que compõe os dados de entrada.

Para alcançar maior eficiência, a etapa de transformação também deve ser capaz de explorar características inerentes aos dados que estão sendo processados; neste caso, se as sequências genômicas consideradas são resultantes de organismos biológicos semelhantes, da mesma espécie. No contexto da compressão vertical de dados genômicos, busca-se reduzir o número de segmentos de símbolos que representam as bases nitrogenadas que são iguais e compartilhados entre as sequências genômicas alvo. Nas ferramentas analisadas, foram identificadas as seguintes técnicas de mapeamento:

Fatoração: O processo de fatoração, baseado no conceito da ferramenta LZ77 (ZIV; LEMPEL, 1977), tenta emparelhar o segmento atual com uma ocorrência anterior de si mesmo. A implementação desta estratégia requer uma estrutura de dicionário, composta apenas por trechos da sequência já percorridos no fluxo de entrada. Na compressão vertical, ao invés de usar segmentos já visitados do fluxo de entrada como dicionário, a busca por segmentos repetitivos fica restrita a outras sequências disponíveis como referências (KURUPPU; PUGLISI; ZOBEL, 2010). Um caso particular são as ferramentas RCC, GeCo (PRATAS; PINHO; FERREIRA, 2016a), SLF

(BEAL et al., 2016) e DNAComp (CHENG et al., 2015) que incluem o fluxo de entrada (a sequência alvo) no espaço de busca. Desta forma, ao invés de utilizar apenas a sequência de referência como espaço de busca, também utiliza informações da sequência alvo. Como resultado deste processo, são geradas correspondências, que também são conhecidas como: fatores (KURUPPU; PUGLISI; ZOBEL, 2010; KURUPPU; PUGLISI; ZOBEL, 2011a; CHERN et al., 2012), LZ-correspondências (DEOROWICZ; GRABOWSKI, 2011), entrada de correspondência relativa (WANDELT; LESER, 2012), entradas de correspondência referencial (WANDELT; LESER, 2013) ou triploide (FAN et al., 2017). Essas correspondências representam segmentos compartilhados entre as sequências e podem ser intercaladas com segmentos de símbolos de uma ou mais bases nitrogenadas não compartilhados entre as sequências, conhecidas como bases de incompatibilidade. As correspondências podem ser representadas como: a) par (p, l), onde p é a posição inicial da correspondência na sequência de referência e l é o tamanho da correspondência; ou b) tripla (p, l, mi), em que p é a posição inicial da correspondência, l é o tamanho da correspondência e mi são as bases da incompatibilidade. Existem várias maneiras de realizar a fatoração e podem ser baseadas no conceito de janela deslizante (*sliding window*) (COOL e DnaCompact (CHERN et al., 2012; LI et al., 2013)), *alignment* (ERGC (SAHA; RAJASEKARAN, 2015)), *greedy search* (HiRGC (LIU et al., 2017), *FM-context* (FAN et al., 2017), FRESCO e RLZ (WANDELT; LESER, 2013; KURUPPU; PUGLISI; ZOBEL, 2010)), *look ahead* (RLZ-Opt (KURUPPU; PUGLISI; ZOBEL, 2011a)) ou *fingerprinting* (NRGC (SAHA; RAJASEKARAN, 2016)).

Com base na gramática: As sequências são processadas iterativamente para que, a cada passo, sejam identificados os segmentos frequentes que são substituídos por símbolos não terminais e que não fazem parte do alfabeto inicial do conteúdo de processamento. O objetivo é que seja possível identificar segmentos cada vez mais longos em cada estágio. Esses segmentos também podem ser compostos de símbolos não terminais. O processo termina quando um número predeterminado de etapas é alcançado ou quando segmentos suficientes não são mais encontrados para serem substituídos (KURUPPU et al., 2012). Dentre as ferramentas avaliadas, apenas a COMRAD atua dessa forma.

Alinhamento: É quando as sequências genômicas são alinhadas com o objetivo de encontrar segmentos compartilhados entre elas. O alinhamento pode ser feito por pares, de forma que uma sequência alvo seja o par de uma sequência de referência, objetivando encontrar a sequência local compartilhada e então extrair a sequência diferente (implementada pela ferramenta GRS (WANG; ZHANG, 2011)). Outro método de busca, implementado pela ferramenta DNAC-K (TAN; SUN, 2014), é alinhar as sequências para encontrar a sequência consenso que irá descrever o grupo de sequências alvo.

Com o resultado do processo de mapeamento de primeira ordem é gerado um mapa que contém informações sobre segmentos compartilhados e não compartilhados (cf. Figura 2.6). As únicas ferramentas que não implementam essa estratégia são: GeCo (PRATAS; PINHO; FERREIRA, 2016b) e GReEn (PINHO; PRATAS; GARCIA, 2012). Elas não possuem uma fase de mapeamento e ambas são ferramentas estatísticas, que enviam o conteúdo da sequência alvo diretamente para a codificação.

Foi identificado que, durante o mapeamento, algumas ferramentas implementam estratégias específicas para realizar a busca por segmentos compartilhados, denominada *Busca Local*. O objetivo é aumentar a eficiência na busca por trechos compartilhados ou identificar correspondências que tenham uma representação mais eficiente e que será explorada na fase de codificação. Ao buscar o início da próxima correspondência nas sequências de referência, a *Busca Local* é feita com a delimitação deste espaço de busca. Dessa forma, a busca é iniciada em uma posição subsequente próxima da última correspondência identificada. Então, a necessidade de percorrer toda a sequência de referência sempre que for necessário buscar uma correspondência é reduzida. Adicionalmente, as correspondências identificadas em uma cadeia crescente são priorizadas, resultando em uma representação relativa das posições. Tal comportamento é possível dada a homologia e colinearidade dos elementos genômicos, tais como genes, elementos reguladores e sequências repetitivas, existentes entre organismos biológicos da mesma espécie ou entre organismos biológicos de espécies geneticamente semelhantes. Isso reduz a chance de que a busca em um longo trecho de símbolos correspondentes seja interrompida por mutações como SNPs, INDELs, rearranjos e CNV (WOLFE et al., 2015).

Existem várias abordagens para a implementação do comportamento de *Busca Local* (cf. Figura 2.9), cujos principais estão sendo destacados aqui:

a) Manter um marcador de posição entre a referência e o alvo (cf. Figura 2.9A), e simultaneamente avançar para cada correspondência ou mutação identificada, conforme implementado pela ferramenta JDNA (ALVES et al., 2015). Outro comportamento, implementado pela ferramenta GDC 2.0, é realizar algumas verificações pontuais antes de iniciar a busca por um segmento mais extenso, avançando o marcador de posição na referência (HOSSEINI; PRATAS; PINHO, 2016);

b) Restringir a busca dentro de uma janela bidirecional (cf. Figura 2.9), que pode ter sua posição e comprimento atualizados dinamicamente. Esse comportamento é implementado pelas ferramentas COOL e DnaCompact (CHERN et al., 2012; LI et al., 2013);

c) Dividir a referência e o alvo em blocos do mesmo tamanho e processar os blocos na mesma ordem em que são lidos (cf. Figura 2.9), dando preferência inicialmente as correspondências nestes blocos. As ferramentas ABRC e ERGC (WANDELT; LESER, 2012; SAHA; RAJASEKARAN, 2015) implementam esse comportamento;

d) Utilizar uma métrica que penaliza as correspondências distantes da última correspondência considerada (cf. Figura 2.9), implementado pela ferramenta GDC-0.3 (DEOROWICZ; GRABOWSKI, 2011);

e) Iniciar a busca pela próxima correspondência mais longa após o término da última correspondência identificada (cf. Figura 2.9), implementado pelas ferramentas GeCo (PRATAS; PINHO; FERREIRA, 2016a), GReEn (PINHO; PRATAS; GARCIA, 2012), RLZ-Opt (KURUPPU; PUGLISI; ZOBEL, 2011a), RLZAP (COX et al., 2016). No entanto, Dai *et al.* (DAI et al., 2013) e CoGI (XIE; ZHOU; GUAN, 2015) restringem a busca a uma região próxima na qual estas ferramentas identificaram segmentos recentes.

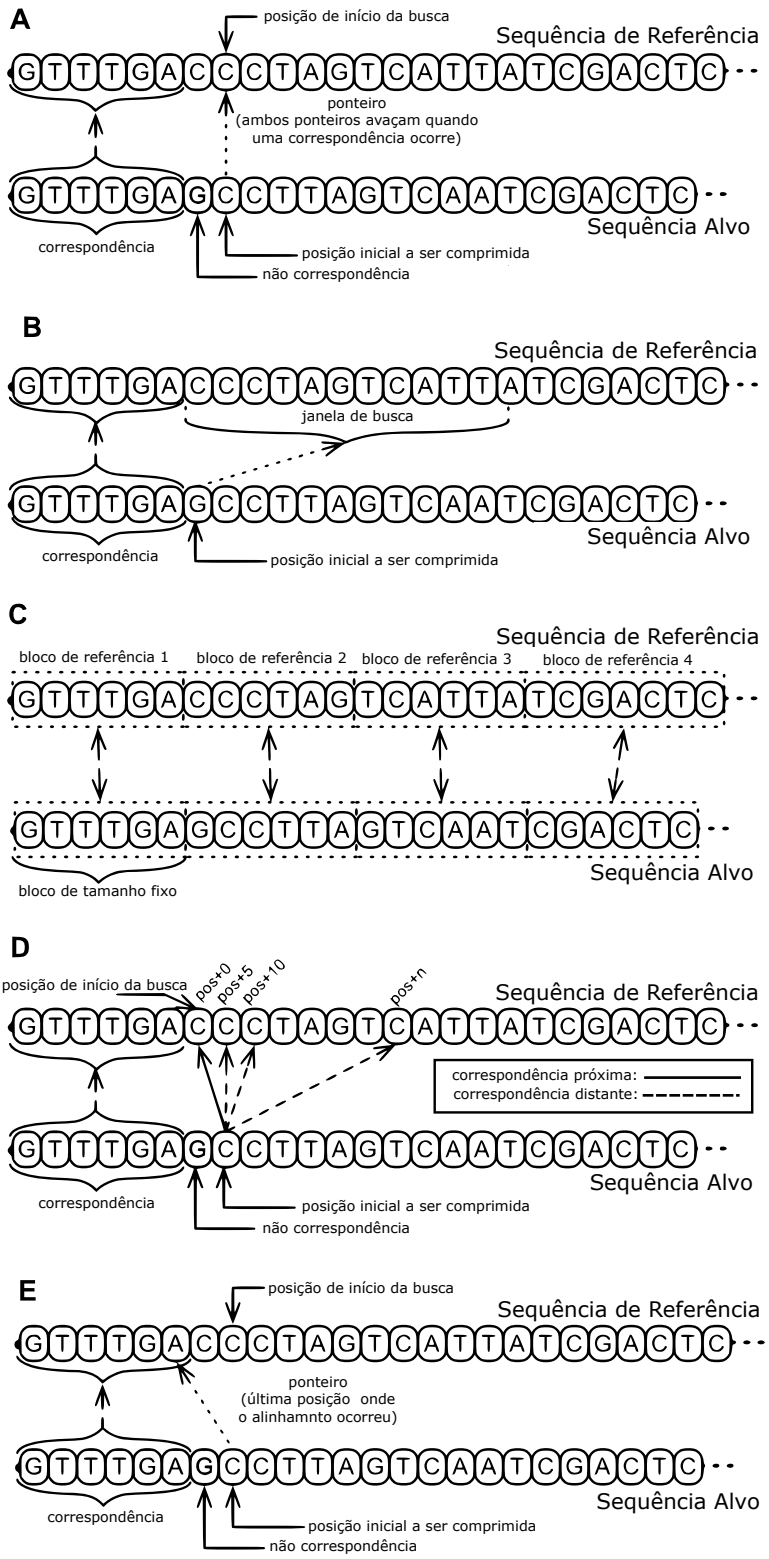

Figura 2.9: Busca Local – adaptada de (KREDENS et al., 2020)

A Figura 2.9 apresenta exemplos ilustrativos de como as ferramentas podem implementar o comportamento de *Busca Local* . Em Figura 2.9A, após identificar uma correspondência ou incompatibilidade, o ponteiro avança, ao mesmo tempo, tanto na sequência de referência quanto na sequência de destino, e então a busca é iniciada. Em Figura 2.9B, a busca é limitada a uma janela que avança sobre cada correspondência ou incompatibilidade. Em Figura 2.9C, a sequência de referência e a sequência de destino são divididas em blocos, sendo a busca de correspondências restrita, no primeiro momento, a tais blocos. Em Figura 2.9D, a partir da última correspondência, a ferramenta pode pesquisar em toda a sequência de referência, mas são aplicadas penalidades para as possíveis correspondências identificadas em posições distantes. Com isso, a ferramenta prioriza correspondências mais próximas às identificadas recentemente. E finalmente, em Figura 2.9E, a busca é sempre iniciada logo após o término da última correspondência encontrada.

Deve-se notar que a decisão de implementar a *Busca Local* depende diretamente do esquema de codificação implementado. Dentre as ferramentas avaliadas, foi identificado em 16 delas o método de *Busca Local* : RLZAP, GeCo, JDNA, ERGC, GDC 2.0, CoGI, DnaCompact, Dai *et al.* , ABRC, FRESCO, COOL, GRS, GReEn, GDC-0.3, RLZ-opt e RLZ-RePair. Algumas ferramentas são baseadas na identificação de correspondências próximas, como RLZ-opt (KURUPPU; PUGLISI; ZOBEL, 2011a) e GDC-0.3 (DEOROWICZ; GRABOWSKI, 2011), visando reduzir o custo de armazenamento da posição de correspondência. Outras ferramentas, como HiRGC (LIU et al., 2017), FM-context (FAN et al., 2017), RLZ (KURUPPU; PUGLISI; ZOBEL, 2010) e FRESCO (WANDELT; LESER, 2013), sugerem que, considerando correspondências mais longas, mesmo que distantes de correspondências anteriores, há uma redução na quantidade final de correspondências que, por conseqüência, reduz o custo final de armazenamento, pois há menos elementos que precisam ser armazenados. A ferramenta FRESCO implementa ambas as possibilidades; a *Busca Local*, que procura correspondências curtas e locais e a *Busca Gulosa* que procura as correspondências mais longas possíveis. Deve-se salientar que algumas ferramentas como RLZ-Opt (KURUPPU; PUGLISI; ZOBEL, 2011a) e JDNA (ALVES et al., 2015) implementam mais de uma estratégia de *Busca Local*, na qual é dinamicamente determinado durante o mapeamento, quais estratégias de *Busca Local* serão aplicadas. As ferramentas GeCo e GReEn, apesar de não executarem uma fase de mapeamento, utilizam

estratégias de *Busca Local* para criar e manter seus modelos estatísticos atualizados.

Outro problema fundamental da compressão vertical de sequências genômicas é o armazenamento dos elementos referentes à posição e comprimento dos segmentos compartilhados (BRANDON; WALLACE; BALDI, 2009). A abordagem primária é armazenar a posição em valor absoluto, apontando para a posição de referência na qual o segmento compartilhado é iniciado. No entanto, uma sequência genética pode variar de tamanho e pode conter de milhões a bilhões de símbolos que representam as bases nitrogenadas. Quanto mais próximo do final desta sequência, maiores serão os valores numéricos para representar a posição. Uma solução possível é armazenar o valor da posição atual com base na diferença (delta) da posição da correspondência anterior. Tal estratégia é possível quando as sequências genômicas alvo estão evolutivamente relacionadas. Assim, embora seja interrompido por mutações, a disposição dos elementos tende a ser colinear (WOLFE et al., 2015). Assim, a probabilidade de que segmentos compartilhados sejam encontrados de forma organizada é alta, de modo que a posição em que ocorrem forme uma cadeia organizada. Isso é exemplificado na Figura 2.6Ca1, onde quatro correspondências são apresentadas: $M(1,8)$, $M(10,6)$, $M(10,7)$ e $M(17,5)$. As correspondências, localizadas nas posições 1, 10 e 17 da sequência de referência, são dispostas para criar uma cadeia incremental e linear. As abordagens para a representação das posições identificadas estão dispostas na Tabela 2.6. Há um caso particular em relação à ferramenta FM-context (FAN et al., 2017), em que é armazenado o elemento de tamanho como resultado do tamanho da correspondência menos o valor do parâmetro $minlen$ (comprimento mínimo). O parâmetro $minlen$ é utilizado para limitar o tamanho mínimo das correspondências durante a etapa de mapeamento. Ao fazer isso, a ferramenta obtém um código em *bits* mais curto para o elemento de tamanho.

Na etapa de mapeamento, as ferramentas podem pesquisar correspondências exatas, correspondências complementares reversas (KURUPPU et al., 2012) ou palíndromos (CHENG et al., 2015). No entanto, é conhecido que as ferramentas de compressão horizontal de dados genômicos exploram esses tipos de correspondências (BEHZADI; Le Fessant, 2005; GRUMBACH; TAHI, 1993). Mesmo a ferramenta DnaCompact (LI et al., 2013), que é uma ferramenta de compressão vertical, mas que, opcionalmente, opera em modo horizontal, du-

rante a fase de mapeamento em seu modo horizontal, busca correspondências considerando complementos reversos e palíndromos.

Outra característica identificada em algumas ferramentas foi a implementação de uma etapa para identificar e tratar bases ambíguas. Esse recurso é relevante porque as tecnologias atuais usadas para sequenciamento de genoma têm algumas limitações computacionais que tornam a reconstrução da molécula original, em um único *contig*, um desafio significativo (SAMEITH; ROSCITO; HILLER, 2017). Isso se deve principalmente à existência de regiões no genoma com muitas repetições ou áreas com baixa cobertura de sequenciamento. É o caso, por exemplo, das regiões do centrômero e telômeros de um cromossomo. O resultado é que as bases nitrogenadas dessas regiões são representadas, na sequência final, pelo símbolo N. Como é esperado haver longos segmentos compostos por Ns, as ferramentas HiRGC (LIU et al., 2017), RLZAP (COX et al., 2016), JDNA (ALVES et al., 2015), GDC 2.0 (DEOROWICZ; DANEK; NIEMIEC, 2015) e GDC 0.3 (DEOROWICZ; GRABOWSKI, 2011) implementam comportamentos específicos para lidar com isso. Mesmo com o recente desenvolvimento de novos softwares de montagem e tecnologias mais modernas de sequenciamento genético, como o Nanopore (BROWN; CLARKE, 2016), ainda é necessário identificar bases ambíguas.

2.10 Mapeamento de Segunda Ordem

É uma etapa de transformação vertical dos dados que adota uma abordagem semelhante à etapa de Mapeamento de Primeira Ordem, identificada nas ferramentas: GDC 2.0 e FRESCO. Os ganhos na taxa de compressão são obtidos porque, ao comprimir sequências de DNA de organismos da mesma espécie, é alta a possibilidade de que as mesmas cadeias de correspondências e incompatibilidades ocorram entre sequências alvo distintas (DEOROWICZ; DANEK; NIEMIEC, 2015). Além disso, a possibilidade de correspondência se torna proporcionalmente maior à medida que aumenta o número de sequências na coleção (WANDELT; LESER, 2013; DEOROWICZ; DANEK; NIEMIEC, 2015). Dessa forma, o espaço de busca é composto pelo conjunto de sequências alvo já mapeadas, e o alfabeto considerado corresponde ao mapa de correspondências e incompatibilidades (cf. Figura 2.10).

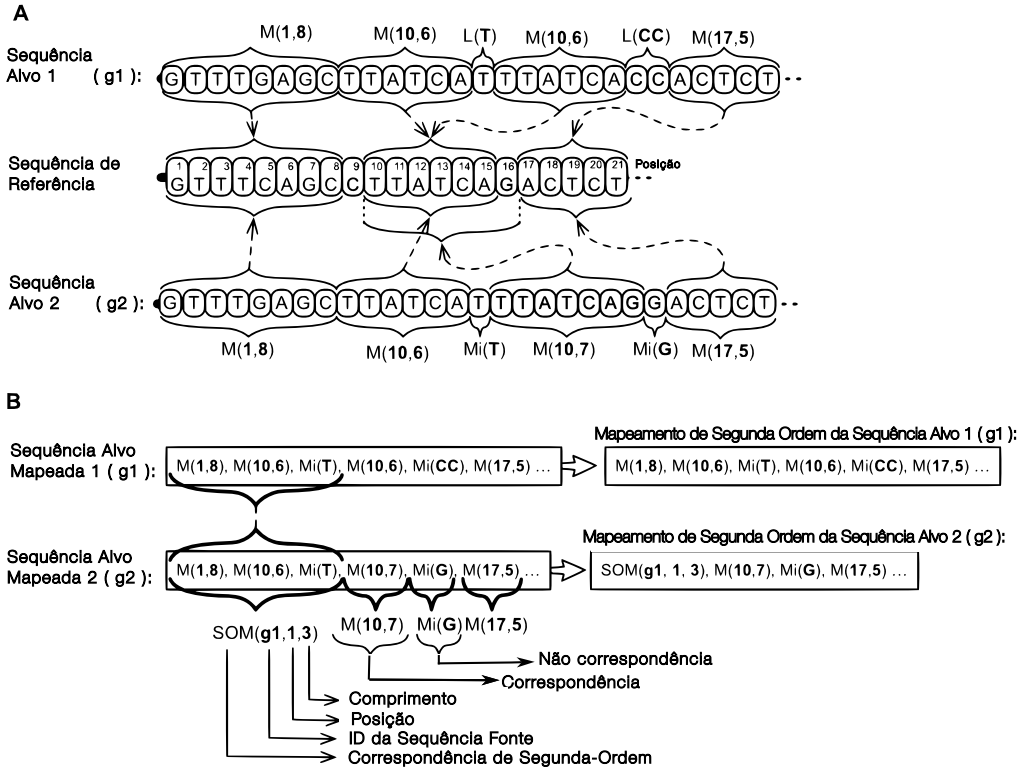

Figura 2.10: Mapeamento de segunda ordem – adaptada de (KREDENS et al., 2020)

A Figura 2.10 exemplifica duas sequências alvo, *Sequência Alvo* 1 e *Sequência Alvo* 2, que são analisadas usando a mesma *Sequência de Referência*. Como resultado, dois mapeamentos são obtidos, um para cada sequência. Então, o Mapeamento de Segunda Ordem Second Order Mapping (SOM) é executado em um segundo momento Figura 2.10B e as entradas são os mapas gerados anteriormente. A ferramenta analisa o mapa da *Sequência Alvo* 2 em relação ao mapa da *Sequência Alvo* 1 e identifica uma sucessão de três elementos, duas correspondências e um literal que são compartilhados, e gera o elemento $SOM(g1, 1, 3)$ como elemento de Correspondência de SOM. *SOM* indica que, na *Sequência Alvo* 2, partindo da posição 1 da *sequência* $g1$, devem ser considerados os três primeiros elementos, neste caso, $M(1, 8)$, $M(10, 6)$ e $L(T)$.

Na ferramenta FRESCO o SOM é feito com a divisão do conjunto de sequências alvo em dois subconjuntos. O primeiro, que será o espaço de busca para o SOM, é composto pelas novas sequências de referência indexadas (*hash table*). Portanto, o espaço de busca tem um tamanho máximo. O segundo conjunto é composto pelas demais sequências que são processadas, uma a uma, verificando

o espaço de busca por sucessões de correspondências ou não-correspondências coincidentes a serem substituídas por novas correspondências. Nas avaliações de desempenho, os autores obtiveram com o SOM, em média, uma taxa de compressão 4 vezes maior. Esses resultados foram obtidos usando um espaço de busca de 70 sequências de referência.

A ferramenta GDC 2.0 implementa estratégia semelhante quando comparada ao implementado pela ferramenta FRESCO. GDC 2.0 aumenta o espaço de busca, incluindo novas sequências mapeadas, para cada nova sequência alvo processada pelo Mapeamento de Primeira Ordem. Assim, o espaço de busca considerado pela ferramenta GDC 2.0 não apresenta limitação de tamanho. No entanto, os resultados relatados são baseados apenas no SOM, o que dificultou uma avaliação conclusiva do impacto dessa estratégia na melhoria da taxa de compressão. Nos experimentos descritos, os autores realizaram testes variando o tamanho do espaço de busca de 10% a 100% das sequências, obtendo taxas de compressão variando desde de 300 : 1 até 900 : 1, respectivamente. O tempo de compressão foi reduzido em 24% e o espaço de busca em 50%, enquanto que a taxa de compressão foi reduzida em 26%.

2.11 Pós-processamento

A fase de transformação, implementada pelas ferramentas descritas em Arram *et al.* (ARRAM et al., 2015), iDoComp e COOL, tem como objetivo reduzir o número de elementos a serem armazenados. Quando comparado ao Mapeamento de Segunda Ordem, a diferença nesta etapa de transformação está na direção em que os dados são analisados: horizontal - as transformações são feitas no mapa de cada sequência de destino usando apenas as informações contidas no mapa em análise. O objetivo é mesclar correspondências adjacentes, separadas ou não por incompatibilidades. A partir da análise das ferramentas, foram identificadas três abordagens diferentes para esta fase. Os autores Arram *et al.* (ARRAM et al., 2015) colocaram em prática uma união de correspondências adjacentes não intercaladas por incompatibilidades. Esta abordagem permitiu uma melhoria de até 41,3 vezes na taxa de compressão. A ferramenta COOL (CHERN et al., 2012) (cf. Figura 2.11) avalia a possibilidade de correspondências separadas por inserções, substituições ou pequenas inclusões de elementos. A ferramenta substitui as correspondências combinadas para cada mesclagem,

criando uma nova correspondência mais significativa e uma instrução de edição (substituição, inserção ou exclusão) que é anexada ao seu respectivo conjunto de instruções. Em iDoComp (OCHOA; HERNAEZ; WEISSMAN, 2015), além de tentar mesclar correspondências que são separadas por inserções ou substituições de base única, a ferramenta tenta combinar pequenas correspondências que são intercaladas com pequenas inserções ou substituições que apontam para posições distantes da correspondência anterior. O objetivo deste segundo mecanismo é evitar o armazenamento de pequenas correspondências que não fazem parte de uma cadeia conhecida como subsequência mais longa crescente Longest Increasing Sub-Sequence (LISS). Os autores concluíram que, nesses casos, é mais eficiente armazenar apenas as diferenças tais como instruções de substituição e inserção do que pequenas correspondências que não fazem parte da cadeia LISS. A razão é que essas correspondências presumem a necessidade de armazenar grandes números para representar sua posição, uma vez que a codificação delta desses valores não resulta em números pequenos.

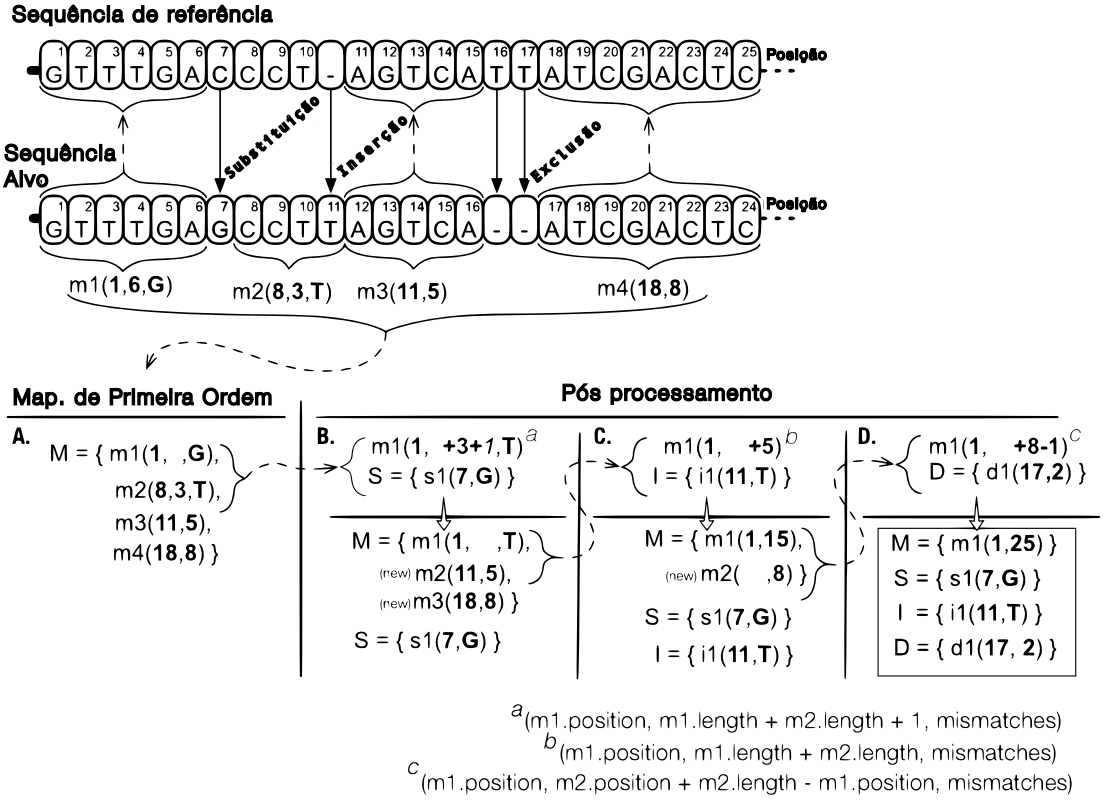

Figura 2.11: Fase de pós-processamento – adaptada de (KREDENS et al., 2020)

A Figura 2.11 ilustra a fase de pós-processamento. Após o mapeamento de primeira ordem (cf. Figura 2.11A), as correspondências do conjunto M são processadas, duas a duas, para fundi-las. Na primeira iteração (cf. Figura 2.11B), em que as correspondências $m1$ e $m2$ são avaliadas, verifica-se que $(m1.position + m1.length + m2.length + 1)$ é igual a $m2.position$, portanto, há uma substituição entre as correspondências. As correspondências são combinadas e $m2$ é removido do conjunto M, enquanto que o restante é reorganizado para a próxima remoção. Então, a instrução $s1$ é criada e inserida no conjunto S que conterá instruções de edição do tipo substituição. Na segunda iteração (cf. Figura 2.11C), as correspondências $m1$ e $m2$ (que eram $m3$) são avaliadas. Verifica-se que $(m1.position + l1.position)$ é igual a $m2.position$, portanto, há uma inserção entre as correspondências. As correspondências são combinadas; $m2$ é removido do conjunto M, e o restante reorganizado para a próxima remoção. A instrução $I1$ é criada e inserida no conjunto I que conterá as instruções de edição do tipo de inserção. Na terceira iteração (cf. Figura 2.11D), as correspondências $m1$ e $m2$ (que eram $m4$) são avaliadas. Verifica-se que há uma exclusão de 2 bases entre as correspondências avaliadas, pois $(2 <= m2.position - (m1.position + m1.length + 1) <= Lmax)$, onde $Lmax$ é um valor predefinido que determina o tamanho máximo que uma exclusão pode ter para ser tratada por pós-processo de processamento. Portanto, as duas correspondências são combinadas e a correspondência $m2$ é removida do conjunto M, que conterá apenas a correspondência $m1$. Então, a instrução $d1$ é criada e inserida no conjunto D que conterá instruções de edição do tipo exclusão. O objetivo é reduzir o número de inteiros. Nesse caso, inicialmente havia 8 números inteiros distintos e, ao final, apenas 6 permaneceram.

2.12 Codificação

Esta é a única fase implementada por todas as ferramentas avaliadas. Essa fase é a última parte do processo de compressão, onde o mapeamento gerado pelas etapas anteriores é codificado em formato binário. Assim, o objetivo é obter a menor entropia possível para os dados comprimidos, em que a menor quantidade de *bits* é buscada para representar os elementos mais frequentes do mapa.

Cada ferramenta também pode aplicar um ou mais métodos de codificação para diferentes elementos do mapa neste estágio. Essa combinação é chamada de estratégia de codificação. Por exemplo, GRS (WANG; ZHANG, 2011) aplica o mesmo método ao mapa como um todo, enquanto GDC-0.3 (DEOROWICZ; GRABOWSKI, 2011) e iDoComp (OCHOA; HERNAEZ; WEISSMAN, 2015) aplicam métodos diferentes para cada elemento do mapa, codificando contextualmente cada grupo de dados. A primeira abordagem (cf. Figura 2.12A) exemplifica o processo em que todos os elementos, que compõem o mapa, são codificados usando o mesmo método. Na segunda abordagem (cf. Figura 2.12B), três fluxos de dados são criados. No primeiro fluxo é aplicado apenas a valores que indicam posições, enquanto que no segundo para valores referentes a coincidir com o tamanho e, finalmente, um outro fluxo de símbolos que não fazem parte de correspondências, chamados literais. Neste exemplo, cada fluxo é codificado separadamente e, ao final, os resultados são combinados, gerando um único arquivo que corresponde a uma representação comprimida da sequência que foi processada. A vantagem dessa segunda abordagem é que cada fluxo tem uma distribuição de símbolo mais homogênea, o que o torna mais previsível. Codificadores estatísticos baseados na distribuição de probabilidade dos símbolos exploram essa homogeneidade. Algumas ferramentas, como JDNA (ALVES et al., 2015) e SLF (BEAL et al., 2016), utilizam outras ferramentas de compressão genéricas, como GZIP ou 7-zip, para alguns elementos que compõem o mapeamento.

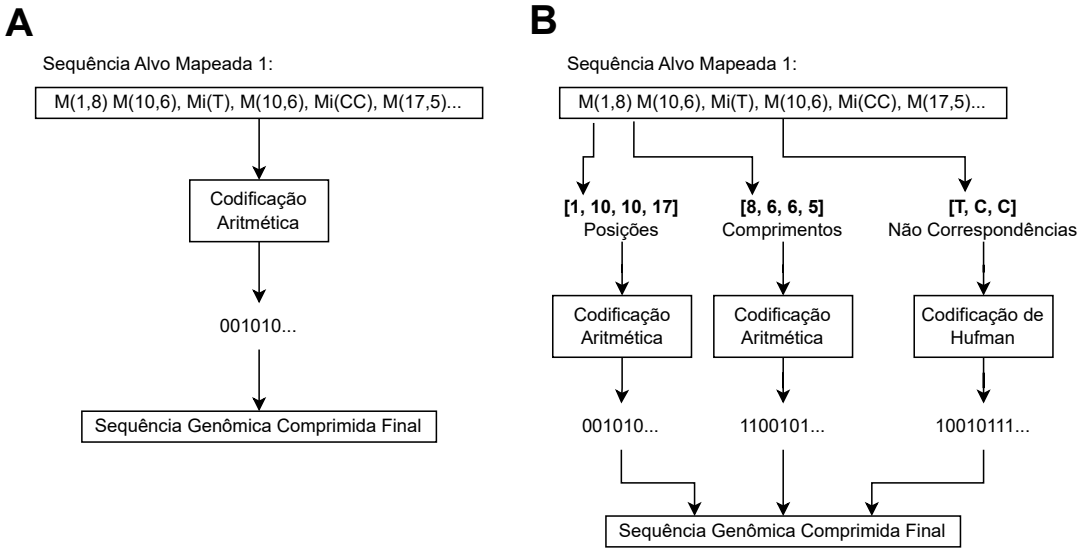

Figura 2.12: Processo de codificação – adaptada de (KREDENS et al., 2020)

A Figura 2.12 ilustra por meio de um exemplo (cf. Figura 2.12A) de como uma ferramenta realiza a codificação quando todos os elementos são enviados para serem codificados em um único fluxo de dados. Em Figura 2.12B, é ilustrada a divisão dos elementos do mapa em três fluxos de dados diferentes. O primeiro contém elementos referentes à posição das correspondências; o segundo referente aos tamanhos; e, por fim, bases não correspondentes. Assim, cada fluxo de dados é codificado separadamente e, ao final, os resultados são combinados em um único arquivo. A vantagem da segunda abordagem é que a distribuição de probabilidade dos elementos é mais concentrada, o que resulta em uma maior taxa de compressão, pois segundo a Teoria da Informação, códigos binários menores devem ser concedidos aos símbolos que aparecem mais.

Considerando todos os artigos avaliados, foi identificado o uso de 15 métodos de codificação: *Huffman Coding* (HUFFMAN, 1952), *Canonical Huffman Coding* (HIRSCHBERG; LELEWER, 1990; SCHWARTZ; KALLICK, 1964; CONNELL, 1973), *Golomb Coding* (GOLOMB, 1966), *Elias Gamma Coding* (ELIAS, 1975), *Arithmetic Coding* (RISSANEN, 1976), *Run Length Encoding* (GOLOMB, 1966), *PPMD* (MOFFAT, 1990), que é uma variante do método *Prediction by Partial Matching* (PPM) (CLEARY; WITTEN, 1984), *Range coding* (ROSEN; GOODWIN; VIDAL, 1991), *GZIP* (GROUP; DEUTSCH; ENTERPRISES, 1996), *Log skewed* (MANZINI; RASTERO, 2004), *Compressed Integer Set* (OKANOHARA; SADAKANE, 2007), *Compressed bit vectors* (FERRADA et al., 2014), *Lempel–Ziv–Markov chain algorithm* (LZMA2), *Variable length byte coding and Arithmetic Coding with encryption capability* (HUANG; LIANG, 2011).

Dentre as ferramentas de compressão vertical avaliadas, identificou-se que apenas o RCSCS (Wiselin Kiruba; RAMAR, 2016) descreve métodos para garantir a proteção de dados com criptografia. Nesse contexto, SECRAM (HUANG et al., 2016) —uma ferramenta de compressão horizontal de dados genômicos— é uma das primeiras a considerar a segurança dos dados baseada em criptografia e controle de acesso aos dados genômicos comprimidos. Outra ferramenta chamada E2FM (MONTECUOLLO; SCHMID; TAGLIAFERRI, 2017), um autoíndice para a coleta de sequências genômicas, também pode criptografar os dados genômicos comprimidos. Porém, em relação à fidelidade das informações, não foi identificado ferramentas com tal controle.

Após avaliar a paisagem atual das ferramentas de compressão vertical e suas respectivas abordagens à segurança e fidelidade dos dados, é imperativo explorar outro paradigma emergente na compressão de sequências genômicas:

as ferramentas baseadas em modelos preditivos. Este enfoque representa uma evolução significativa no domínio da compressão genômica e oferece vantagens e desafios únicos que merecem uma discussão aprofundada.

2.13 Ferramentas baseadas em modelos preditivos

"Aprendizado de máquina, em inteligência artificial (disciplina da ciência da computação), é uma disciplina que se preocupa com a implementação de software de computador que pode aprender de forma autônoma"((Encyclopaedia Britannica, 2009), tradução nossa).

A partir do ano de 2018, a literatura científica tem testemunhado o advento de métodos que empregam modelos preditivos para a compressão de sequências genômicas baseados em predição do próximo símbolo. Nestes métodos, a fase de predição é um componente importante do processo de compressão. Quando a predição do símbolo subsequente falha, tanto as coordenadas espaciais quanto o tipo de símbolo devem ser armazenados, podendo ou não ser submetidos a transformações adicionais para otimização. O algoritmo, uma vez treinado, gera um modelo preditivo que mimetiza estreitamente a sequência alvo em termos de capturar informações da sequência. Diferentemente da compressão vertical referencial, onde a sequência de referência serve como a fonte primária de informação para comparações e alinhamentos, na abordagem baseada em modelos preditivos, é o próprio modelo que atua como a fonte primária de informação. Portanto, pode-se argumentar que a compressão baseada em modelo preditivo representa uma forma especializada de compressão referencial.

No vasto domínio da Aprendizagem de Máquina e Aprendizagem Profunda, uma quantidade relevante de modelos de redes neurais tem sido desenvolvida, cada um com características únicas e adaptadas para diferentes aplicações e desafios. Entretanto, neste contexto, o foco recai sobre um conjunto específico de tecnologias que foram empregados nos artigos selecionados pertinentes ao tema. Esses modelos representam o estado da arte na área de predição do próximo símbolo para sequências de DNA.

2.13.1 Predição do Próximo Símbolo em Aprendizagem de Máquina

Feedforward Neural Network (FNN) As redes neurais *feedforward* representam uma classe de redes neurais onde as conexões entre os neurônios não for-

mam ciclos (ZELL, 1994). Esta arquitetura implica em um fluxo unidirecional da informação, partindo das camadas de entrada, passando pelas camadas ocultas (se houver) e culminando na camada de saída. Uma característica fundamental da rede Feedforward Neural Network (FNN) é a ausência de conexões recorrentes, o que as torna particularmente adequadas para tarefas onde as dependências temporais ou sequenciais não são primordiais. O modelo *GeCo3* emprega uma arquitetura *FNN* totalmente conectada com uma única camada oculta, escolhida pela sua implementação simples e eficiência comparável a redes mais complexas (SILVA; PRATAS; PINHO, 2020). A rede utiliza a função de ativação sigmoide, com o erro quadrático médio como função de perda. A inicialização dos pesos segue o método de Xavier (GLOROT; BENGIO, 2010), facilitando a convergência durante o treinamento.

Recurrent Neural Network (RNN) As Redes Neurais Recorrentes (Recurrent Neural Network (RNN)) são redes neurais projetadas para processar dados sequenciais (DUPOND, 2019). Diferentemente das Redes Neurais Convolucionais (Convolutional Neural Network (CNN)), que capturam padrões espaciais em janelas de contexto fixas, as RNNs mantêm uma memória de eventos passados, permitindo que informações anteriores influenciem a saída atual. Isso é útil em tarefas onde a ordem dos dados é importante, como em textos, séries temporais e sequências de DNA. As RNNs possuem *loops* internos que transmitem informações de um passo de tempo para o próximo, criando uma memória dinâmica. No entanto, enfrentam dificuldades com dependências de longo prazo devido ao problema do desvanecimento do gradiente, que impede o aprendizado de relações distantes. Apesar dessas limitações, as RNNs são utilizadas em aplicações como modelagem de linguagem e previsão de séries temporais. A performance das RNNs pode ser aprimorada com técnicas como normalização dos dados de entrada, ajustes na arquitetura da rede e combinação com outras redes neurais, como CNNs e LSTMs, para capturar padrões locais e globais.

Convolutional Neural Network (CNN) Redes Neurais Convolucionais (LECUN; BENGIO, 1995) são utilizadas em tarefas de processamento de imagens, reconhecimento de padrões e análise de sequências, como em genômica. Em contraste com a RNN, que é especializadas em dados sequenciais como textos ou séries temporais, a CNN é mais eficiente na captura de padrões espaciais e temporais dentro de uma janela fixa de contexto. Em genômica, por exem-

plo, uma CNN pode identificar padrões significativos em sequências de DNA, considerando características locais e globais. Uma característica fundamental das CNNs é sua capacidade de aprender representações hierárquicas de dados, usando camadas de convolução com filtros que detectam características em diferentes níveis de abstração (MATSUGU et al., 2003). Isso as torna particularmente eficazes para dados com padrões complexos e variados. No entanto, uma limitação das CNNs é a necessidade de pré-determinar o tamanho da janela de contexto, o que pode ser desafiador em aplicações genômicas onde a relevância do contexto pode variar substancialmente. A eficácia de uma CNN em tarefas específicas depende fortemente da escolha e da configuração desses filtros e camadas.

Long Short-Term Memory (LSTM) Uma rede LSTM é um tipo de rede neural recorrente capaz de aprender a ”dependência de ordem” em problemas de previsão do próximo elemento em dados sequenciais (HOCHREITER; SCHMIDHUBER, 1997). A principal diferença entre uma rede CNN (FUKUSHIMA, 1980) e uma rede LSTM está na forma como os parâmetros são calculados e construídos. Uma das vantagens da LSTM é sua indiferença ao comprimento dos intervalos. O diferencial da arquitetura LSTM é que este modelo, por causa de sua estrutura celular, é capaz de “lembrar”. A presença de ativações do "portão de esquecimento"(*forget gate*) permite que uma rede LSTM decida, a cada passo de tempo, que determinada informação não deve ser negligenciada, e assim, atualize os parâmetros do modelo. O modelo do compressor DeepDNA combina as capacidades das redes CNN e LSTM (WANG et al., 2018). Esta fusão aproveita a eficiência da CNN na captura de padrões espaciais e temporais em sequências de DNA e a habilidade da rede LSTM em processar sequências temporais, considerando dependências de longo prazo. A combinação de CNN e LSTM permite que o modelo DeepDNA capture características locais e dependências contextuais de longa distância em sequências de DNA, o que é crucial para prever o próximo símbolo na sequência.

Bi-directional Long Short-Term Memory (biLSTM) Uma abordagem biLSTM melhora a arquitetura LSTM tradicional permitindo que a rede tenha uma visão tanto do passado quanto do futuro no momento atual (BALDI et al., 1999). Em um modelo LSTM padrão, a rede só pode ter acesso às informações anteriores na sequência, o que pode ser uma limitação em tarefas onde o contexto futuro é

igualmente importante para a tomada de decisão. A rede biLSTM resolve isso ao processar os dados em duas direções com duas camadas separadas de LSTM. Uma camada lida com a sequência na ordem cronológica (direta), enquanto a outra processa na ordem inversa (reverso). As saídas destas duas camadas são combinadas a cada etapa de tempo, resultando em uma rica representação do contexto da sequência. Essa abordagem tem demonstrado melhorar significativamente o desempenho em tarefas como reconhecimento de fala, tradução automática e análise de sentimentos, onde o contexto em ambas as direções é crucial. O modelo dos autores Cui et al. (2020), amplia a abordagem da biLSTM com a adição do mecanismo de atenção. Aqui, a CNN identifica padrões locais na sequência de DNA, enquanto a biLSTM processa estas informações em ambas as direções temporais, capturando as dependências passadas e futuras. O componente de atenção potencializa este modelo ao permitir que ele se concentre em partes específicas da sequência de DNA que são mais relevantes para a tarefa de predição.

2.14 Considerações do capítulo

A demanda por espaço de armazenamento de dados genômicos tornou-se uma das partes importantes no avanço da era genômica. No entanto, a compressão de dados é uma alternativa importante para reduzir tal limitação. Nesse sentido foi investigado amplamente, em forma de uma revisão da literatura, todas as ferramentas existentes para compressão, com abordagem vertical, de sequências genômicas montadas e armazenadas no formato FASTA. No total, foram identificadas e caracterizadas 36 ferramentas em diferentes aspectos: o esquema de compressão e a forma como comprimem uma única sequência ou um conjunto delas; o tamanho dos alfabetos considerados; se as ferramentas usam memória externa durante os processos de compressão; se elas permitem acesso direto aos dados comprimidos, com ou sem a necessidade de descompressão.

Dos trabalhos analisados na revisão da literatura, foi obtido o fluxo de execução de cada ferramenta, que é constituído pelas fases de pré-processamento, transformação e codificação dos dados. Para cada fase, foram identificadas abordagens específicas para lidar com: a) seleção, indexação e uso de sequência de referência (cf. Figuras 2.6 e 2.8); b) busca de segmentos compartilhados entre sequências que apresentam comportamento específico por meio da téc-

nica chamada *Busca Local* (cf. Figura 2.9); c) tentativa de reduzir a quantidade total de elementos a serem armazenados (Mapeamento de Segunda Ordem ou Pós-processamento) (cf. Figuras 2.10 e 2.11), e d) codificação com o uso de diferentes técnicas para diferentes elementos resultantes da fase de mapeamento (cf. Figura 2.12).

A análise das ferramentas também permitiu identificar a falta de um protocolo padronizado para avaliar o desempenho das ferramentas de compressão vertical de dados genômicos. Assim, para poder comparar diferentes ferramentas qualitativamente, seria necessária uma análise comparativa com os mesmos conjuntos de dados. Foi considerado também que, para melhor avaliar cada fase de compressão, as ferramentas poderiam apresentar algumas métricas referentes aos processos executados. Por exemplo, em relação à fase de transformação, detalhes podem ser descritos, como a quantidade total de correspondências entre sequências, tamanho médio de correspondência, tamanho total (em *bytes*) de correspondências, quantidade de incompatibilidades, tamanho médio de incompatibilidades, tamanho total (em *bytes*) de incompatibilidades e a entropia final do mapa gerado. Essas informações podem ser usadas para verificar qual ferramenta (ou abordagem) cria o melhor mapa ou qual pode identificar a quantidade mais significativa de correspondências. Com isso, seria possível realizar uma verificação entre a porcentagem de sequência alvo identificada como semelhante, ou o tamanho médio de cada correspondência em relação à taxa de compressão final. Desta forma, seria possível verificar a relação entre a capacidade de localizar semelhanças e a taxa de compressão alcançada. Isso também permitiria avaliar a eficiência de cada abordagem em relação à compressão de sequências genômicas de diferentes organismos e com diferentes níveis de similaridade. Na análise realizada, apenas a ferramenta SLF forneceu o tamanho do mapa gerado —esta informação é crucial para identificar como a codificação influenciou a taxa de compressão final.

Os resultados apresentados pelos trabalhos avaliados mostram que as ferramentas especializadas de compressão vertical de dados genômicos permitem reduzir o espaço necessário para armazenar os dados, independentemente do método considerado. Um dos principais problemas na compressão vertical de dados genômicos é encontrar uma relação otimizada entre os mapeamentos que capture o máximo de segmentos compartilhados com os menores custos de codificação possíveis.

Além disso, as ferramentas avaliadas pouco exploram as características biológicas (propriedades estatísticas) das sequências genômicas. Várias características biológica-estatísticas poderiam ser consideradas com o objetivo de aumentar a taxa de compressão. Por exemplo, algumas dessas características são: os palíndromos, as repetições aproximadas, as repetições exatas, a distribuição das bases nitrogenadas, a distribuição de pares de bases CG, a distribuição de pares de bases AT, o complemento da sequência, o reverso complemento, a distribuição dos códons, os padrões encontrados em determinada região genômica, entre outras características. Estes são apenas alguns exemplos das características biológicas (propriedades estatísticas) existentes em uma sequência genômica e que podem ser exploradas para construção de ferramentas especializadas.

3

Método de Compressão

Neste capítulo, apresentam-se duas contribuições metodológicas principais: um protocolo de avaliação de desempenho para ferramentas de compressão de sequências genômicas e um método de compressão baseado em modelo preditivo. Inicialmente, definiu-se um protocolo mínimo que estabelece padrões uniformes de avaliação, emergindo da revisão sistemática da literatura discutida no Capítulo 2, a qual identificou a lacuna significativa de um método padronizado para avaliação de desempenho. Tal protocolo já foi publicado em (KREDENS et al., 2020). Esta padronização visa possibilitar comparações coerentes entre diferentes estudos. Em seguida, detalha-se o método de compressão proposto, que integra técnicas preditivas com transformações de dados para otimizar o armazenamento dos símbolos que representam as bases nitrogenadas, reduzindo deste modo o espaço necessário para armazenamento dos dados sem comprometer sua integridade. Finalmente, o protocolo de avaliação é aplicado ao método de compressão desenvolvido, permitindo sua análise comparativa com ferramentas *baseline*, utilizando métricas detalhadas para assegurar uma avaliação transparente dos resultados.

3.1 Protocolo: Métricas para avaliação de desempenho

Para fornecer resultados reproduzíveis e métricas de compressão significativas, este Capítulo também descreve os critérios para avaliação de desempenho

a serem utilizados na compressão de sequências genômicas. Conforme mencionado anteriormente neste trabalho, o processo de compressão de dados é similar entre as ferramentas presentes na literatura, que se baseia em três fases distintas. As fases são: pré-processamento, transformação e codificação. Apesar disso, para que os resultados de *benchmark* a serem apresentados para cada ferramenta possam ser comparados entre si, é necessário seguir um protocolo de avaliação comum. Tal protocolo deve conter um conjunto mínimo de requisitos básicos em relação aos dados que estão sendo comprimidos (conjunto de dados) e métricas indispensáveis para a avaliação do desempenho da compressão. No entanto, não foi encontrado na literatura um protocolo com o objetivo específico de padronizar a avaliação da ferramenta de compressão de genoma que atendesse a tais critérios. Para contornar este problema, foram propostos alguns critérios a serem observados para que os resultados de *benchmark* entre as diferentes ferramentas de compressão de dados de genoma possam ser comparados.

3.1.1 Conjunto de dados para testes

A padronização do conjunto de dados a ser utilizado nos testes de compressão é um dos principais pontos para garantir que os resultados obtidos pelas ferramentas de compressão de dados sejam comparáveis entre si. Encontrou-se na literatura dois autores que publicaram conjuntos de dados para uso em avaliação de desempenho de compressores de genoma, conforme Referências (PRATAS; PINHO, 2019; BIJI; NAIR, 2017).

O conjunto de dados de referência proposto por Pratas & Pinho (2019) contém 15 sequências genômicas. Este conjunto de dados contém um total de 534.263.017 símbolos representando as bases nitrogenadas, o que é aproximadamente 0,5 *gigabyte*. Além disso, mantém um equilíbrio consistente entre o número de sequências e seus respectivos tamanhos. Ainda, o conjunto também reflete os principais domínios e reinos dos organismos biológicos e, portanto, permite uma comparação abrangente e equilibrada para métodos de compressão de dados de genoma. Para ferramentas que manipulam arquivos do tipo FASTA (ou multi-FASTA), é fundamental destacar que cada sequência deste conjunto de dados está em formato bruto. Assim, não possui cabeçalhos, quebras de linha na sequência ou qualquer símbolo diferente dos do alfabeto ACGT.

O conjunto de dados proposto por Biji & Nair (2017) consiste em sequências de vários organismos biológicos diferentes, com 1105 procariotos, 200 plasmídeos, 164 vírus e 65 eucariotos. Além disso, o autor deste trabalho encontrou uma forma estatística de selecionar as amostras para compilar o conjunto de dados para uso em *benchmark*, usando estratégias de amostragem em múltiplos estágios. Os dados neste conjunto são compostos por arquivos FASTA e multi-FASTA.

Ambos os conjuntos de dados são bem fundamentados em sua representação do universo de sequências genômicas em bases de dados genômicos públicos, como o National Center for Biotechnology Information (NCBI) (AGARWALA et al., 2016). No entanto, o primeiro conjunto de dados mencionado anteriormente e introduzido em (PRATAS; PINHO, 2019) é relativamente menor e, portanto, o tempo de processamento para compressão e descompressão será consequentemente mais curto.

Os conjuntos de dados genômicos mencionados até agora oferecem uma representação diversificada, cobrindo uma amostra significativa de diferentes espécies e reinos biológicos. Esta diversidade é crucial para uma análise abrangente e equitativa das ferramentas de compressão. Embora os conjuntos de dados genômicos supramencionados apresentem ampla cobertura de múltiplas espécies, eles incluem apenas uma única sequência por organismo, limitando assim a sua aplicabilidade em contextos que requerem várias sequências de um único organismo para análises aprofundadas. Esta limitação torna-se decisiva na aprendizagem profunda, onde a diversidade e a quantidade de dados desempenham um papel fundamental no desenvolvimento de modelos robustos e precisos.

A necessidade de tais conjuntos de dados é imperativa para superar os desafios atuais e impulsionar inovações no aprendizado de máquina aplicado à genômica. Assim, para a formação de modelos preditivos, onde é necessária uma maior quantidade de sequências genômicas, semelhantes mas não idênticas, para garantir a robustez e precisão do modelo, pode-se recorrer aos extensos conjuntos de sequências disponíveis em (AGARWALA et al., 2016) . Esses conjuntos mais amplos fornecem uma ampla gama de sequências genômicas para o treinamento e validação de modelos preditivos, permitindo uma análise mais detalhada e representativa das características genômicas.

3.1.2 Métricas De Eficiência de Armazenamento

Para este quesito, duas métricas são de fundamental importância para a avaliação do desempenho das ferramentas especializadas de compressão de sequências genômicas. Tais métricas são: a economia de espaço (cf. Equação 3.1) e o número de *bits* para armazenar a informação de um símbolo (*bits* por base, ou *bpb*) (cf. Equação 3.2). O uso dessas métricas está relacionado à compressão usando a codificação ingênua de *bits* ou *Naïve-bit Encoding* (GRUMBACH; TAHI, 1994). Portanto, a abordagem mais simples para comprimir uma sequência genômica, sem a necessidade de qualquer ferramenta especializada, é atribuir dois *bits* para cada símbolo do *Alfabeto Genômico {A, T, C, G}* que representam as bases nitrogenadas. Como resultado, é possível verificar que, sem o uso de técnicas avançadas, é possível obter uma economia de espaço de 75% usando a codificação de 2 *bits* por base. Isso ocorre porque são necessários 8 *bits* para armazenar um único símbolo American Standard Code for Information Interchange (ASCII) no formato FASTA. No entanto, ao codificar cada símbolo com 2 *bits*, há uma economia de espaço de 75%. A economia de espaço é calculada pela equação:

$$economiaDeEspaco = \left(1 - \frac{tamanhoComprimido}{tamanhoDescomprimido}\right) * 100 \tag{3.1}$$

Por exemplo, considere que um arquivo original de 5000 *bytes* é comprimido para 1000 *bytes*. A economia de espaço pode ser calculada substituindo os valores na equação: $economiaDeEspaco = \left(1 - \frac{1000}{5000}\right) * 100 = 80\%$. Este resultado indica que a compressão reduziu o tamanho do arquivo original em 80%, demonstrando uma significativa economia de espaço.

Da mesma forma, para calcular o número de *bits* usados para armazenar cada símbolo, a equação usada é:

$$bitsPorBase = \frac{numeroDeBits}{numeroDeSimbolos} \tag{3.2}$$

A variável *numeroDeBits* representa a quantidade de *bits* usados para armazenamento dos dados da sequência após a compressão, e *numeroDeSimbolos* é o número de símbolos, que representam as bases nitrogenadas na sequência genômica, antes da compressão. Considere que uma sequência é representada por 1000 bases (símbolos) e após a compressão com uma ferramenta especializada o total de *bits* usados para armazenar essa sequência foi de 1500. Aplicando

a equação 3.2, tem-se: $bitsPorBase = \frac{1500}{1000} = 1.5$ *bits* por base. Se a mesma sequência fosse armazenada usando a codificação ASCII, que utiliza 8 *bits* por símbolo, seriam necessários 8000 *bits* para armazená-la. Este resultado de 1.5 *bits* por base indica que a compressão especializada é mais eficiente que a codificação "ingênua" (*naïve-bit*), que utilizaria 2 *bits* por base. Assim, considerando a codificação ASCII, a economia de espaço alcançada pela compressão especializada pode ser mais significativa, correspondendo a 81,25% contra 75,00% (fixo) da codificação "ingênua". Consequentemente, é possível determinar se o desempenho da ferramenta especializada é melhor do que o desempenho de compressão usando codificação "ingênua" de 2 *bits* por base.

Além disso, para calcular a taxa de compressão, ou seja, quantas vezes diminuiu do tamanho original para o tamanho comprimido, usa-se a fórmula:

$$taxaDeCompressao = \frac{tamanhoDescomprimido}{tamanhoComprimido} \tag{3.3}$$

Dessa forma, se um arquivo no formato FASTA tem x *bytes* pré-compressão e y *bytes* pós-compressão, a taxa de compressão pode ser expressa como $\frac{x}{y} : 1$. Por exemplo, considerando $x = 200$ *bytes* pré-compressão e $y = 50$ *bytes* pós-compressão, a taxa de compressão é de $\frac{200}{50} : 1$, que simplifica para $4 : 1$ (quatro para um) (cf. Equação 3.3).

3.1.3 Métricas para Medir Eficiência de Tempo

É relevante coletar informações sobre o tempo de compressão e descompressão quando o arquivo comprimido não permite acesso aleatório aos dados. Assim, sempre que for necessário acessar alguma informação do genoma que foi comprimido, como por exemplo, uma posição de um determinado símbolo ou ainda, uma cadeia de símbolos, será necessário primeiro descomprimir todo o genoma. Portanto, o processo de descompressão ocorrerá toda vez que houver necessidade de realizar uma busca local nos dados do genoma comprimido. Nessa perspectiva, pode-se afirmar que o tempo de descompressão (cf. Equação 3.5) é mais importante do que o tempo de compressão (cf. Equação 3.4), uma vez que o processo de compressão acontecerá apenas uma vez, ainda que esse processo demore mais para ser concluído. Assim, o tempo de compressão e descompressão, em segundos, dos métodos e ferramentas de compressão de dados do genoma pode ser medido usando as equações:

$$tempoDeCompressao = (tempoFinal - tempoInicial) \tag{3.4}$$

e

$$tempoDeDescompressao = (tempoFinal - tempoInicial) \tag{3.5}$$

Em que a variável *tempoDeCompressao* denota o intervalo necessário para a compressão dos dados, enquanto *tempoDeDescompressao* refere-se ao período requerido para a recuperação dos dados ao seu estado original. Por sua vez, *tempoInicial* é definido como o instante preciso, medido em segundos, no qual o processo de compressão ou descompressão é iniciado, e *tempoFinal* representa o instante, também em segundos, em que tal processo atinge sua conclusão. Por exemplo, se um processo de compressão é iniciado no instante *tempoInicial* de 100 segundos e concluído no *tempoFinal* de 400 segundos, então o *tempoDeCompressao* seria calculado como: $tempoDeCompressao = 400 - 100 = 300$ segundos. Da mesma forma, se o processo de descompressão começar no *tempoInicial* de 200 segundos e terminar no *tempoFinal* de 350 segundos, o *tempoDeDescompressao* será: $tempoDeDescompressao = 350 - 200 = 150$ segundos. Esse processo assegura uma avaliação precisa da performance temporal das ferramentas de compressão, fornecendo indicadores fundamentais para a comparação entre diferentes algoritmos e implementações.

3.1.4 Métricas para Medir Eficiência de Uso de Memória e CPU

A discussão dessas métricas precisa ser aprofundada em termos de como elas são coletadas e analisadas. A dificuldade está em saber o tempo ideal para coletar o pico ou média de uso da memória e a porcentagem de uso da CPU. Dependendo da estratégia a ser abordada, o processo de compressão ou descompressão dos dados do genoma pode usar até 100% da memória e CPU disponíveis de acordo com a complexidade dos algoritmos e o *hardware* disponível. Em contrapartida, algumas ferramentas podem usar porcentagens de memória mais baixas e atingir o máximo de memória ou uso de CPU em apenas uma das fases da compressão/descompressão. Portanto, a sugestão nesse ponto é que a coleta mínima de dados para essa métrica capture a média de memória utilizada durante o processo de compressão e descompressão (cf. Equação 3.6). Amostras do uso de memória podem ser coletadas em intervalos

regulares, formando um conjunto de dados que reflete o consumo de memória ao longo da operação. Para obter uma medida precisa e representativa do consumo de memória, calcula-se a média do uso de memória a partir dessas amostras. Esta média é determinada somando-se todas as amostras de memória coletadas e dividindo-se pelo número total de amostras. Este valor médio serve como um indicador do consumo de memória da ferramenta durante todo o período de sua execução.

A expressão da média de memória pode ser representada pela equação:

$$usoMedioDeMemoria = \frac{\sum_{i=1}^{n} usoDeMemoria\_i}{n} \tag{3.6}$$

Onde *usoDeMemoria_i* representa a quantidade de memória utilizada na i-ésima amostra. n é o número total de amostras coletadas durante o processo de compressão ou descompressão.

Considere que, durante um processo de compressão, foram coletadas cinco amostras de uso de memória, registradas como 1024, 2048, 1536, 1024 e 2048 *Megabytes* respectivamente. Aplicando a equação 3.6, o uso médio de memória pode ser calculado como: $usoMedioDeMemoria = \frac{1024+2048+1536+1024+2048}{5} = 1536$ Megabytes. Assim, a média de memória utilizada durante o processo é de 1536 *Megabytes*.

Para a avaliação mínima do consumo de CPU pelas ferramentas, sugere-se a capture da média do uso da CPU durante o processo (cf. Equação 3.7). Este método envolve a coleta contínua de amostras do uso da CPU ao longo da operação, fornecendo uma visão do comportamento de consumo de recursos do processador.

As amostras do uso da CPU podem ser coletadas em intervalos regulares e armazenadas para análise posterior. A média do uso da CPU é calculada somando todas as amostras coletadas e dividindo pelo número total de amostras. Este valor médio oferece um indicador representativo do consumo de CPU ao longo da operação da ferramenta, seja durante o processo de compressão ou descompressão.

A equação para calcular a média do uso da CPU é:

$$usoMedioDeCPU = \frac{\sum_{i=1}^{n} usoDeCPU\_i}{n} \tag{3.7}$$

Onde *usoMedioDeCPU* representa a média do uso da CPU. *usoDeCPU_i* é a porcentagem de uso da CPU na i-ésima amostra. n é o número total de

amostras coletadas durante o processo. Suponha que, durante uma análise de desempenho de um algoritmo de compressão, foram coletadas quatro amostras do uso da CPU, com os seguintes valores percentuais: 70%, 65%, 80% e 75%. Utilizando a equação 3.7, o uso médio de CPU é calculado da seguinte maneira: $usoMedioDeCPU = \frac{70+65+80+75}{4} = 72.5\%$. Portanto, a média do uso da CPU durante o processo é de 72,5%. Este método assegura uma avaliação consistente do desempenho da CPU, permitindo comparações entre diferentes ferramentas de compressão e descompressão.

3.1.5 Comparação Estatística de Desempenho das Ferramentas

Comparar o desempenho de compressão/descompressão com os resultados de outras ferramentas especializadas de compressão de genoma como linha de base torna os resultados mais relevantes. A relevância está na possibilidade de verificar se as taxas de compressão da ferramenta especializada de compressão de dados do genoma que está sendo desenvolvida alcançam melhores resultados do que as taxas de compressão das ferramentas existentes. Para a definição dessas ferramentas *baseline* deve ser avaliado na literatura ferramentas Estado da Arte cujos autores, dos trabalhos avaliados, utilizaram para comparação de resultados. Além disso, para a validação dos resultados, faz-se necessário a aplicação de testes estatísticos não paramétricos, como o Teste de *Friedman* (FRIEDMAN, 1937) e o Teste de *Nemenyi* (NEMENYI, 1962). A escolha de testes estatísticos não paramétricos é motivada pelas características dos dados que estão sendo analisados. No contexto da comparação de ferramentas de compressão de genoma, os dados coletados geralmente não seguem uma distribuição normal, o que torna os testes paramétricos, como a *ANOVA*, inadequados.

Os testes não paramétricos são apropriados quando se lida com variáveis dependentes que não necessariamente seguem uma distribuição normal. As variáveis dependentes podem ser as taxas de compressão, economia de espaço ou mesmo os tempos de compressão/descompressão das ferramentas, que são medidas quantitativas contínuas. As variáveis independentes são as diferentes ferramentas especializadas de compressão de genoma e os conjuntos de dados utilizados nas avaliações. Dada a natureza dos dados e a possibilidade de não atenderem aos pressupostos de normalidade, o uso de testes não paramétricos permite uma análise estatística adequada. Esses testes em questão não fazem suposições rígidas sobre a distribuição dos dados, permitindo uma compara-

ção mais confiável entre as ferramentas avaliadas. Além disso, os testes não paramétricos são adequados para dados de natureza ordinal, como os ranques de desempenho das ferramentas, que são obtidos ao comparar as variáveis dependentes. O Teste de *Friedman*, por exemplo, é ideal para comparar múltiplos grupos em diferentes condições (cf. Equação 3.8), e o Teste de *Nemenyi* é usado para realizar comparações pareadas *post-hoc*, identificando quais grupos diferem significativamente entre si (cf. Equação 3.9). A equação para o Teste de *Friedman* é:

$$\chi_F^2 = \frac{12N}{k(k+1)} \left[\sum_j R_j^2 - \frac{k(k+1)^2}{4} \right] \tag{3.8}$$

Onde χ_F^2 representa o valor do teste estatístico de *Friedman*, N é o número de condições ou conjuntos de dados, k é o número de ferramentas comparadas, R_j é a soma dos $ranks$ para a ferramenta j. O termo $\frac{12N}{k(k+1)}$ é um fator de normalização utilizado no cálculo do teste, $\sum_j R_j^2$ representa a soma dos quadrados dos $ranks$ para cada ferramenta e $\frac{k(k+1)^2}{4}$ é um ajuste que faz parte do cálculo do valor do teste. Considere uma análise estatística onde quatro ferramentas de compressão de genoma são avaliadas utilizando três diferentes conjuntos de dados. Suponha que as somas dos ranques para cada ferramenta, baseadas na eficiência de compressão, são respectivamente 6, 8, 10 e 12. Aplicando a equação 3.8 do Teste de *Friedman*, com $N = 3$ (conjuntos de dados) e $k = 4$ (ferramentas), tem-se:

$$\begin{aligned}
\chi_F^2 &= \frac{12 \times 3}{4 \times (4+1)} \left[(6^2 + 8^2 + 10^2 + 12^2) - \frac{4 \times (4+1)^2}{4} \right] \\
&= \frac{108}{20} \left[56 + 64 + 100 + 144 - 50 \right] \\
&= \frac{108 \times 314}{20} \\
&= 1699.2
\end{aligned}$$

Para interpretar esse valor (1699.2), é necessário compará-lo com os valores críticos da distribuição qui-quadrado (χ^2) com os graus de liberdade apropriados para determinar a significância estatística (DANIEL, 1990). Os graus de liberdade para o Teste de *Friedman* são $k-1$, onde k é o número de grupos ou ferramentas comparadas. No exemplo dado com quatro ferramentas, os graus de

liberdade seriam $4 - 1 = 3$. Além disso, para encontrar o *p-valor* correspondente ao valor de χ_F^2 calculado, consulta-se a tabela de distribuição qui-quadrado ou usa-se uma função de software estatístico que pode calcular o *p-valor* a partir do valor do teste estatístico e dos graus de liberdade. O *p-valor* indica a probabilidade de observar um valor de teste tão extremo quanto o χ_F^2 ou mais, sob a hipótese nula de que todas as ferramentas têm desempenho semelhante (ou seja, as diferenças nos ranques são atribuíveis ao acaso). Um *p-valor* baixo (tipicamente menor que 0,05) sugere que as diferenças entre as ferramentas são estatisticamente significativas, levando à rejeição da hipótese nula (DANIEL, 1990). Este cálculo mostra que, usando o Teste de *Friedman*, pode-se avaliar objetivamente as diferenças no desempenho entre as ferramentas de compressão de genoma sob diversas condições.

Já o Teste de *Nemenyi* é usado subsequentemente para avaliar quais pares de ferramentas/experimentos de um dado conjunto apresentam diferenças significativas entre si (cf. Equação 3.9). A equação para o Teste de *Nemenyi* é:

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}} \tag{3.9}$$

Onde CD é a diferença crítica de *Nemenyi*, q_α é o valor crítico de *Nemenyi*, dependente do nível de significância escolhido α, k é o número de ferramentas comparadas, N é o número de condições ou conjuntos de dados. O termo $\sqrt{\frac{k(k+1)}{6N}}$ é um fator de escala que ajusta a diferença crítica com base no número de ferramentas e condições. Considere quatro ferramentas de compressão de genoma e três conjuntos de dados, cujo nível de significância α seja de 0.05, e o valor crítico de *Nemenyi* q_α para este caso seja aproximadamente 2.728 (dependendo da tabela de valores críticos mencionada anteriormente). Utilizando a equação 3.9, tem-se:

$$\begin{aligned} CD &= 2.728 \times \sqrt{\frac{4 \times (4+1)}{6 \times 3}} \\ &= 2.728 \times \sqrt{\frac{20}{18}} \\ &= 2.728 \times \sqrt{1.111} \approx 2.728 \times 1.054 \approx 2.876 \end{aligned}$$

Portanto, uma diferença crítica de aproximadamente 2.876 pontos nos ranques entre quaisquer par de ferramentas é necessária para considerar que há uma

diferença estatisticamente significativa entre elas no nível de 0.05. Isso significa que, se a diferença entre os ranques de duas ferramentas de compressão de genoma exceder 2.876, pode-se concluir que elas têm desempenhos significativamente diferentes sob as condições testadas. Esta abordagem estatística permite uma análise quantitativa com maior precisão para saber se as ferramentas de compressão de dados genômicos apresentam variações de desempenho verdadeiramente distintas ou são estatisticamente comparáveis quando os desempenhos observados são equivalentes.

3.1.6 Análise de Correlação de Desempenho das Ferramentas

O coeficiente de correlação de *Pearson* ou apenas correlação de *Pearson* (BENESTY et al., 2009) é uma medida que pode ser aplicada para complementar a análise estatística das ferramentas de compressão de genoma. A medida avalia a força e a direção do relacionamento linear entre duas variáveis quantitativas. No contexto da comparação de desempenho das ferramentas de compressão de sequências genômicas, a correlação de *Pearson* pode ser utilizada para investigar a relação entre as taxas de compressão e outras métricas de desempenho, como o tempo de compressão/descompressão, entre outras. A aplicação dessa correlação permite identificar possíveis dependências entre as variáveis analisadas, fornecendo uma visão mais abrangente das interações entre diferentes fatores que influenciam o desempenho das ferramentas. Ao incluir a correlação de *Pearson* na análise, é possível corroborar os resultados obtidos pelos testes não paramétricos e oferecer uma interpretação mais detalhada das variáveis envolvidas, assegurando que as conclusões serão suportadas por múltiplas evidências estatísticas.

A fórmula para o coeficiente de correlação de Pearson é:

$$r = \frac{\sum(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum(x_i - \overline{x})^2 \sum(y_i - \overline{y})^2}} \tag{3.10}$$

Onde x_i e y_i são os valores das variáveis X e Y, $\overline{x}$ e $\overline{y}$ são as médias das variáveis X e Y.

O coeficiente de correlação de *Pearson* (r) varia de -1 até 1 e indica a força e a direção da relação linear entre duas variáveis (cf. Tabela 3.1). Esses intervalos proporcionam uma interpretação clara da força e direção da relação linear entre as variáveis analisadas.

Tabela 3.1: Intervalos do coeficiente de correlação de *Pearson* e sua interpretação.

Intervalo	Interpretação
$r = 1$	Correlação positiva perfeita.
$0.7 \leq r < 1$	Correlação forte positiva.
$0.4 \leq r < 0.7$	Correlação moderada positiva.
$0.1 \leq r < 0.4$	Correlação fraca positiva.
$r = 0$	Nenhuma correlação linear.
$-0.1 \leq r < 0$	Correlação fraca negativa.
$-0.4 \leq r < -0.1$	Correlação moderada negativa.
$-0.7 \leq r < -0.4$	Correlação forte negativa.
$r = -1$	Correlação negativa perfeita.

No contexto da compressão de dados genômicos, compreender a correlação entre as métricas pode fornecer percepções importantes sobre o desempenho das ferramentas avaliadas.

3.1.7 Discussão de outras dimensões

Além das métricas e conjuntos de dados propostos, existem ainda outras dimensões a serem padronizadas, como o método de execução da ferramenta de compressão em relação à interface de execução (linha de comando ou janela) e a ordem dos argumentos, além de definir os intervalos ideais de tempo para a coleta de dados da memória e uso da CPU. Da mesma forma, será discutido e padronizado quais informações devem estar disponíveis ao leitor para garantir a reprodutibilidade do experimento e como os resultados serão apresentados, por exemplo, usando medidas de tamanho em *bytes* em vez de *kilobytes*, *Megabytes* ou unidades maiores desta medida. Por exemplo, para compressão de sequências genômicas com modelos preditivos, a necessidade de disponibilidade do modelo de predição pode ser comparado ao da compressão referencial *pairwise*, ou seja, tanto a sequência de referência quanto o modelo preditivo devem estar sempre disponíveis para o processo de descompressão. Assim, o tamanho final da compressão se dá pela seguinte equação:

$$tamanhoComprimido = \left(\sum_{i=1}^{N} |file_i| \right) + |sequenciaComprimida| \tag{3.11}$$

Onde $tamanhoComprimido$ é a soma total dos tamanhos de todos arquivos auxiliares gerados no processo de compressão ($\sum_{i=1}^{N} |file_i|$) e que são necessários para a descompressão, somado com o tamanho da sequência genômica comprimida ($|sequenciaComprimida|$) (cf. Equação 3.11). Considere que uma sequência pós-compressão, denominada $sequenciaComprimida$, tem 5000 *bytes*. Adicionalmente, considere que há três arquivos auxiliares necessários para a descompressão, cada um com tamanhos de 200, 300 e 150 *bytes*, respectivamente. Aplicando a equação 3.11, o tamanho total comprimido seria 5000 + 200 + 300 + 150 = 5650 *bytes*.

Além disso, aos autores que propõem ferramentas especializadas de compressão de dados genômicos, sugere-se destacar os tamanhos da sequência de referência ou modelo preditivo e quaisquer arquivos necessários para descompressão. Assim, deve informar ao leitor que é necessário levar em conta o tamanho da sequência de referência ou do modelo preditivo gerado para o cálculo final dos desempenhos de compressão e descompressão.

3.2 Método: Ferramenta de Compressão Baseada em Modelo Preditivo

Este segmento da pesquisa foca no desenvolvimento e na implementação de uma ferramenta especializada de compressão para sequências genômicas. Esta ferramenta é fundamentada em um modelo preditivo, cuja eficiência de compressão está ligada à sua capacidade de prever acuradamente o próximo símbolo na sequência genômica e armazenar essas predições ou erros de forma eficiente. Os modelos preditivos empregados são baseados em algoritmos, que podem ser estáticos ou adaptativos, e são essenciais para determinar as probabilidades de subsequências dentro do contexto genômico.

No processo de compressão e descompressão de dados genômicos (cf. Figura 3.1), a abordagem adotada envolve múltiplas etapas cruciais. Inicialmente, o arquivo FASTA é lido, de onde se extrai o cabeçalho e as bases genéticas não representadas pelos nucleotídeos padrão adenina (A), timina (T), citosina (C) e guanina (G), armazenando essas informações em um *Codebook*. Segue-se a etapa de predição, na qual um algoritmo preditor é utilizado para predizer o próximo símbolo na sequência genética, resultando em uma sequência codificada em *bits*. Essa sequência, juntamente com informações adicionais são armazenadas

no *Codebook*. O *Codebook* é então comprimidos, utilizando a ferramenta *7-zip*, para minimizar o espaço de armazenamento necessário.

No estágio de descompressão, o arquivo comprimido é primeiro extraído para recuperar o *Codebook* e a cadeia de *bits*. Utilizando as informações contidas no *Codebook*, a cadeia de *bits* é decodificada para reconstruir a sequência genética original. Essa sequência é então complementada com o cabeçalho e as bases não-ATCG previamente armazenadas, reconstruindo assim o arquivo FASTA original.

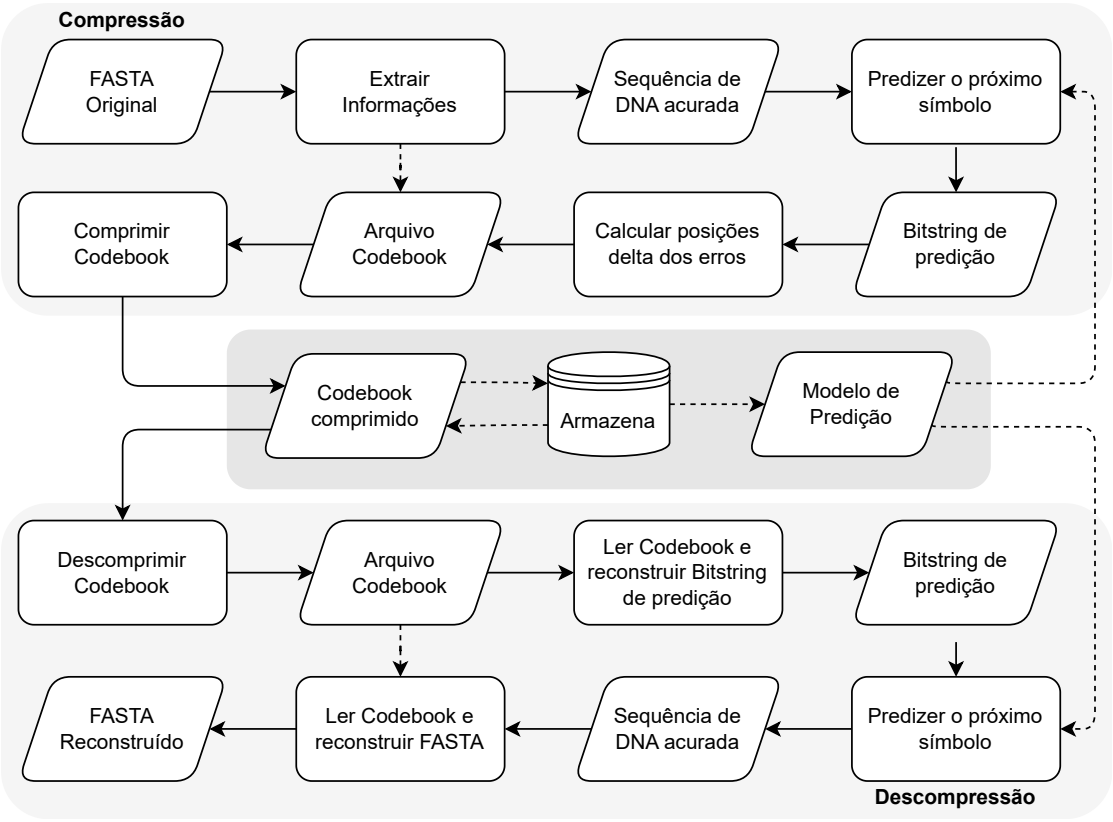

Figura 3.1: Fluxograma do Processo de Compressão e Descompressão de Dados Genômicos por ferramenta baseada em modelo preditivo: este diagrama ilustra as etapas sequenciais envolvidas na compressão e subsequente descompressão de arquivos FASTA. As fases abrangem desde a leitura e extração de informações necessárias do arquivo FASTA, passando pela predição e codificação, até a reconstrução final do arquivo FASTA, após a descompressão – o autor 2024

3.2.1 Compressão de dados genômicos

A seguir, será realizada a explanação detalhada de cada fase deste processo, enfatizando a funcionalidade e a importância de cada etapa no contexto do mé-

todo em questão. Esta exploração abrangerá desde a leitura inicial do arquivo FASTA até a compressão final dos dados após a codificação, proporcionando uma compreensão abrangente dos mecanismos envolvidos. Quando mencionado FASTA nesse processo, subentende-se que se trata de arquivos no formato FASTA quanto multi-FASTA.

A fase de pré-processamento é uma etapa importante no processo de compressão baseado em modelo preditivo. Esta fase envolve a leitura do arquivo FASTA e a preparação e organização dos dados genômicos que serão submetidos à compressão. O método foi desenhado para processar sequências genômicas escritas em arquivos no formato *FASTA*, configurando-os como entradas adequadas para o algoritmo de compressão.

Leitura do arquivo FASTA e geração do *Codebook*

O processo de pré-processamento inicia-se com a leitura linha a linha do arquivo *FASTA* de entrada. Este formato de arquivo baseado em texto é composto por duas partes: 1) o cabeçalho da sequência cujo conteúdo inicia com o símbolo > (maior que) seguido do código identificador único (*Accession Number*) ou do nome do organismo biológico, além de outras informações opcionais; e 2) a representação da sequência biológica composta por uma sequência de símbolos que caracterizam as bases nitrogenadas. Conforme já mencionado neste trabalho, devido à atual tecnologia de sequenciamento e do software de montagem de dados genômicos, um arquivo FASTA pode apresentar longas cadeias de símbolos N que correspondem a locais genômicos inacessíveis ou regiões repetitivas altamente redundantes. Para ambos os casos, uma estratégia de simplificação de sequência é usada para reduzir o espaço necessário para a representação dos dados.

O *Codebook* desenvolvido para este trabalho está estruturado em quatro seções distintas, que utilizam uma ou mais linhas, tendo cada uma das seções uma finalidade específica no contexto da compressão e descompressão de dados genômicos (cf. Tabela 3.2). Essa Tabela mostra a estrutura do *Codebook*. A primeira linha do *Codebook* constitui o "Cabeçalho FASTA", que é restrito a uma única linha. Nas linhas subsequentes, cada linha representa uma entrada única no *Codebook*, onde "Posição Delta (p)" indica a diferença de posição na sequência original para a ocorrência do símbolo ou cadeia não-ATCG, "Símbolos" identifica o símbolo não-ATCG ou o início da cadeia de símbolos não-ATCG, e "Compri-

mento (l)" especifica o número de repetições para um símbolo individual ou o comprimento de uma cadeia de símbolos.

Seção 1: Cabeçalho. Esta Seção, estabelecida na linha 1 do *Codebook*, é dedicada ao armazenamento do cabeçalho do arquivo FASTA. O cabeçalho, que inicia com o símbolo >, contém informações cruciais como o código identificador único ou o nome do organismo biológico, além de outras informações relevantes. A importância desta Seção reside na necessidade de preservar metadados essenciais para a interpretação correta das sequências genômicas para a reconstrução do arquivo FASTA.

Seção 2: Informações Gerais. A segunda Seção do *Codebook*, disposta na linha 2, armazena um conjunto de informações gerais em formato *JSON*, ou seja, chaves e valores em uma estrutura de dicionário. Estas informações incluem, entre outras, o número de colunas na sequência FASTA, detalhes sobre a transformação de dados aplicada e a frequência das bases nitrogenadas. A representação em *JSON* foi escolhida pela sua flexibilidade e capacidade de armazenar dados estruturados de forma compacta e legível. Nos dados *JSON* fornecidos como exemplo na Tabela 3.2, as chaves cl, sl e bl representam respectivamente "colunas", "*sequence length*" e "*bitstring length*". Os valores correspondem ao número de colunas onde ocorre as quebras de linhas do arquivo FASTA, ao comprimento original da sequência genômica e ao comprimento final da *bitstring* para o armazenamento (ELIAS, 1975). Quaisquer dados essenciais para a reconstrução do arquivo FASTA podem ser organizados e armazenados na Seção 2 do *Codebook* como pares de chave e valor utilizando a estrutura *JSON*.

Seção 3: Símbolos Não ATCG. Esta Seção é destinada a registrar, a partir da linha 3 até N, as posições delta dos símbolos não ATCG encontrados na sequência, juntamente com o comprimento dessas ocorrências. O uso da posição delta, que indica a diferença de posição em relação ao símbolo anterior, permite uma representação mais eficiente dos dados, reduzindo o espaço necessário para armazenar informações sobre a localização de cada símbolo do *Alfabeto Genômico diferente de {A, T, C, G}*. Os símbolos não-ATCG são descritos no arquivo *Codebook* juntamente com o valor das suas posições existentes na sequência original. Os símbolos repetidos são representados como uma única entrada no *Codebook*. Cada símbolo não-ATCG ou cada cadeia de símbolos não-ATCG repetitivos é

Tabela 3.2: Estrutura Detalhada do *Codebook* com dados de exemplo – o autor 2024

Linha	**Dados**		
Seção 1: Cabeçalho FASTA			
1	*>gi|2444159403|gb|OP620800.1| Homo sapiens...*		
Seção 2: Informações Gerais			
2	`{"cl": 60, "sl": "16530", "bl": "1059", ...}`		
Seção 3: Símbolos Não ATCG			
	Posição Delta (p)	**Símbolo (s)**	**Comprimento (l)**
3	p_1	s_1	l_1
4	p_2	s_2	l_2
...	...	...	...
N	p_n	s_n	l_n
Seção 4: Erros de Predição			
N_1	*0001 0101 0000* ***00***		
...	*0101 0101* ***01***		
N_n	*1001 0101* ***11***		

descrito no *Codebook* como uma tríade, uma em cada linha, de valores (p, s, l), onde p é a posição delta na qual o símbolo ou cadeia ocorre na sequência original, s é o símbolo e l é o comprimento da cadeia.

Seção 4: Erros de Predição. Embora não foi tratado ainda sobre a predição do próximo símbolo, a última Seção do *Codebook* é dedicada aos erros de predição. Nesta Seção, as posições delta onde ocorrem erros são representadas em formato binário, seguidas de uma representação literal de dois *bits* do símbolo errôneo (cf. Algoritmo 9). A escolha de representar cada dígito da posição delta com 4 *bits* é baseada no cálculo $\lceil \log_2(10) \rceil$, que determina o número mínimo de *bits* necessários para representar 10 dígitos (0 a 9). Por exemplo, considere um erro de predição na posição delta 150. Essa posição é representada como uma sequência binária de 4 *bits* para cada dígito, resultando em *000101010000* para o número 150. Supondo que nessa posição o modelo de predição errou, e o símbolo literal seja *A* (adenina), representado por *00*, a entrada correspondente no *Codebook* seria *1=0001, 5=0101, 0=0000* e *A=00*, assim, tem-se a sequência binária *00010101000000*. Esse processo de codificação assegura uma representação precisa de erros de predição, permitindo uma correção efetiva durante a descompressão, pois os 2 *bits* derradeiros de cada entrada sempre representam o

símbolo literal. A leitura desta Seção está ancorada na busca por linhas contendo apenas os valores binários 0 e 1.

Quando todas as informações pertinentes são compiladas nas respectivas seções do *Codebook*, o mesmo é armazenado em um arquivo de texto (ASCII) e posteriormente comprimido usando o compressor de uso geral *7-zip* (PAVLOV, 2023) que usa codificação PPMd otimizada para texto. Cada seção do *Codebook* desempenha um papel crucial na sua estruturação, garantindo que a descompressão da sequência genômica ocorra com precisão.

Extração das características

O processo para extração de características das sequências genômicas foi estruturada para abordar a complexidade e singularidade dos dados genéticos. O foco principal deste estágio foi expandir a representatividade dos dados, ultrapassando a abordagem convencional de codificação *one-hot* dos símbolos nucleotídicos ATCG. Assim, um conjunto diversificado de algoritmos para a extração de características adicionais foram implementados, visando destacar aspectos particulares das sequências genômicas. Estas características, incorporadas ao vetor de atributos, visavam enriquecer o conjunto de dados para análises preditivas. Todavia, é importante ressaltar que tais extrações foram realizadas de forma exploratória (cf. Anexo A). Posteriormente, constatou-se que o vetor *one-hot* em sua forma mais simples com as representações de ATCG, apresentava desempenho superior, levando à decisão de não utilizar as características adicionais no modelo final.

Padronização e Normalização dos Dados

Embora a padronização e a normalização sejam etapas cruciais no pré-processamento de dados para muitos métodos de aprendizado de máquina, neste estudo específico, adotou-se a codificação *one-hot* para as características genômicas. A codificação *one-hot* transforma as categorias em uma representação binária —um vetor de zeros e uns— que já está, por sua natureza, normalizado. Neste contexto, cada sequência genômica é representada por um vetor onde apenas um elemento é "1" e todos os outros são "0", indicando a presença de um determinado nucleotídeo. Esta abordagem elimina a necessidade de padronização adicional, como o *Z-score*, uma vez que os dados já estão em uma escala uniforme e os atributos possuem a mesma importância relativa. Além

disso, a estrutura binária desta codificação assegura que os algoritmos de aprendizado de máquina não sejam influenciados por variações na escala dos dados, mantendo a integridade e a interpretabilidade das informações genômicas. A codificação *one-hot* oferece uma solução eficiente e adequada para o tratamento de dados genômicos em análises preditivas e modelagem estatística.

Apesar de a abordagem adotada ter eliminado a necessidade de padronização adicional dos dados, um processo complementar de extração de características, para expandir a matriz proveniente da codificação *one-hot*, está descrito no Anexo A. Neste contexto, características adicionais foram extraídas, visando enriquecer o conjunto de dados com uma representação mais abrangente e detalhada. Devido à heterogeneidade inerente a estas características suplementares, as quais podem apresentar variações significativas em termos de magnitude e escala, tornou-se imperativo o emprego de técnicas de padronização e normalização. Este procedimento assegura que todas as características, incluindo as recém-incorporadas, estejam normalizadas e padronizadas em uma escala uniforme, mitigando assim possíveis distorções oriundas de discrepâncias nas escalas dos valores. A implementação da padronização, efetuada por meio do cálculo do *Z-score*, e da normalização *Min-Max*, foi realizada para integrar estas variáveis adicionais aos dados originais do *one-hot encoding*. Esta estratégia é fundamental para preservar a consistência e melhorar a eficácia dos algoritmos de aprendizado de máquina utilizados nas análises subsequentes, conforme descrito no Anexo A.

Treinamento e validação para otimização do modelo de predição

Durante a fase de seleção e avaliação do modelo, dois modelos de rede de aprendizagem profunda das ferramentas *baseline*, além do modelo proposto, foram treinados e testados utilizando um conjunto específico de sequências genômicas. Os modelos considerados incluem variantes baseadas em CNN e LSTM, bem como combinações destas com outras técnicas como biLSTM em conjunto com mecanismos de atenção.

Os resultados obtidos indicaram que o modelo com redes *CNN+LSTM* superou as demais em termos de acurácia. Conforme ilustrado na Tabela 3.3, o modelo *CNN+LSTM* alcançou uma acurácia de 99,75%, enquanto a variante *CNN+biLSTM* com mecanismo de atenção registrou uma acurácia de 99,55%. Entretanto, a escolha do modelo proposto (*CNN+Double(biLSTM)*), que exibiu

um equilíbrio entre as classes A, C, G e T, foi fundamentada não apenas pela sua acurácia global de 99,69%, mas principalmente pela sua capacidade de manter uma consistência equilibrada entre as diferentes bases nitrogenadas. Esta característica distingue o modelo proposto em relação ao modelo *CNN+LSTM*, cuja estrutura mais simplificada não apresentava tal equilíbrio entre as classes.

Tabela 3.3: Comparação da acurácia entre modelos de ferramentas *baseline* utilizados no estudo e o modelo proposto. A tabela mostra a acurácia alcançada pelo modelo proposto (*CNN+Double(biLSTM)*) em comparação com o modelo *CNN+LSTM* a variante CNN+biLSTM+AttentionWithContext (Operação de atenção, com vetor de contexto, para dados temporais) – o autor 2024

Modelo	**Acurácia**	**Tamanho (*bytes*)**
CNN+LSTM	99,75%	84.771.760
CNN+Double(biLSTM) (proposta)	99,69%	932.987.296
CNN+biLSTM+AttentionWithContext	99,55%	23.982.880

A escolha do modelo *CNN+Double(biLSTM)*, fundamentou-se principalmente na sua capacidade de capturar eficientemente as características importantes das sequências genômicas. A combinação de *CNN*, para extração de características locais, e duas camadas Bidirecionais de biLSTM, para captura de dependências de longo alcance, mostrou-se eficaz para o conjunto de dados em questão. Portanto, com base nos resultados obtidos e considerando as características específicas das sequências genômicas selecionadas, o modelo *CNN+Double(biLSTM)* foi escolhido como a abordagem mais promissora para a análise subsequente.

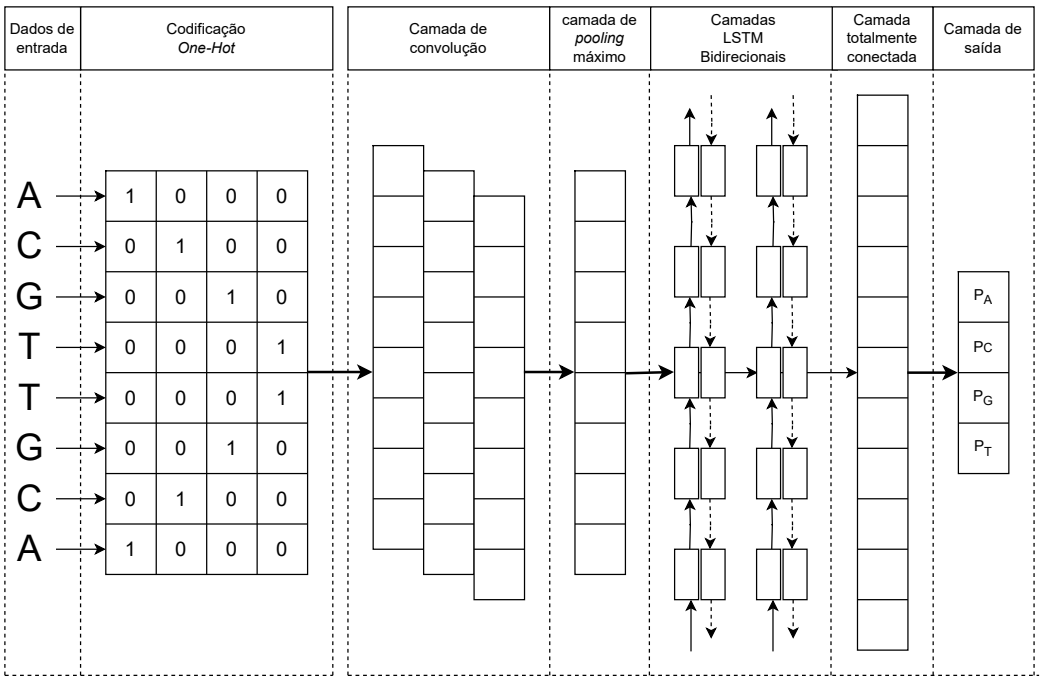

Figura 3.2: Arquitetura da rede neural, utilizada no processo de compressão e descompressão das sequências genômicas, com múltiplas camadas, incluindo processamento convolucional, *pooling* máximo, biLSTM, e camadas totalmente conectadas, com as respectivas entrada e saída de dados. É importante notar que, embora as Camadas de *Dropout* não esteja explicitamente representada na ilustração da arquitetura, estas desempenham um papel relevante na configuração do modelo. A omissão da Camada *Dropout* na figura segue uma convenção comum em representações gráficas de modelos de rede neural, onde elementos como camadas de regularização frequentemente não são mostrados para simplificar a visualização – o autor 2024

A Figura 3.2 apresenta a arquitetura de uma rede neural, composta por várias camadas distintas, cada uma com um papel específico no processamento dos dados genômicos. Inicialmente, os dados genômicos pertencentes ao alfabeto ATCG são inseridos no modelo e submetidos à codificação *One-Hot*, um método eficiente para lidar com variáveis categóricas. Após essa transformação, os dados passam por uma *Camada de Convolução*, que extrai características relevantes utilizando filtros convolucionais. Essa etapa é essencial em tarefas como processamento de imagens e reconhecimento de padrões. Posteriormente, a *Camada de Pooling Máximo* reduz a dimensionalidade dos dados, preservando as características mais importantes, o que ajuda a minimizar o volume de cálculos e a prevenir *overfitting*.

A rede então utiliza duas camadas biLSTM para capturar dependências temporais de longo e curto alcance, importante em aplicações de processamento de linguagem natural e análise de séries temporais. Cada camada biLSTM consiste em duas sub-camadas LSTM, uma processando a sequência de entrada na direção normal (*forward layer*) e outra na direção inversa (*backward layer*), ambas com 256 unidades.

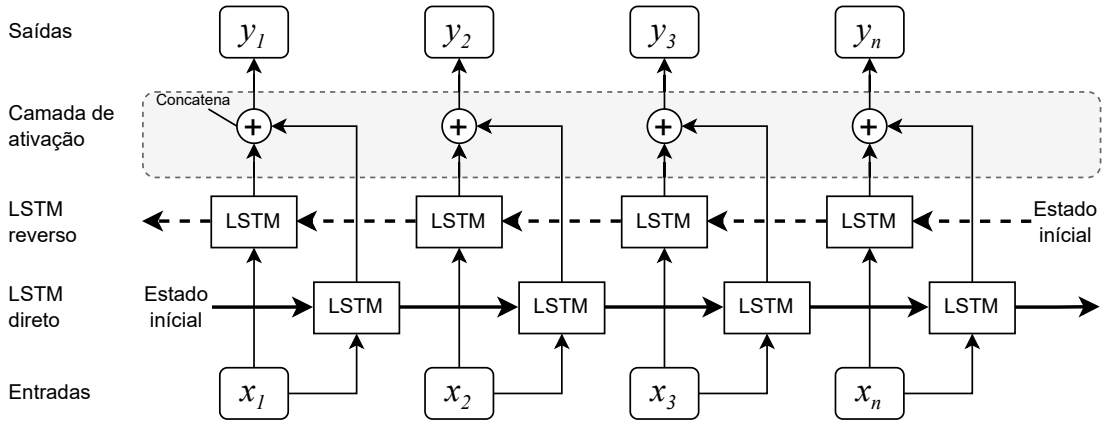

Figura 3.3: Arquitetura de uma Rede Neural biLSTM. As entradas sequenciais são denotadas por $x_1, x_2, \ldots, x_n$, processadas em paralelo por uma camada LSTM direta e uma camada LSTM reversa. A camada LSTM direta propaga a informação ao longo da sequência temporal do início ao fim, enquanto a camada LSTM reversa processa a sequência na ordem inversa. As saídas de cada passo temporal de ambas as camadas são concatenadas e, em seguida, passadas por meio de uma camada de ativação. As saídas resultantes da rede em cada passo temporal são representadas por $y_1, y_2, \ldots, y_n$ – adaptada de (SCHUSTER; PALIWAL, 1997)

Um biLSTM (cf. Figura 3.3), é composto por dois componentes LSTM. O primeiro é o "LSTM direto", que processa os dados sequenciais do início ao fim, isto é, do primeiro passo de tempo até o último. O segundo componente é o "LSTM reverso", que opera na direção oposta: começa pelo último passo de tempo e avança até o primeiro. Após o processamento dos dados por ambos os componentes LSTM, as saídas de cada um são combinadas. Esta combinação é feita por meio da operação de concatenação ao longo da dimensão do canal, efetivamente juntando as informações capturadas em ambas as direções temporais. Esta configuração permite que a rede capture dependências de longo alcance em ambas as direções da sequência de entrada.

Ainda, na arquitetura do modelo proposto, segue-se uma *Camada Totalmente Conectada*, que integra e classifica as características extraídas. Por fim, a *Camada*

de Saída fornece as quatro probabilidades (P_A, P_C, P_G *e* P_T) finais do modelo, uma para cada símbolo do alfabeto ATCG.

A otimização do modelo de predição é uma etapa importante neste método de compressão de sequências genômicas. Para isso, as sequências genômicas pré-processadas são divididas em três conjuntos distintos: 70% para treinamento, 20% para validação e 10% para testes. A escolha desta configuração é fundamentada em práticas amplamente aceitas na área de aprendizado de máquina, que recomendam esta divisão para equilibrar a quantidade de dados disponíveis para ajuste do modelo e a quantidade de dados para validação e teste, minimizando o risco de sobre-ajuste e sub-ajuste (GHOLAMY; KREINOVICH; KOSHELEVA, 2018). O conjunto de treinamento é utilizado para ajustar os parâmetros do modelo, enquanto o conjunto de validação é empregado para aferir a eficácia do modelo e realizar ajustes iterativos nos hiper-parâmetros, visando aprimorar o equilíbrio entre viés (*bias*) e variância (*variance*). Este processo é fundamental para prevenir problemas de sobre-ajuste (*overfitting*) e sub-ajuste (*underfitting*). Por fim, o conjunto de testes é utilizado para avaliar objetivamente o desempenho final do modelo de predição.

A definição dos parâmetros dessa arquitetura de rede foram definidos da seguinte forma:

Tabela 3.4: Especificação das camadas da rede neural.

Camada	**Parâmetro**	**Valor**
Convolução	Quantidade de Filtros	1024
	Tamanho da Janela	24
	Tamanho do Passo	1
Max-Pooling	Tamanho da Janela	3
	Tamanho do Passo	1
Dropout	Probabilidade	0.1
Duas Camadas biLSTM	Número de Neurônios	256
Dropout	Probabilidade	0.2
Totalmente Conectada	Número de Neurônios	1024
Saída Sigmóide	Número de Neurônios	4

Quanto aos inicializadores, os pesos da rede foram inicializados aleatoriamente (*random uniform*) de $-0,05$ até $0,05$, e todos os vieses (*bias*) foram inicialmente definidos como "zeros", ou seja, inicializador que gera tensores inicializados em 0. Além disso, foi utilizado uma estratégia de treinamento em mini-lotes (*batch size*) cujo comprimento de cada sequência de entrada foi de

128 símbolos. A definição desse comprimento foi fundamentada em testes preliminares realizados com diferentes comprimentos, incluindo 32, 64 e 256. Os testes conduzidos revelaram que o comprimento de 128 símbolos ofereceu uma performance do modelo pouco melhor que as demais. Com isso, buscou-se minimizar a entropia cruzada binária (*log loss*) multitarefa média no conjunto de dados de treinamento. A função de perda logarítmica, também conhecida como *log loss*, é uma métrica usada em modelos de classificação binária para quantificar o desempenho do modelo. Esta métrica mede o quão próximas as probabilidades previstas do modelo estão dos rótulos precisos, 0 ou 1. Valores mais baixos de *log loss* indicam um modelo com melhores previsões, enquanto valores mais altos sugerem previsões imprecisas. O *dropout* foi avaliado no final de cada período (*epochs*) de treinamento para monitorar a convergência do modelo.

A rede experimental foi configurada para processar até 100 épocas, com o treinamento sendo interrompido de forma antecipada pela implementação de um mecanismo de *early stopping*. Este mecanismo estava ajustado para interromper o treinamento após 3 épocas consecutivas sem melhora no desempenho do modelo. Devido a esta abordagem adaptativa, o tempo total de treinamento variou, mas cada época individual de treinamento manteve uma duração aproximada de 30 minutos. Assim, o tempo total despendido no experimento para o treinamento e validação do modelo foi condicionado pelo critério de parada antecipada, garantindo um equilíbrio entre o desempenho ótimo do modelo e a eficiência do treinamento.

Teste do modelo de predição

Na fase de teste do modelo de predição, o modelo previamente treinado foi submetido a um processo de avaliação de desempenho, visando verificar a precisão alcançada na predição das sequências genômicas mitocondriais. Para tal, foi empregado um conjunto de dados específico, separado durante a fase de pré-processamento e exclusivamente reservado para este fim. Este conjunto, denominado conjunto final de teste, ou seja 10% das sequências reservadas ao teste, foi utilizado para avaliar a capacidade do modelo em prever corretamente o próximo símbolo genômico.

Além disso, nesta etapa crucial, foram selecionadas e aplicadas diversas métricas de avaliação de desempenho. Estas métricas incluem, mas não se

limitam a, medidas de acurácia, as quais são importantes para uma análise abrangente da eficácia do modelo. A utilização destas métricas permite uma avaliação quantitativa, oferecendo percepções sobre a confiabilidade e precisão do modelo de predição desenvolvido. Tais métricas e os resultados obtidos são discutidos detalhadamente em seções subsequentes.

Codificação dos erros da predição

A ideia básica dessa etapa é gerar uma sequência de *bits* de acordo com os erros e acertos da predição do próximo símbolo (cf. Figura 3.4). A estratégia se dá da seguinte forma: cria-se uma sequência binária, inicial, de comprimento 256 que representa literalmente os primeiros 128 símbolos da sequência genômica codificada com codificação ingênua, ou seja, 2 *bits* por base nitrogenada ($A = 00, C = 01, G = 10$ *e* $T = 11$). A partir disso, a predição começa a ser realizada a partir do símbolo que está na posição 129 da sequência genômica, uma vez que os primeiros 128 símbolos fazem parte do conjunto (contexto) de entrada para o processo de predição. Se o modelo de predição acertar qual é o símbolo daquela posição, um *bit* de valor 1 é adicionado à sequência de *bits*. Em contrapartida, caso o modelo erre a predição, são adicionados 3 *bits* à sequência de *bits*. Isso se deve ao fato de que é necessário manter presente, na sequência de *bits*, a informação exata da base nitrogenada não predita. Dessa forma, adiciona-se a sequência de *bits* um marcador representado pelo *bit* 0 seguido por um par de *bits* que representam cada base nitrogenada. As bases nitrogenadas não preditas têm a seguinte representação binária: $A = 000, C = 001, G = 010$ *e* $T = 011$.

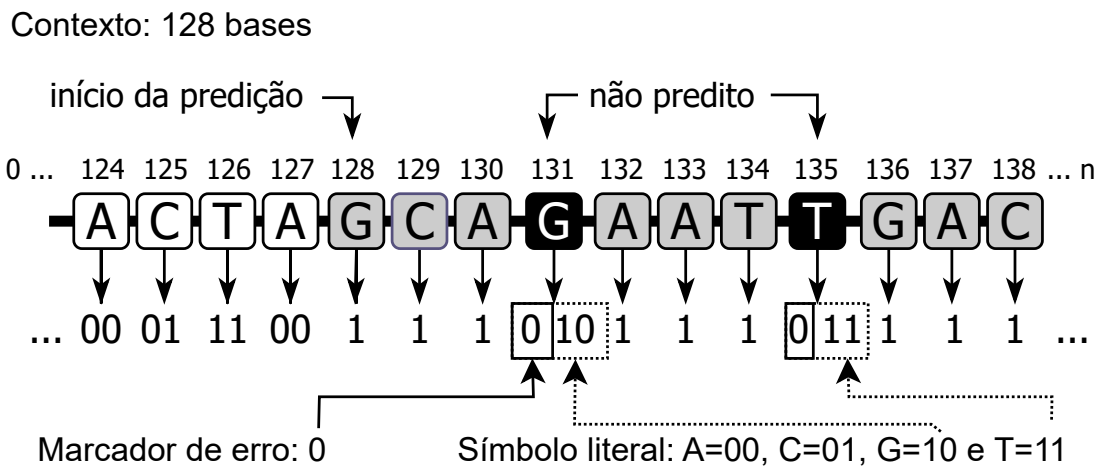

Figura 3.4: Processo de predição do próximo símbolo – o autor 2024

A Figura 3.4 ilustra como a sequência de *bits* é construída a partir dos acertos e erros do modelo preditivo. Além disso, a cadeia de *bits* de comprimento 256, ou seja, representação binária dos primeiros 128 símbolos levando em conta 2 *bits* por base, é concatenada como prefixo na cadeia de *bits* final. Logo, as informações referentes aos símbolos existentes nas primeiras 128 posições da sequência genômica são preservadas. Tal informação será usada no processo de descompressão.

Transformação e compressão final dos Dados

No processo final de transformação dos dados, após análises e experimentações, optou-se pela codificação da *bitstring*, onde estão anotados em formato binário, os acertos e erros de predição (cf. Algoritmo 9). Esta abordagem, que se mostrou mais eficiente para o *dataset* selecionado, envolve a codificação da posição delta (δ) onde ocorre o erro, juntamente com o símbolo literal correspondente. Todos estes elementos são codificados em binário, formando a base para o processo de compressão.

Inicialmente, a técnica de Run-Length Encoding (RLE) *binary-based* (ELIAS, 1975) foi explorada como uma possível abordagem (cf. Anexo B) (cf. Algoritmo 2). No entanto, após avaliações, observou-se que a abordagem de codificar diretamente a *bitstring* de acertos e erros no *Codebook*, oferecia resultados superiores em termos de eficiência para o conjunto de dados em questão. Por esse motivo, a descrição detalhada e os resultados obtidos com o uso do RLE foram realocados para os anexos, como parte de uma fase exploratória.

Aqui é descrito o processo responsável pela transformação da *bitstring* contendo os erros e acertos de predição em uma lista binária, a qual registra as posições delta dos erros e os respectivos símbolos literais das bases nitrogenadas. Este processo inicia com a definição de uma lista vazia *data*, que armazenará a representação binária resultante. O processo percorre a *bitstring*, identificando cada símbolo. Quando um símbolo "1" (acerto da predição) é encontrado, o índice avança. No entanto, ao detectar um símbolo "0" (erro da predição), calcula-se o delta, que é a diferença entre o índice atual e a soma do último índice, onde ocorreu o erro, com o comprimento do erro anterior. Este delta representa a posição relativa do erro atual em relação ao anterior. Em seguida, captura-se o erro, representado pelos dois *bits* subsequentes na *bitstring*. O delta é convertido para binário e concatenado com os *bits* do erro, formando assim um

elemento da lista *data*. Esse processo continua até que toda a *bitstring* seja analisada, conforme descrito anteriormente na seção onde é tratado da construção do *Codebook*.

Considere a *bitstring* "11111101011111110111111111100011111" a ser transformada e anotada na Seção 4 do *Codebook*. Cada "1" na *bitstring* simboliza um acerto na predição, ao passo que cada "0" indica um erro. À medida que um erro é detectado, o delta —que representa a distância desde o último erro— é calculado, e os dois *bits* subsequentes ao "0" denotam o erro propriamente dito. Considerando as atribuições para os nucleotídeos $A, C, G\ e\ T$ e para os dígitos de 0 a 9, definimos que:

- Cada nucleotídeo é representado por 2 bits, pois $2^2 = 4$ cobre as 4 possíveis bases (A, C, G, T).
- Cada número de 0 a 9 é representado por 4 bits, pois $2^4 = 16$ cobre as 10 possíveis opções de dígitos (0 a 9).

Com base nessas atribuições, constrói-se a codificação *Bitstring as Codebook* com os valores de delta (Δ) seguidos pelos *bits* correspondentes ao erro. A *bitstring* é então analisada conforme descrito a seguir:

- A análise inicia-se percorrendo a *bitstring* até encontrar o primeiro *bit* 0:
  - **Índice do erro:** 6 ($6 - 0 = \Delta 6$)
  - **Erro:** 10 (conforme as atribuições $A = 00$, $C = 01$, $G = 10$, $T = 11$)

  Primeira entrada no *Codebook*: Sendo este o primeiro erro, o Δ é equivalente à posição do erro, ou seja, 6. Convertendo 6 para binário, obtém-se 0110. A entrada no *Codebook* configura-se como 0110 10.
- A análise prossegue da posição após o primeiro erro até o próximo *bit* 0:
  - **Índice do erro:** 17 ($17 - 6 = \Delta 11$, desde o último erro)
  - **Erro:** 11

Segunda entrada no *Codebook*: Convertendo $\Delta 11$ para binário, obtém-se 0001 0001. A entrada no *Codebook* configura-se como 0001 0001 11.

- Continua-se até o próximo *bit* 0:
  - **Índice do erro:** 30 ($30 - 17 = \Delta 13$, desde o último erro)
  - **Erro:** 00

Terceira entrada no *Codebook*: Convertendo $\Delta 13$ para binário, obtém-se 0001 0011. A entrada no *Codebook* configura-se como 0001 0011 00.

Codebook resultante:

- Primeira linha: "0110**10**" (equivale a delta 6 e erro 01 (Citosina))
- Segunda linha: "0001000**11**" (delta 11, erro 11 (Timina))
- Terceira linha: "0001001**00**" (delta 13, erro 00 (Adenina))

Os passos acima refletem o método pelo qual os deltas são calculados com base nos índices dos erros, e os *bits* de erro são adicionados subsequentemente aos deltas no *Codebook*. Cada delta é calculado relativamente ao erro anterior, não de maneira absoluta desde o início da *bitstring*.

Considere b como a *bitstring* de entrada e n seu comprimento. Define-se a função que gera os dados binários finais da seguinte forma:

Para cada $i \in \{0, 1, \ldots, n-1\}$ em b :

Se $b_i = 1$:

$i \leftarrow i + 1$

Se $b_i = 0$:

$\delta \leftarrow i - \text{last\_delta} - \text{error\_length}$

$\text{last\_delta} \leftarrow i$

$i \leftarrow i + 1$

$\text{error} \leftarrow b_{i+1}\ e\ b_{i+2}$

Adicione Bin(δ) e error à lista d

$i \leftarrow i + 2$

$\text{error\_length} \leftarrow 3$

A saída r é construída por:

$$r = \text{concatenação de } (\text{Bin}(\delta)\ e\ \text{error}) \text{ para cada } \delta \text{ e error encontrados}$$

onde cada concatenação em r representa uma entrada no *Codebook*, separadas por novas linhas, e Bin(δ) é a representação binária de δ, e error são os dois *bits* que seguem b_i na *bitstring* b.

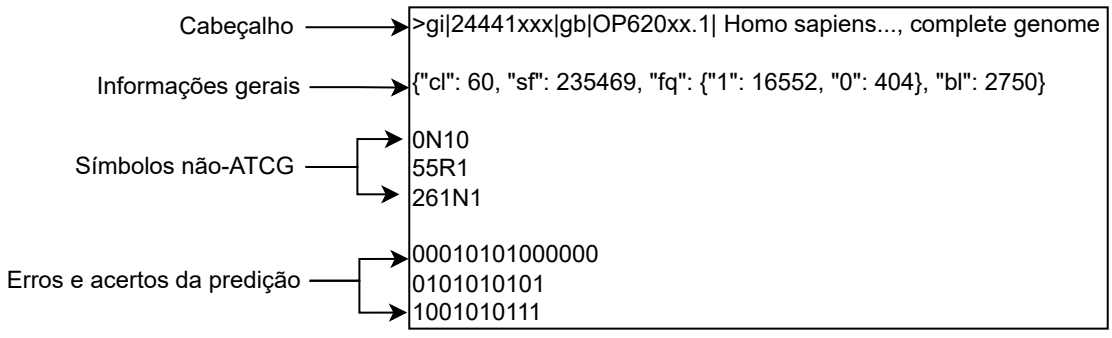

Figura 3.5: Figura ilustrativa demonstrando as quatro seções do Codebook, sendo estas, o cabeçalho, informações gerais, símbolos não-ATCG e os errors e acertos de predição – o autor 2024

Por fim, o Codebook (cf. Figura 3.5), que contém informações para o processo de descompressão, é comprimido usando o compressor de uso geral *7-zip* na configuração mais eficiente para texto, utilizando o argumento "*-m0=PPMd*". Este algoritmo é baseado principalmente no código-fonte PPMdH de *Dmitry Shkarin* (SHKARIN; SUBBOTIN, 2006). O PPMd fornece uma taxa de compressão eficiente para arquivos de texto.

Descompressão dos Dados Genômicos

O processo de descompressão dos dados genômicos, envolve realizar as etapas de compressão de forma reversa. Este procedimento começa com a leitura do arquivo comprimido em formato *7-zip*. Após a abertura do arquivo, o objeto *Codebook* é extraído. Esta etapa é crucial, pois o *Codebook* contém informações detalhadas necessárias para a reconstrução correta da sequência genética.

Uma vez o *Codebook* extraído, o processo prossegue com a leitura da lista de *strings* de *bits*, descrito na Seção 4 do *Codebook* (cf. Tabela 3.2). Essa lista de *strings* de *bits* representa os erros e acertos da predição, formando a base para a reconstrução da sequência genética.

A etapa subsequente envolve a decodificação da *bitstring* de predição. Esta decodificação é realizada por meio de um algoritmo preditor que usa as primeiras posições da *bitstring* como contexto para prever o próximo símbolo da sequência de DNA. O resultado deste processo é a transformação da sequência codificada de volta ao alfabeto original, A, C, G e T, que representa as bases nitrogenadas da sequência genética. Esta reconstrução é fundamental para restaurar a sequência genética à sua forma original e funcional.

A quarta etapa do processo é a leitura do *Codebook* para a inserção do cabeçalho e das bases genéticas não-ATCG na sequência reconstruída, descrito na seções 1, 2 e 3 do *Codebook* (cf. Tabela 3.2). Esta etapa é vital para garantir a restauração completa de todas as informações originais do arquivo FASTA, incluindo informações que não se enquadram nas categorias padrão de bases nucleotídicas.

Finalmente, a sequência genética reconstruída é salva em um novo arquivo com formato FASTA. Este passo conclui o processo de descompressão, resultando em um arquivo FASTA completo e funcional, pronto para ser utilizado em análises e estudos subsequentes.

Limitação na Busca Aleatória

Um aspecto importante que deve ser levado em conta é a limitação inerente à capacidade de realização de busca aleatória no arquivo pós-compressão (em estado comprimido) para encontrar coordenadas ou *substrings*. Esta restrição decorre da necessidade de descompressão integral da sequência genômica para a leitura de porções de dados. Tal exigência é explicada pelo fato de que o símbolo subsequente depende do contexto anterior —cadeia de símbolos de determinado tamanho— que foi aprendido pelo modelo durante o processo de treinamento. Assim, cada símbolo predito faz parte do contexto do próximo símbolo a se predizer. Consequentemente, a estrutura de dados comprimida não permite a extração direta ou a busca aleatória de informações sem antes passar por um processo completo de descompressão, restringindo assim a busca e análise de dados em arquivos comprimidos.

Conclusão do Capítulo

Até o momento, foi apresentada uma abordagem detalhada como proposta de protocolo para a avaliação do desempenho de algoritmos de compressão e descompressão de dados, com ênfase particular nos dados genômicos. Inicialmente, foi delineado os critérios para a avaliação de desempenho, que incluíram a seleção de conjuntos de dados apropriados para os testes e a definição de métricas específicas para as avaliações. A discussão avançou com a proposição de uma ferramenta de compressão e descompressão de dados genômicos, fundamentada em um modelo preditivo. Esse modelo incorpora uma estratégia de codificação para erros e acertos de predição, armazenando-os diretamente no *Codebook*.

Este capítulo conclui com uma explanação detalhada dos métodos propostos, das métricas de avaliação de desempenho e da ferramenta de compressão baseada em modelo preditivo, estabelecendo uma fundação apropriada para as propostas apresentadas. É crucial ainda, proceder com a análise do método de compressão, bem como a preparação e seleção de conjuntos de dados para testes, para compreender as capacidades e limitações da ferramenta de compressão desenvolvida. Os resultados destas avaliações e comparações são explorados no próximo capítulo, onde apresenta-se uma análise comparativa entre as ferramentas de compressão avaliadas. Esta análise oferece percepções importantes

sobre a eficácia, eficiência e viabilidade das diversas técnicas de compressão no contexto da predição do próximo símbolo em sequência genômica.

4

Testes e avaliação de desempenho

No âmbito da presente investigação, os testes de desempenho assumem uma posição central, dada a sua relevância na avaliação e validação da eficácia do Método de compressão. Estes testes determinam a viabilidade e a aplicabilidade prática do Método em comparação com as ferramentas existentes no domínio da compressão de sequências genômicas, baseadas em modelos preditivos. A principal motivação destes testes é estabelecer um quadro comparativo que não apenas destaque as capacidades do Método, mas também forneça percepções significativas sobre o seu desempenho em cenários práticos.

O propósito desta seção é realizar uma comparação de desempenho entre o compressor de genoma desenvolvido e duas ferramentas *baseline* da literatura, com o intuito de verificar a viabilidade do modelo de predição e do método de codificação adotados frente às soluções existentes. Esta análise é essencial para estabelecer um *benchmark* e contribuir para o progresso no campo da compressão de dados genômicos. Por meio de testes de desempenho, esta seção visa esclarecer questões fundamentais relacionadas à precisão e viabilidade do compressor desenvolvido, oferecendo uma avaliação completa das suas capacidades e limitações. Além disso, busca-se evidenciar a relevância do método de avaliação de desempenho para ferramentas de compressão de genomas introduzido neste trabalho, e sua contribuição para a área de compressão de dados genômicos.

4.0.1 Processo de Comparação Entre as Ferramentas

O processo adotado para comparar a ferramenta desenvolvida com as ferramentas *baseline* está delineado nesta seção, seguindo o protocolo de avaliação de desempenho desenvolvido nesse trabalho. Este protocolo estabelece um conjunto de critérios e métricas que avalia de forma objetiva as ferramentas em questão. As métricas de comparação, previamente descritas, incluem: *Bits* Por Base, Economia Média de Espaço, Taxa Média de Compressão, Tempo Médio de Compressão e Tempo de Descompressão, Uso Médio ($\bar{x}$) de Memória e CPU. Tais métricas fornecem uma base quantitativa para a análise comparativa, permitindo uma avaliação do desempenho.

Não se faz necessário reiterar detalhadamente cada aspecto do protocolo, pois este já foi exposto nas seções anteriores deste trabalho. Em vez disso, enfatiza-se aqui a aplicação desse protocolo como um guia para a execução e análise dos testes de desempenho. Esta abordagem não apenas reforça a consistência metodológica, mas também assegura que a comparação entre a ferramenta desenvolvida e as ferramentas *baseline* seja realizada de forma estruturada e sistemática. Portanto, a presente seção se concentra na aplicação prática das métricas e critérios já estabelecidos, destacando a relevância do protocolo de avaliação de desempenho desenvolvido para este tipo de pesquisa.

4.0.2 Descrição das Ferramentas *Baseline*

Aqui procede-se à descrição das duas ferramentas *baseline* selecionadas para fins de comparação de desempenho de compressão. A escolha destas ferramentas foi fundamentada com base em critérios específicos, que incluíram a relevância no campo da bioinformática, a aplicabilidade em compressão de sequências genômicas, e a inovação no uso de técnicas de aprendizado de máquina. A primeira ferramenta, denominada *DeepDNA* (WANG et al., 2018), emprega uma combinação de redes CNN e LSTM. Esta abordagem é amplamente reconhecida por sua eficácia na modelagem de sequências temporais, o que é importante na análise de sequências genômicas.

A segunda ferramenta, ainda sem nome formal, representa um avanço significativo no campo, implementando uma arquitetura composta por redes neurais convolucionais, biLSTM e um mecanismo de atenção, conhecido como *AttentionWithContext* (BALDI et al., 1999). Esta configuração complexa visa aprimo-

rar a capacidade de reconhecimento de padrões e a eficiência na compressão de sequências genômicas, oferecendo uma abordagem inovadora no processamento e análise de dados genéticos. Ambas as ferramentas selecionadas para os testes de desempenho já foram discutidas no Capítulo 2.

Embora os autores dessas ferramentas não tenham disponibilizado uma implementação completa e prontamente utilizável, mas apenas apresentado os resultados finais em termos de *bits* por base baseados em cálculos de entropia, tornou-se imperativo para o presente estudo desenvolver uma codificação aritmética para as predições geradas por ambas as ferramentas. Esta etapa adicional, implementada no contexto deste trabalho, é indispensável para uma comparação efetiva de compressão. A inclusão da codificação aritmética é uma adaptação necessária para possibilitar uma avaliação realista e comparativa do desempenho de compressão, alinhando as ferramentas *baseline* ao contexto específico desta investigação. Tal abordagem enfatiza a importância de não apenas confiar nos resultados publicados, mas também de adaptar e aplicar abordagens existentes para atender às necessidades específicas de um estudo comparativo.

4.0.3 Seleção do *Dataset*

Para a realização dos testes de compressão, optou-se por utilizar sequências do organismo da mesma espécie que foi empregada nos trabalhos dos autores das ferramentas *baseline* mencionadas anteriormente (WANG et al., 2018; BALDI et al., 1999). Assim, selecionou-se um conjunto de mil (1.000) sequências genômicas mitocondriais, com tamanho médio de 16.924, 52 *bytes*, extraídas da base de dados do NCBI utilizando o recurso interno "*Nucleotide*". A busca foi realizada utilizando os termos "*mitochondrion*", "*Homo sapiens*", "*complete genome*"e "*circular DNA*". Cada termo possui uma relevância específica no contexto da seleção do *dataset*: ***Mitochondrion***: este termo foi utilizado para garantir que as sequências genéticas fossem especificamente mitocondriais, focando no DNA localizado na mitocôndria; ***Homo sapiens***: a especificação de "*Homo sapiens*"assegura que todas as sequências genômicas selecionadas sejam de origem humana; ***Complete genome***: este termo foi utilizado para filtrar as sequências que representam genomas completos; ***Circular DNA***: o termo "*circular DNA*"refere-se à estrutura física do DNA mitocondrial, que é tipicamente circular e não linear, como o DNA encontrado no núcleo das células. A inclusão deste termo na busca visou assegurar que as sequências obtidas correspondessem à natureza estrutural do

DNA mitocondrial. Cada um desses termos foi escolhido para definir e refinar o conjunto de dados utilizado, assegurando que as sequências genéticas fossem pertinentes para os propósitos dos testes de compressão.

Após a seleção do conjunto de dados de sequências genômicas mitocondriais, foi analisado o perfil de bases nitrogenadas presentes nas sequências. A contagem total das bases nitrogenadas nas 1.000 sequências foi de 16.562.554 bases, distribuídas da seguinte forma: Adenina (A) com 5.109.372 bases, Citosina (C) com 5.171.876 bases, Guanina (G) com 2.171.972 bases e Timina (T) com 4.085.334 bases. Os percentuais relativos de cada base nitrogenada no conjunto analisado são: Adenina (A) representa 30,84%, Citosina (C) 31,22%, Guanina (G) 13,11% e Timina (T) 24,66%.

Posteriormente, o conjunto de sequências mitocondriais foi dividido de maneira aleatória em três grupos distintos para diferentes finalidades referentes ao modelo de predição, sendo: 70% das sequências alocadas para o conjunto de treinamento, 20% para o conjunto de validação. Da mesma forma, os últimos 10%, com 100 sequências e totalizando 1.656.159 bases nitrogenadas, foram utilizados para os testes de compressão e descompressão das sequências mitocondriais.

4.0.4 Ambiente de Execução dos Testes de Compressão

Os testes de compressão foram planejados e executados, aderindo ao protocolo de comparação estabelecido. Para o treinamento, validação e teste dos modelos preditivos, foi empregado o uso do *Google Colab* (COLABORATORY, 2024), especificamente utilizando a *GPU Tesla 4* com 16GB de RAM na GPU e 12GB de memória RAM em disco. Esta escolha de plataforma foi motivada pela necessidade de alto poder computacional e eficiência no processamento, características essenciais para o manejo efetivo de grandes conjuntos de dados genômicos.

Para o desenvolvimento e implementação dos experimentos de compressão, utilizou-se a linguagem de programação *Python* (PYTHON, 2024), com o auxílio da biblioteca *Keras* (CHOLLET et al., 2015) para facilitar a construção, treinamento e validação dos modelos de aprendizado profundo. A escolha de *Python* e *Keras* deve-se à sua ampla adoção na comunidade científica e à sua capacidade de suportar a rápida prototipagem e execução eficiente em ambientes com recursos computacionais avançados, como o *Google Colab*.

Após a fase de treinamento, os testes de compressão foram conduzidos utilizando um *MacBook* Pro (Retina, 13-inch, Early 2015), Processador 2,7 GHz Intel Core i5 Dual-Core com 8GB de memória RAM. Esta decisão foi tomada para assegurar que os testes refletissem um cenário de uso realista, por exemplo os laboratórios de menor porte, onde recursos computacionais mais limitados são frequentemente empregados. Esta etapa é importante para avaliar a viabilidade prática do modelo em condições de *hardware* comumente disponíveis, principalmente em laboratórios de análises genômicas supracitados.

4.0.5 Análise de Acurácia dos Modelos

Para uma avaliação quantitativa do desempenho de cada modelo, foram geradas matrizes de confusão. Estas matrizes proporcionam uma visão clara sobre a acurácia de cada modelo, permitindo a comparação direta entre o modelo *CnnDoubleBiLstm* (modelo de rede da proposta) e os modelos das ferramentas *baseline CnnLstm* utilizado na ferramenta *DeepDNA* e *CnnBiLstmAtt*. As matrizes de confusão são instrumentos analíticos fundamentais, pois oferecem percepções detalhadas sobre o desempenho dos modelos em termos de verdadeiros positivos, falsos positivos, verdadeiros negativos e falsos negativos.

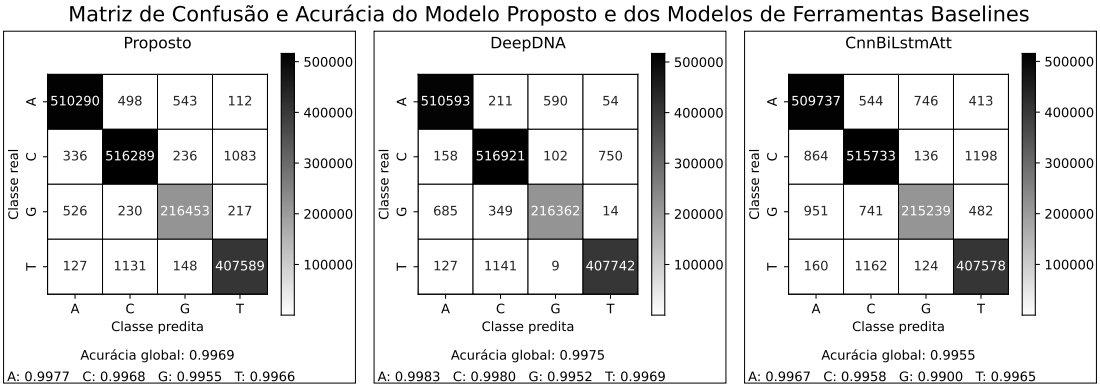

Figura 4.1: Comparação das Matrizes de Confusão e Acurácias Globais e por Classe entre o Modelo de rede *CnnDoubleBiLstm* e os Modelos de rede das ferramentas *Baseline*. Cada sub-figura ilustra a matriz de confusão correspondente a cada modelo, destacando tanto os acertos quanto os erros de classificação. As acurácias globais são destacadas abaixo de cada matriz, juntamente com as acurácias específicas para cada classe representada pelas bases nitrogenadas: Adenina (A), Citosina (C), Guanina (G) e Timina (T) – o autor 2024

As matrizes de confusão apresentadas (cf. Figura 4.1) refletem o desempenho de três diferentes modelos de classificação aplicados a uma tarefa de predição do próximo símbolo de sequências de DNA, onde as classes são representadas pelas bases nitrogenadas adenina (A), citosina (C), guanina (G) e timina (T). O Modelo *CnnDoubleBiLstm*, o *DeepDNA* e o *CnnBiLstmAtt* exibem grau de acurácia global, com 99, 69%, 99, 75% e 99, 55%, respectivamente, mostrando a eficácia na predição do próximo símbolo das sequências genéticas.

O Modelo *CnnDoubleBiLstm* apresenta uma acurácia levemente inferior ao modelo *CnnLstm* (utilizado no compressor *DeepDNA*), contudo, sua distribuição de erros é equilibrada, indicando uma consistência no desempenho entre as classes. O *CnnLstm* tem a maior acurácia global e a maior acurácia por classe para a adenina (A), citosina (C) e timina (T), sugerindo uma vantagem na identificação destas bases específicas.

Por outro lado, o modelo *CnnBiLstmAtt*, apesar de ter a acurácia global mais baixa, apresenta uma acurácia por classe comparável aos outros modelos para as bases citosina (C) e guanina (G), mas uma discrepância maior para as bases adenina (A) e timina (T). Isso pode indicar uma especialização do modelo nas classes C e G em detrimento de A e T, ou uma susceptibilidade maior a desequilíbrios de classe. Para um uso em que a identificação equitativa de todas as bases é importante, o Modelo *CnnDoubleBiLstm* é preferível.

Tabela 4.1: Comparação do Percentual de Acurácia dos Modelos de Predição do Próximo Símbolo – o autor 2024

Modelo	**Acurácia Global**	**A**	**C**	**G**	**T**
CnnLstm	**99,75%**	**99,83%**	**99,80%**	99,52%	**99,69%**
CnnDoubleBiLstm	99,69%	99,77%	99,68%	**99,55%**	99,66%
CnnBiLstmAtt	99,55%	99,67%	99,58%	99,00%	99,65%

O equilíbrio entre sensibilidade e especificidade em todas as classes é um fator importante na avaliação dos modelos em contextos práticos, particularmente em aplicações de predição do próximo símbolo, onde o custo de falsos positivos ou negativos pode ser significativo. Estudos adicionais sobre a distribuição das classes nos conjuntos de dados de treinamento e validação podem fornecer percepções sobre como os modelos podem ser ajustados ou combinados para melhorar o desempenho geral na classificação de sequências de DNA.

4.0.6 Análise dos resultados dos Testes de Compressão e Descompressão

Na presente análise, foi abordada a comparação entre modelos de predição para a compressão de sequências mitocondriais, enfocando especificamente em estabelecer uma comparação justa entre diferentes técnicas de codificação aplicadas a modelos previamente propostos por outros autores e um modelo de rede utilizado neste estudo. Os modelos de base, *CnnLstm* e *CnnBiLstm* com mecanismo de *Atenção com Contexto*, que foram introduzidos respectivamente nos trabalhos denominados *DeepDNA* (WANG et al., 2018) e outro estudo subsequente (BALDI et al., 1999), atuam como *benchmarks* para esta análise. Tais estudos anteriores limitaram-se a apresentar os resultados de compressão baseando-se no cálculo da entropia de *Shannon* (SHANNON, 1948), sugerindo que as probabilidades de predição das bases nitrogenadas (Adenina, Citosina, Guanina e Timina) poderiam ser diretamente submetidas a um codificador aritmético, sem, contudo, implementar efetivamente o processo de codificação em seus códigos disponibilizados.

Em resposta a esta lacuna, este estudo avançou na implementação prática de três métodos distintos de transformação de dados: Codificação Aritmética (ARE) (RISSANEN, 1976) (cf. Algoritmo 5), Codificação RLE baseado em binário (ELIAS, 1975) (cf. Algoritmo 2) e um último que faz parte do desenvolvimento desse trabalho, denominado *Bitstring as Codebook (BAC)* (cf. Algoritmo 9) que anota, no *Codebook*, os deltas das posições onde o modelo preditivo erra a previsão do próximo símbolo.

Desempenho Entre a Ferramenta Desenvolvida e as Ferramentas *Baseline*

A avaliação preliminar de desempenho focou nos resultados alcançados por cada ferramenta, *baseline* e desenvolvida, que utilizaram as técnicas *Arithmetic Encoding* (ARE) e *Bitstring as Codebook* (BAC), respectivamente, para transformação dos dados. Este procedimento teve como objetivo medir a capacidade média de cada ferramenta em minimizar o espaço de armazenamento necessário, focando especificamente na métrica Economia Média de Espaço, fundamental para avaliar o desempenho das técnicas de compressão utilizadas. Assim, três experimentos foram realizados e avaliados, sendo dois nomeados como *CnnLst-*

mARE e *CnnBiLstmARE* para as ferramentas *baseline* e outro nomeado como *CnnDoubleBiLstmBAC* para a ferramenta desenvolvida.

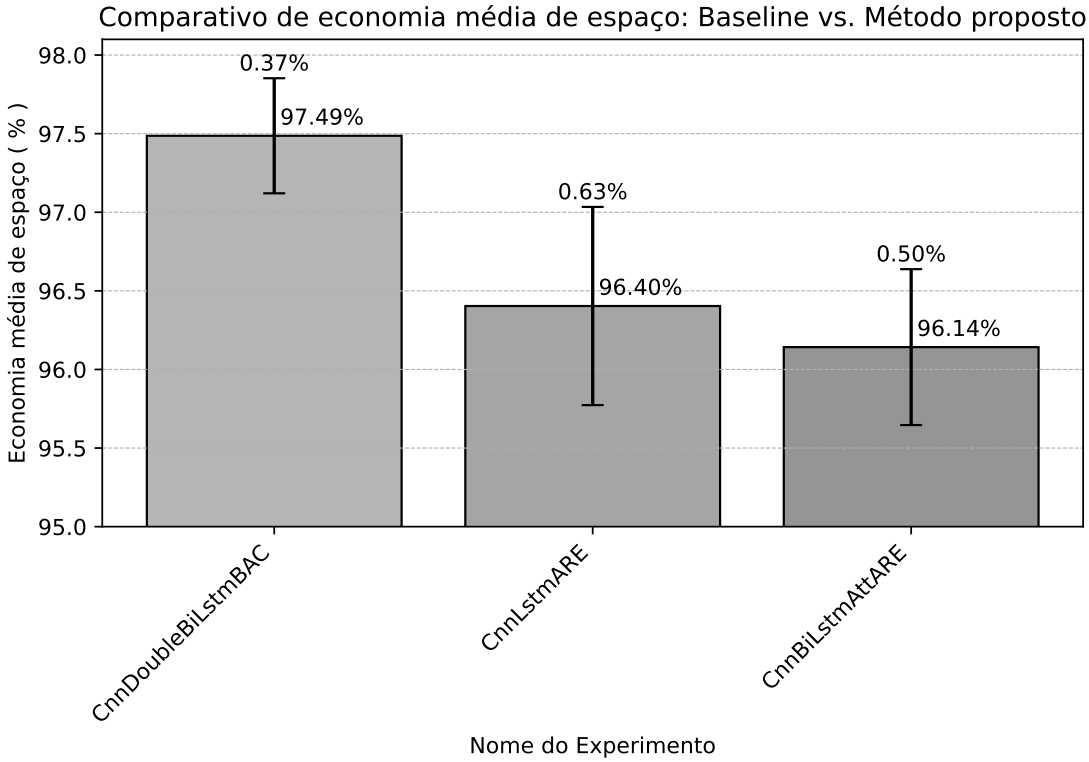

Figura 4.2: Comparativo de economia média ($\bar{x}$) de espaço entre os experimentos representando as ferramentas *baseline* e a ferramenta proposta (experimento *CnnDoubleBiLstm*), destacando a superioridade deste último em termos de porcentagem média de espaço economizado (97,49%) e menor variabilidade, como indicado pelo menor desvio padrão (±0,37%) – o autor 2024

A análise comparativa da economia média de espaço (cf. Figura 4.2) revela diferenças significativas entre os experimentos que representam as ferramentas *baseline* e o experimento da ferramenta proposta *CnnDoubleBiLstmBAC*. O experimento *CnnDoubleBiLstmBAC* apresentou uma economia de espaço superior, com uma média de 97,49%, superando os experimentos *CnnLstmARE* e *CnnBiLstmARE*, que alcançaram 96,40% e 96,14% respectivamente. Além disso, o valor do desvio padrão σ para o experimento *CnnDoubleBiLstmBAC* foi o menor, com ±0,37%, indicando uma consistência maior na economia média de espaço em comparação com os desvios padrões de ±0,63% e ±0,50% para os experimentos das ferramentas *baseline*. Estes resultados evidenciam a superioridade

do experimento *CnnDoubleBiLstmBAC* em termos de eficiência e consistência na compressão de dados.

Apesar do desempenho notável do experimento *CnnDoubleBiLstmBAC*, conforme destacado na Figura 4.2, o desempenho obtido passou pelo desenvolvimento e reuso de técnicas na condução dos experimentos. Tal desempenho sublinha a relação da combinação dos modelos de predição do próximo símbolo e técnicas de transformação de dados. Esta constatação conduziu essa pesquisa a uma investigação mais aprofundada sobre como as variações nas abordagens de transformação de dados e nas arquiteturas de predição do símbolo seguinte influenciam a eficiência de armazenamento e a consistência dos resultados obtidos.

Desempenho Entre Experimentos de Compressão

A análise subsequente, apresentada na Tabela 4.2, propõe uma avaliação sistemática das influências de diferentes combinações de técnicas de transformação de dados e modelos de predição por meio de nove experimentos distintos. Esta abordagem permite compreender as potenciais relações entre técnicas de predição e transformação e realizar uma avaliação abrangente e equitativa dos modelos em termos de desempenho de compressão dos dados das sequências mitocondriais. Ao aplicar consistentemente cada método de transformação de dados a todos os modelos de predição considerados, esta abordagem vai além da simples aplicação teórica de técnicas de codificação, revelando percepções práticas sobre o desempenho de compressão de diferentes pares (modelo preditivo e técnicas de transformação), oferecendo uma comparação direta da eficiência em economizar espaço de armazenamento de cada combinação.

Tabela 4.2: Resumo dos Experimentos de Compressão das sequências mitocondriais. Os sufixos nos nomes dos experimentos significam respectivamente: *Run-Length Encoding (RLE) binary-based*, *Arithmetic Encoding (ARE)* e *Bitstring as Codebook (BAC)* – o autor 2024

Nome do Experimento	**Modelo de Predição**	**Codificação**
CnnLstmARE	Cnn+Lstm	*Arithmetic Encoding*
CnnBiLstmAttARE	Cnn+BiLstm+Attention	*Arithmetic Encoding*
CnnDoubleBiLstmARE	Cnn+Double(BiLstm)	*Arithmetic Encoding*
CnnLstmRLE	Cnn+Lstm	*Run-Length Encoding*
CnnBiLstmAttRLE	Cnn+BiLstm+Attention	*Run-Length Encoding*
CnnDoubleBiLstmRLE	Cnn+Double(BiLstm)	*Run-Length Encoding*
CnnLstmBAC	Cnn+Lstm	*Bitstring as Codebook*
CnnBiLstmAttBAC	Cnn+BiLstm+Attention	*Bitstring as Codebook*
CnnDoubleBiLstmBAC	Cnn+Double(BiLstm)	*Bitstring as Codebook*

A Tabela 4.2 resume as abordagens de compressão empregadas nos experimentos, detalhando os métodos de codificação utilizados após a etapa de predição por cada modelo. Os modelos servem como base para a geração da *bitstring* de predições que é, posteriormente, codificada por meio de diferentes técnicas de transformação dos dados. Cada técnica oferece um método único de reduzir o tamanho dos dados, seja pela simplificação das sequências de caracteres por codificação *Run-Length* (RLE), pela eficiência na representação de probabilidades pela Codificação Aritmética (ARE), ou pelo mapeamento das posições deltas dos erros de predição em um *codebook* (BAC).

A Tabela 4.3 mostra um resumo de três métricas de compressão relativas a eficiência de armazenamento para os experimentos de compressão. Os experimentos foram avaliadas com base em três critérios: a quantidade de *bits* necessários para armazenar cada base (acompanhados de seu desvio padrão σ), a Taxa de Compressão e seu desvio padrão σ, e a economia média de espaço. A disposição dos experimentos em categorias distintas de técnicas de transformação dos dados permite uma comparação direta do desempenho entre as

variantes (ARE), (RLE) e (BAC) em contraste com três abordagens diferentes de predição do próximo símbolo: *CnnLstm*, *CnnBiLstmAtt* e *CnnDoubleBiLstm*.

Tabela 4.3: Resumo dos resultados das métricas de compressão avaliando as dimensões quantidade média ($\bar{x}$) de *Bits* por base, taxa média ($\bar{x}$) de compressão e porcentagem média ($\bar{x}$) de Economia Média de Espaço para os diferentes experimentos de compressão de sequência mitocondrial – o autor 2024

Experimentos	**Bits Por Base ($\bar{x}$ e σ)**	**Taxa De Compressão ($\bar{x}$ e σ)**	**Economia De Espaço ($\bar{x}$ e σ)**
CnnLstmARE	**0,29 (±0,05)**	**28:1 (±3,43)**	**96,40% (±0,63%)**
CnnBiLstmAttARE	0,31 (±0,04)	26:1 (±2,49)	96,14% (±0,50%)
CnnDoubleBiLstmARE	0,30 (±0,05)	28:1 (±3,19)	96,37% (±0,63%)
CnnLstmRLE	0,26 (±0,03)	32:1 (±2,78)	96,85% (±0,42%)
CnnBiLstmAttRLE	0,29 (±0,04)	29:1 (±2,81)	96,50% (±0,44%)
CnnDoubleBiLstmRLE	**0,26 (±0,03)**	**32:1 (±3,04)**	**96,86% (±0,43%)**
CnnLstmBAC	**0,20 (±0,03)**	**41:1 (±3,92)**	**97,51% (±0,37%)**
CnnBiLstmAttBAC	0,22 (±0,03)	37:1 (±3,40)	97,25% (±0,35%)
CnnDoubleBiLstmBAC	0,20 (±0,03)	40:1 (±3,84)	97,49% (±0,37%)

A análise dos dados apresentados (cf. Tabela 4.3) revela percepções importantes sobre o desempenho de compressão dos experimentos considerados. Em termos de eficiência de compressão, medida pela *Taxa Média de Compressão*, observa-se que o experimento *CnnLstmBAC* se destaca com a maior taxa média (41:1), indicando seu desempenho em reduzir o tamanho do arquivo em comparação com os demais experimentos. Esta taxa média de compressão é acompanhada de uma economia média de espaço significativa, atingindo até 97,51%, o que ressalta sua eficácia em comprimir os dados.

Por outro lado, a análise dos *Bits Por Base*, que oferece uma medida da quantidade de informação comprimida por unidade de dado, mostra que *CnnLstmBAC*, *CnnDoubleBiLstmBAC*, e *CnnBiLstmAttBAC* possuem os valores médios mais baixos, sugerindo que estes experimentos são particularmente eficientes na compressão dos dados. No entanto, a variação no desempenho (indicada pelos desvios padrão σ) sugere que a eficácia da compressão pode variar dependendo do tipo específico de dado sendo processado e a capacidade do modelo acertar a predição do próximo símbolo.

Tabela 4.4: Comparativo detalhado dos tempos médios de compressão e descompressão para variantes de experimentos baseados em *CnnLstm*, *CnnBiLstmAtt* e *CnnDoubleBiLstm*, evidenciando a eficiência e desempenho em diferentes contextos para ARE, RLE e BAC. Os tempos médios de compressão e descompressão dos diferentes experimentos estão representados em segundos e a conversão correspondente em horas – o autor 2024

Experimentos	**Tempo de compressão ($\bar{x}$)**	**Tempo de descompressão ($\bar{x}$)**
CnnLstmARE	**2.024,43 (00:33:44)**	**2.041,23 (00:34:01)**
CnnBiLstmAttARE	2.125,01 (00:35:25)	2.163,55 (00:36:03)
CnnDoubleBiLstmARE	5.461,84 (01:31:01)	5.478,38 (01:31:18)
CnnLstmRLE	**2.002,12 (00:33:22)**	**2.057,52 (00:34:17)**
CnnBiLstmAttRLE	2.139,38 (00:35:39)	2.263,82 (00:37:43)
CnnDoubleBiLstmRLE	5.443,21 (01:30:43)	5.583,36 (01:33:03)
CnnLstmBAC	**2.003,21 (00:33:23)**	**2.019,65 (00:33:39)**
CnnBiLstmAttBAC	2.123,76 (00:35:23)	2.367,48 (00:39:27)
CnnDoubleBiLstmBAC	5.443,70 (01:30:43)	5.443,88 (01:30:43)

A Tabela 4.4 apresenta uma análise dos tempos médios de compressão e descompressão, aplicados a uma série de experimentos distintos. A análise revela diferenças significativas (mais de 50 minutos) nos tempos de compressão e descompressão entre os grupos de experimentos, proporcionando percepções sobre a eficácia e a eficiência das variantes arquitetônicas em processar e recuperar informações. Notavelmente, os experimentos baseados na arquitetura *CnnDoubleBiLstm* mostram os maiores tempos médios de compressão quanto de descompressão, refletindo a complexidade desta configuração e sugerindo uma menor eficiência operacional em comparação com as demais arquiteturas testadas.

Em contrapartida, os experimentos *CnnLstm* e *CnnBiLstmAtt* apresentam tempos de compressão e descompressão mais moderados, com destaque para *CnnLstmBAC* que registra tempos ligeiramente inferiores aos demais dentro do mesmo grupo arquitetônico, apontando para uma economia de tempo potencial nesta configuração. A inclusão do mecanismo de atenção bidirecional (*BiLstmAtt*) nos experimentos *CnnBiLstmAtt* não resultou em uma melhoria substancial dos tempos de compressão e descompressão, indicando que as melhorias

podem ser marginais ou contextuais, dependendo da especificidade das tarefas executadas.

A análise dos valores dos resultados coletados dos testes dos experimentos revela informações sobre o desempenho computacional relacionado às operações de compressão e descompressão. A Tabela 4.5 apresenta um resumo detalhado desses desempenhos, englobando o uso médio de CPU, expresso em percentual (%), e o uso médio de memória, apresentado tanto em *bytes* quanto em *megabytes* (MB), para facilitar uma interpretação precisa dos resultados.

Tabela 4.5: Análise detalhada do percentual médio de Uso de CPU e memória em diferentes experimentos. Os valores de Uso de Memória são apresentados em *bytes* e entre parênteses a sua conversão para *megabytes* (MB), considerando a conversão 1 MB = 1.048.576 $bytes$ – o autor 2024

Experimentos	**Uso de CPU (%)**		**Uso de Memória (*bytes* e *MB*)**	
	Comp.	**Desc.**	**Comp.**	**Desc.**
CnnLstmARE	**31,89**	**28,20**	24.322.890	20.794.820
			(23,19 MB)	(19,82 MB)
CnnBiLstmAttARE	35,20	31,69	14.852.170	**9.925.941**
			(14,18 MB)	**(9,46 MB)**
CnnDoubleBiLstmARE	45,31	37,40	**5.290.229**	10.636.530
			(5,04 MB)	(10,14 MB)
CnnLstmRLE	**23,92**	**26,02**	21.648.690	19.523.030
			(20,63 MB)	(18,61 MB)
CnnBiLstmAttRLE	27,50	30,24	19.884.700	15.830.640
			(18,96 MB)	(15,11 MB)
CnnDoubleBiLstmRLE	24,64	29,74	**16.276.720**	**15.412.440**
			(15,52 MB)	**(14,70 MB)**
CnnLstmBAC	27,17	**24,40**	34.360.500	22.769.900
			(32,77 MB)	(21,70 MB)
CnnBiLstmAttBAC	**22,97**	33,08	42.945.460	**16.271.970**
			(40,93 MB)	**(15,51 MB)**
CnnDoubleBiLstmBAC	24,17	24,86	**20.242.470**	18.848.970
			(19,31 MB)	(17,97 MB)

É importante destacar que os valores de Uso de Memória estão expressos tanto em *bytes* quanto em *megabytes* (MB) na Tabela 4.5, proporcionando uma compreensão mais precisa do consumo de recursos de *hardware*. Esta dualidade na apresentação dos dados assegura que a análise seja acessível tanto para leitores que preferem uma visão mais técnica quanto para aqueles que buscam uma interpretação mais imediata dos resultados.

A análise dos resultados (cf. Tabela 4.5) indica que o experimento *CnnDoubleBiLstmARE* registrou o maior uso médio de CPU durante a compressão, alcançando 45,31%, enquanto o *CnnBiLstmAttBAC* mostrou o menor uso médio de CPU, com 22,97%, para o mesmo processo. Por outro lado, a descompressão apresentou uma variação menos acentuada no Uso de CPU entre os experimentos, sugerindo uma consistência maior nesta fase do processo.

Uma variação significativa no uso de memória foi observada nos processos de compressão e descompressão. O experimento *CnnBiLstmAttBAC* mostrou o maior uso médio de memória durante a compressão, aproximadamente 40,93 MB, enquanto *CnnDoubleBiLstmARE* mostrou o menor percentual de uso, aproximadamente 5,04 MB. Essas variações refletem a eficiência no gerenciamento de memória dos algoritmos preditores utilizados em cada experimento, uma vez que *CnnDoubleBiLstm*, com os codificadores ARE, RLE e BAC, obteve os menores valores médios em todas as compressões.

Além das análises das métricas já apresentadas, é possível observar várias características importantes sobre a distribuição dos dados de economia de espaço entre os diferentes experimentos de compressão fazendo uso de um gráfico de caixas. Este gráfico proporciona uma visão comparativa detalhada, destacando não apenas as diferenças na eficiência de cada experimento, mas também revelando variabilidades, consistências e eventuais anomalias nos resultados (cf. Figura 4.3).

A linha que corta cada caixa indica a mediana da economia de espaço para cada experimento. A posição vertical dessas medianas varia entre os experimentos, sugerindo diferenças na eficiência de compressão entre eles. Experimentos com medianas mais altas indicam uma maior economia de espaço média como é o caso para os experimentos que usam a codificação BAC.

Além disso, a altura das caixas (que representa o intervalo interquartil, ou IQR) varia entre os experimentos, indicando diferenças na dispersão ou variabilidade da economia de espaço. Caixas mais altas sugerem uma maior variação na economia de espaço alcançada, enquanto caixas de menor altura indicam

resultados mais consistentes entre as execuções do mesmo experimento, como ocorreu com os experimentos *CnnLstmRLE* e *CnnDoubleBiLstmRLE*.

Em relação aos pontos de dados de compressão discrepantes, os pontos abaixo de cada caixa no gráfico representam *outliers*, ou seja, pontos de dados que se desviam significativamente dos demais dentro do mesmo experimento. A presença de *outliers* pode indicar variações extremas no desempenho da compressão, o que pode ser devido a erros de predição específicos das sequências mitocondriais processadas. Além disso, é possível que não tenham havido amostras suficientes das sequências que apresentaram mais erros de predição. No entanto, estas observações requerem uma investigação mais aprofundada para compreender completamente as causas subjacentes. Alguns experimentos apresentam um número significativo de *outliers*, o que sugere que, sob certas condições, o desempenho desses experimentos pode variar consideravelmente. A proximidade e padronização de *outliers* nos experimentos, que utilizam BAC na etapa de codificação, indicam maior consistência neste método de codificação. Isto sugere que as variações extremas são menos pronunciadas mesmo nos casos em que a codificação BAC não segue a tendência de economia de espaço (resultando em *outliers*). Em outras palavras, a codificação BAC tem desempenho relativamente estável em diferentes conjuntos de dados.

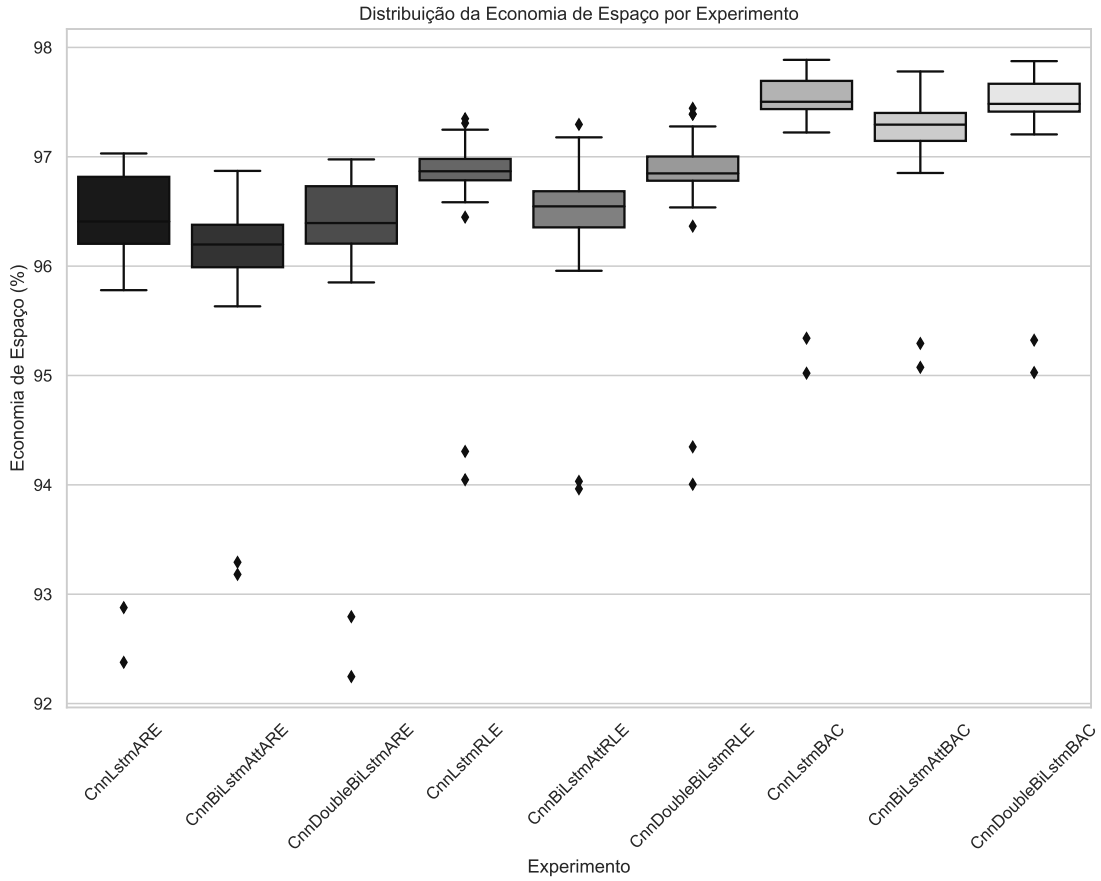

Figura 4.3: Ilustração da distribuição da Economia de Espaço para cada experimento. Este gráfico permite observar não apenas as medianas, mas também a variabilidade e possíveis *outliers* na Economia de Espaço alcançada em cada experimento na tarefa de compressão – o autor 2024

A análise dos *outliers* (cf. Figura 4.3) também pode revelar os casos extremos de Economia de Espaço, tanto positivamente (maior Economia de Espaço) quanto negativamente (menor Economia de Espaço). Estes casos extremos é de particular interesse para análise adicional, na medida que eles fornecem percepções sobre as limitações ou vantagens dos experimentos de compressão em casos específicos. Essas observações servem como uma avaliação preliminar dos desempenhos relativos dos diferentes experimentos de compressão. No entanto, para conclusões definitivas sobre a superioridade de um método sobre os outros, análises estatísticas adicionais, como o teste de *Friedman* (FRIEDMAN, 1937) seguido pelo teste *post-hoc* de *Nemenyi* (NEMENYI, 1962), são necessárias para confirmar diferenças significativas.

Os testes não paramétricos *Friedman* e *Nemenyi* foram utilizados para validar os resultados obtidos. O teste de *Friedman* constitui um método experimental de blocos aleatórios análogo à ANOVA para medidas repetidas (FRIEDMAN, 1937). Este teste procede à classificação dos algoritmos em cada conjunto de dados individualmente para determinar se as variações observadas entre as populações são estatisticamente significativas. Em vez de usar valores brutos, o teste de *Friedman* é baseado nos ranques (classificação ordenada de dados) para calcular estatísticas relevantes. Dado que o teste de *Friedman* se abstém de fazer suposições sobre a distribuição dos dados, a sua capacidade de detectar a normalidade nas populações não é tão prática como poderia ser, comprometendo, até certo ponto, o seu poder estatístico para determinar diferenças genuínas entre populações.

Neste contexto, o teste *Nemenyi* é frequentemente aplicado como uma extensão do teste *Friedman*, destinado a realizar comparações *post-hoc* entre todos os pares de grupos. O teste *Nemenyi* é baseado em uma comparação de médias de classificação, onde a **hipótese nula** assume que todas as populações têm médias de classificação iguais (NEMENYI, 1962). Portanto, este tipo de teste identifica quais pares específicos de grupos ou experimentos apresentam diferenças estatisticamente significativas em seu desempenho após o teste de *Friedman* indicar a existência de diferenças globais entre tais grupos.

O teste de *Friedman* aplicado à variável "Economia de Espaço" gerou uma estatística de teste de aproximadamente $734,77$ e um valor de p extremamente baixo ($p < 2,33 \times 10^{-153}$), conforme apresentado na Tabela 4.6. Esses resultados evidenciam uma diferença estatisticamente significativa na economia de espaço entre os diversos experimentos realizados. Tais experimentos incluíram a previsão do próximo símbolo e a transformação dos dados, aplicados a 100 sequências mitocondriais, afetando diretamente os resultados de compressão observados.

Tabela 4.6: Resultados do Teste de *Friedman* para a Análise da Economia de Espaço – o autor 2024

Métrica	**Valor**
Estatística	$734,77$
Valor-p	$2,33 \times 10^{-153}$

Dada a rejeição da **hipótese nula** pelo teste de *Friedman*, sugere-se que pelo menos um dos experimentos de compressão difere significativamente dos outros em termos de Economia de Espaço. Para identificar especificamente quais experimentos diferem entre si, é apropriado prosseguir com um teste *post-hoc*, como o teste de *Nemenyi* (cf. Figura 4.4), que compara todos os pares de grupos para localizar onde essas diferenças significativas ocorrem. Os experimentos são classificados pelo seu ranque médio de Economia de Espaço, indicando a eficiência relativa em reduzir o tamanho dos dados. A linha que conecta os experimentos representa a distância crítica (*Critical Distance - CD*), com grupos de experimentos não significativamente diferentes (p > 0,05) indicados por não sobreposição das linhas. Esta análise destaca as diferenças estatísticas no desempenho de compressão entre os experimentos avaliados.

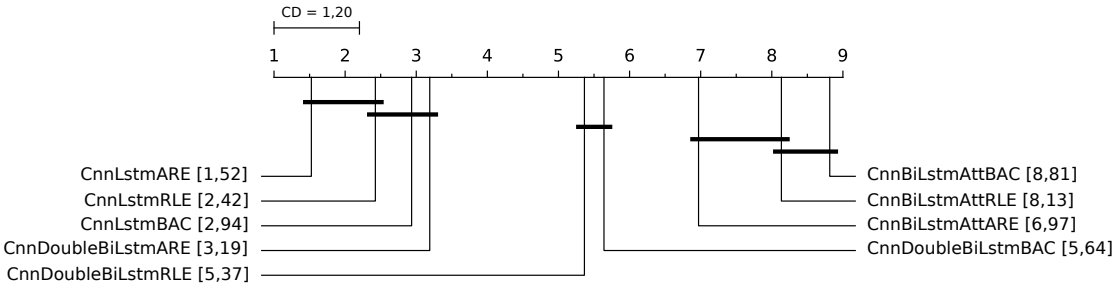

Figura 4.4: Comparação dos ranques médios de Economia de Espaço entre diferentes experimentos de compressão de sequências mitocondriais, utilizando o teste de *Nemenyi* para cálculo de distância crítica ($CD = 1,20$) – o autor 2024

Com base na análise estatística detalhada na Figura 4.4 dos diferentes experimentos de compressão aplicados a sequências mitocondriais, utilizando a métrica de Economia de Espaço, os resultados obtidos por meio do teste de *Friedman* e do subsequente teste *post-hoc* de *Nemenyi* oferecem percepções significativas sobre as relações de desempenho entre os experimentos avaliados. A distância crítica (CD), calculada em $1,20$, serve como um parâmetro fundamental para interpretar as diferenças estatísticas entre os ranques médios dos experimentos.

Os ranques médios, indicativos do desempenho relativo de cada método, no contexto de Economia de Espaço, são apresentados como segue em ordem crescente: *CnnLstmARE* (1,52), *CnnLstmRLE* (2,42), *CnnLstmBAC* (2,94), *CnnDoubleBiLstmARE* (3,19), *CnnDoubleBiLstmRLE* (5,37), *CnnDoubleBiLstmBAC* (5,64), *CnnBiLstmAttARE* (6,97), *CnnBiLstmAttRLE* (8,13) e *CnnBiLstmAttBAC* (8,81).

Esses valores refletem a posição média de cada método em termos de sua capacidade de economizar espaço, sem necessariamente indicar uma hierarquia de eficiência absoluta.

A análise dos ranques médios revela padrões de agrupamento que são influenciados pelas técnicas de predição. Observa-se que métodos que utilizam o mesmo modelo de preditor geralmente apresentam desempenhos similares, o que indica que a seleção do preditor é um fator importante no desempenho da compressão. Especificamente, os experimentos que empregam a técnica *CnnLstm* são agrupados com ranques abaixo de 3,5, aqueles que utilizam *CnnDoubleBiLstm* são agrupados com ranques entre 5,0 e 6,0, e os que fazem uso de *CnnBiLstmAtt* estão agrupados com ranques superiores a 6,5 (cf. Figura 4.4).

A proximidade dos ranques médios dentro destes grupos sugere que, embora possam existir diferenças, elas não são estatisticamente significativas dentro da margem estabelecida pela distância crítica. Por exemplo, a proximidade entre os ranques médios de *CnnLstmARE*, *CnnLstmRLE* e *CnnLstmBAC* indica uma semelhança no desempenho da economia de espaço que é consistente sob diferentes técnicas de codificação, destacando a influência predominante do componente de predição do próximo símbolo.

Por outro lado, a variação maior entre os grupos sugere que diferentes combinações de técnicas de predição e codificação impactam significativamente o desempenho global dos métodos de compressão. O teste de Nemenyi serve então para ilustrar não apenas as semelhanças significativas entre métodos com ranques próximos, mas também para enfatizar as distinções claras entre os grupos, que excedem a margem da distância crítica. Portanto, a análise não busca estabelecer uma ordenação linear de eficácia entre os métodos, mas sim revelar a complexidade e as nuances nas relações de desempenho entre as diversas abordagens experimentais, conforme evidenciado na Figura 4.4.

A avaliação da eficiência em compressão de dados é importante para entender o impacto de diferentes fatores na economia de espaço alcançada. Neste contexto, a correlação de *Pearson* emerge como uma métrica estatística fundamental, revelando percepções quantitativas sobre como variáveis relacionadas ao processo de compressão influenciam a economia de espaço. Esta análise não só destaca a importância de características da sequência genômica, como o número de bases no arquivo FASTA e o tamanho original do arquivo, mas também examina o desempenho do processo de compressão, incluindo o uso de CPU e Memória, bem como os tempos de compressão e descompressão. Os

resultados detalhados na Tabela 4.7, ilustram as correlações entre a economia de espaço e as variáveis consideradas, oferecendo uma base para otimizações e aprimoramentos nos algoritmos de compressão de dados.

Tabela 4.7: Correlação de *Pearson* entre a **Economia de Espaço** e Outras Dimensões – o autor 2024

Dimensão	**Correlação**	**Tipo**
Taxa de Compressão	0,94	Muito forte
Número de Bases	0,61	Moderada
Tamanho Original (*bytes*)	0,60	Moderada
Uso de Memória (comp.)	0,37	Fraca
Uso de Memória (descomp.)	0,25	Muito fraca
Tempo de Predição (comp.)	0,08	Quase inexistente
Tempo de Predição (descomp.)	0,08	Quase inexistente
Tempo de Codificação (comp.)	0,03	Quase inexistente
Tempo de Codificação (descomp)	-0,30	Inversamente fraca
Tempo Total (comp.)	0,08	Quase inexistente
Tempo Total (descomp.)	0,08	Quase inexistente
Uso de CPU (comp.)	-0,20	Inversamente fraca
Uso de CPU (descomp.)	-0,36	Inversamente fraca
Tamanho Comprimido	-1,00	Inversamente perfeita
Bits por Base	-1,00	Inversamente perfeita

A análise de correlação de *Pearson* (cf. Tabela 4.7) revelou as seguintes correlações com a Economia de Espaço:

- **Correlações Positivas Fortes:**
  - *Taxa de Compressão*: 0,94, indicando uma correlação muito forte. Isso sugere que quanto maior a Taxa de Compressão, maior a Economia de Espaço, o que é esperado, pois uma Taxa de Compressão elevada implica em uma redução significativa do tamanho do arquivo.
  - *Número de Bases* e *Tamanho Original (bytes)*: Ambas apresentam correlações moderadas (aproximadamente 0,61 e 0,60, respectivamente)

com a Economia de Espaço, sugerindo que arquivos maiores ou com mais bases tendem a proporcionar uma maior Economia de Espaço absoluto após a compressão.

  - *Uso de Memória (comp)* e *Uso de Memória (descomp)*: Correlações leves (0,37 e 0,25, respectivamente) indicam que o Uso de Memória durante a compressão e descompressão tem uma relação positiva com a Economia de Espaço, embora menor do que a relação com outros fatores.

- **Correlações Negativas Fortes:**

  - *Tamanho Comprimido* e *Bits Por Base*: Ambos apresentam uma correlação inversamente perfeita de -1,00 com a Economia de Espaço. Esses valores negativos indicam que um aumento nessas métricas está associado a uma diminuição na Economia de Espaço. Para *Tamanho Comprimido*, isso é diretamente inversa, já que uma Economia de Espaço maior é alcançada com tamanhos de arquivo comprimido menores. Para *Bits Por Base*, menos *bits* por base significam uma compressão mais eficiente, levando a uma maior Economia de Espaço.

- **Correlações Negativas Leves a Moderadas:**

  - *Uso de CPU (comp.)*, *Tempo de Codificação (comp.)*, e *Uso de CPU (descomp.)*: Estas colunas mostram correlações negativas leves a moderadas com a Economia de Espaço, indicando que um aumento no Uso de CPU e no Tempo de Compressão pode estar associado a uma menor Economia de Espaço, possivelmente devido à complexidade da compressão ou eficiência do algoritmo.

Esses resultados enfatizam a importância da *Taxa de Compressão* como principal influenciador da Economia de Espaço, ao mesmo tempo em que considera o impacto do tamanho original do arquivo e a eficiência de compressão (*Bits Por Base*). As métricas de desempenho, como Uso de CPU e memória durante a compressão e descompressão, também são relevantes, embora tenham uma influência menor na Economia de Espaço final.

A precisão na predição do próximo símbolo das sequências de DNA é fundamental para a eficácia do modelo. Neste estudo, foi analisado o desempenho de três modelos de predição: *CnnLstm*, *CnnDoubleBiLstm* e *CnnBiLstmAtt* com mecanismo de atenção, focando em sua capacidade de prever corretamente as bases nitrogenadas A, C, G e T em sequências mitocondriais. Para isso, realizou-se a contagem global de erros anotados nas 100 *bitstrings* de predição geradas por cada modelo. A Tabela 4.8 apresenta uma comparação desses modelos, destacando as taxas de erro globais e a distribuição dos erros por cada base nitrogenada. Este comparativo permite identificar as especificidades de cada modelo.

Tabela 4.8: Taxas de erro e distribuição dos erros por base nitrogenada referentes aos modelos de predição – o autor 2024

Modelo	**Taxa de Erro (%)**	**Distribuição de erros por base (%)**			
		A	**C**	**G**	**T**
CnnLstm	0,67	24,50	32,86	18,06	24,59
CnnDoubleBiLstm	0,70	23,70	35,34	16,24	24,72
CnnBiLstmAtt	0,82	23,42	32,41	24,16	20,00

O modelo *CnnLstm* mostrou a menor taxa de erro global, com 0,67%, indicando um desempenho superior na predição das sequências de DNA mitocondriais em comparação aos demais modelos. Nota-se, entretanto, que o *CnnDoubleBiLstm* apresentou a maior dificuldade em prever corretamente a base nitrogenada C, com uma taxa de erro para esta base de 35,34%, a mais alta entre os modelos avaliados. Por outro lado, o modelo *CnnBiLstmAtt*, além de apresentar a maior taxa de erro global de 0,82%, exibiu um comportamento diferenciado na predição da base G, com uma taxa de erro significativamente maior (24,16%) em relação aos outros modelos. Este padrão sugere que o mecanismo de atenção incorporado neste modelo pode influenciar na maneira como as sequências mitocondriais são analisadas, resultando em uma distribuição de erros distinta. A análise das taxas de erro e de sua distribuição por base nitrogenada evidencia

a complexidade da tarefa de modelagem preditiva e reforça a importância de se considerar as características específicas de cada modelo na seleção da abordagem mais adequada para aplicações de compressão e descompressão de sequências genômicas mitocondriais. Assim, a dificuldade dos modelos em prever corretamente uma determinada base nitrogenada pode ser atribuída a uma combinação de fatores relacionados à arquitetura do modelo, à distribuição das bases nas sequências mitocondriais e às características contextuais das regiões onde essas bases estão presentes. Uma investigação mais detalhada envolvendo a análise de padrões de erros específicos e a exploração de diferentes configurações do modelo poderia fornecer *insights* adicionais sobre esse comportamento.

Conclusão do Capítulo

Este capítulo apresentou uma comparação detalhada envolvendo vários experimentos de compressão de genoma baseados em um protocolo de avaliação de desempenho bem-definido. Por meio da implementação de três modelos de predição distintos, associados a técnicas de codificação específicas, procedeu-se à realização de uma série de testes destinados a avaliar o desempenho dessas combinações na compressão de dados genômicos. As ferramentas de referência, nomeadamente *DeepDNA* (WANG et al., 2018) e uma ferramenta que implementa uma rede biLSTM com mecanismo de atenção, conhecido como *AttentionWithContext* (BALDI et al., 1999), além da ferramenta desenvolvida em questão, foram submetidas a uma série de testes, utilizando-se de um conjunto de dados selecionado, em um ambiente de execução controlado.

A investigação dos resultados revelou que no contexto de compressão de dados, particularmente de sequências mitocondriais, a estratégia em análise que utiliza *CnnDoubleBiLstm* na etapa de predição e *Bitstring as Codebook (BAC)* na etapa de codificação apresentou desempenho superior em relação às ferramentas *baseline*, ambas empregando o método de codificação *Run Length Encoding (RLE)* na etapa de codificação. Contudo, foi a combinação do modelo de rede *Convolutional Neural Network and Long Short-Term Memory (CnnLstm)* com o método de codificação *Bitstring as Codebook (BAC)*, cuja combinação é referida neste estudo como *CnnLstmBAC* (cf. Tabela 4.2), que se destacou como o método de compressão mais eficiente, alcançando um desempenho mais significativo em termos de Economia de Espaço. Isso era esperado uma vez que o modelo *CnnLstm* apresentou a menor taxa de erro de predição, precisamente 0,67% (cf.

Tabela 4.8). Quanto menor o número de erros preditos e anotados na *bitstring* de predição (cf. Figura 3.4), maior é a distância entre os deltas das posições dos erros a serem armazenados no *codebook*, reduzindo assim a quantidade de informações necessárias para serem armazenadas e reutilizadas na descompressão.

Ao concluir este capítulo, enfatiza-se a relevância da abordagem comparativa e do protocolo de avaliação de desempenho para análise de ferramentas de compressão de genoma. As percepções obtidas dos testes de compressão e descompressão contribuem significativamente para este trabalho, oferecendo orientações para novas ferramentas de compressão com diferentes abordagens. No capítulo subsequente são apresentadas as conclusões gerais do estudo, enfatizando os melhores resultados dos experimentos avaliados.

5

Conclusão e Trabalhos Futuros

Deve-se relembrar que a crescente necessidade de métodos eficazes para o armazenamento de dados genômicos foi impulsionada pelo avanço do Sequenciamento de Nova Geração (*Next Generation Sequencing - NGS*) que revolucionou a bioinformática e a pesquisa genômica ao permitir a coleta de dados em uma escala sem precedentes. No entanto, essa evolução também gerou desafios significativos relacionados ao armazenamento, à gestão, à análise e transmissão desses volumosos conjuntos de dados. A expansão exponencial dos dados genômicos resulta em aumento dos custos e complexidades logísticas, ressaltando a importância de desenvolver estratégias de compressão de dados que não apenas minimizem a demanda por espaço de armazenamento mas também promovam uma gestão de dados mais eficiente e economicamente viável.

Este capítulo apresenta os avanços deste trabalho na área de compressão de dados de sequências genômicas por meio do desenvolvimento de um compressor baseado em modelos preditivos e da definição de um protocolo padronizado para o *benchmark* de ferramentas especializadas nesta área. A pesquisa contribuiu em duas direções: o compressor (experimento) *CnnLstmBAC* que apresentou a melhor economia de espaço nos testes de desempenho, e o protocolo de avaliação de desempenho comparativo que aborda uma lacuna importante na literatura científica.

Além da definição de um protocolo padronizado, uma necessidade identificada, desenvolveu-se também um compressor de dados genômicos que utiliza modelos preditivos. Esses modelos possuem o potencial de aprimorar significativamente a eficiência em termos de Economia de Espaço. Eles operam

aprendendo, dado um contexto, as informações presentes na sequência de DNA para prever o próximo símbolo. Esse processo resulta na compressão de dados mais eficiente quando comparado com métodos tradicionais. A razão para essa eficiência é que, caso o modelo preditivo alcance 100% de precisão toda a informação necessária para descomprimir a sequência genômica estaria armazenada no próprio modelo. Dessa forma, o compressor desenvolvido, baseado em modelo preditivo, representa uma solução importante entre as já existentes.

5.1 Protocolo e Compressor Preditivo

De forma mais detalhada, essa pesquisa colocou em prática duas realizações para avançar na compressão de dados genômicos e padronizar a avaliação de desempenho das ferramentas especializadas para compressão de sequências genômicas, a saber:

a) **Protocolo Padronizado para Avaliação de Desempenho**: Foi desenvolvido um protocolo padronizado para avaliar o desempenho de ferramentas de compressão de genomas, considerando critérios para medir a eficiência da compressão e descompressão das sequências de DNA. As métricas presentes no protocolo são: Taxa de Compressão, Bits por Base, Percentual de Economia de Espaço, Tempo de execução, bem como uso de CPU e Memória na compressão e descompressão. Além disso, este protocolo oferece uma diretriz consistente para comparar diferentes ferramentas de compressão em que todos os tamanhos sejam apresentado em *bytes*, facilitando a identificação das mais eficientes.

b) **Compressor de Genoma Baseado em Predição do Próximo Símbolo**: Foi desenvolvido um compressor que integra técnicas de predição envolvendo duas camadas de redes biLSTM com mapeamento das posições deltas dos erros de predição, anotadas em um *Codebook*. Este modelo, cujo experimento foi nomeado como *CnnDoubleBiLstmBAC*, demonstrou-se eficiente em termos de percentual de Economia de Espaço em comparação com os resultados de compressão das ferramentas *baseline* encontradas na litera-

tura, cujos experimentos foram nomeados como *CnnLstmARE* e *CnnBiLstmAttARE*.

Adicionalmente, os testes de compressão das 100 sequências mitocondriais revelaram que o modelo *CnnLstmBAC* superou todas as outras combinações testadas, inclusive o o modelo (*CnnDoubleBiLstmBAC*), em termos de Economia de Espaço. Esta descoberta ressalta a importância da escolha de estratégias de transformação dos dados na otimização da compressão de dados genômicos. A combinação do modelo de predição *CnnLstm* com a codificação da *bitstring* de predição baseada em *codebook* (BAC), especificamente, mostrou-se a mais eficaz entre os outros modelos avaliados, sugerindo que a simplicidade do modelo preditivo, combinado com uma técnica eficiente de transformação de dados, obteve um ganho de desempenho em comparação às abordagens mais complexas estudadas.

As limitações mais importantes são as seguintes:

- Ter trabalhado com sequências pequenas.
- O tamanho do modelo preditivo gerado em Megabytes em relação a uma sequência de referência.

5.2 Trabalhos futuros

As contribuições avançaram sobre compressão de dados genômicos. O protocolo de avaliação padronizado é essencial para validar e comparar novas técnicas de compressão. Ao mesmo tempo, o compressor de genoma baseado em modelo preditivo e a codificação baseada em *Codebook* pode ser visto como uma nova referência na área, promovendo o desenvolvimento de soluções ainda mais avançadas no campo.

Os passos seguintes envolvem a viabilidade de uma ferramenta baseada em um modelo preditivo e codificação baseada em *Codebook* adaptada a sequências genômicas maiores, como os genomas humano e de plantas, ou mesmo estendida a todos os organismos. Este objetivo demanda infraestrutura capaz

de processar o aprendizado de máquina para grandes sequências genômicas. Portanto, uma direção para pesquisas futuras é o desenvolvimento de métodos e tecnologias que possibilitem a aplicação eficiente desses modelos de compressão em uma escala muito maior. Isto suplantaria limitações técnicas significativas e expandiria o campo de aplicação para abranger uma variedade mais ampla de dados genômicos.

Além dos objetivos mencionados acima, outra abordagem que se pretende é viabilizar o aprendizado profundo sobre as sequências de DNA por meio da implementação de camadas personalizadas de redes neurais, em vez de expandir o vetor com características pré-extraídas das sequências. Esta última estratégia mostrou-se menos eficaz do que a manutenção do uso exclusivo da codificação *one-hot*. A inclusão de novas camadas na arquitetura do modelo de predição visa a captura e o processamento eficiente das informações contidas nas sequências, com o objetivo de aprimorar a precisão na predição do próximo símbolo.

5.3 Outras Contribuições do Autor

Algumas contribuições foram realizadas na área de compressão de sequências genômicas. Kredens et al. (2018) propuseram uma abordagem baseada em imagens para a compressão de genomas. Este trabalho foi apresentado no livro "*Artificial Intelligence and Soft Computing*", editado por *Rutkowski et al.*, e publicado pela *Springer International Publishing*, evidenciando um método de compressão de dados genômicos.

Em um estudo correlato, Martins et al. (2018) apresentaram uma abordagem para a compressão de dados genômicos também baseada no formato de arquivo de imagem. Este estudo foi detalhado na conferência *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, destacando o potencial dos formatos de compressão baseado em formato de arquivo de imagem para dados genômicos.

Por fim, Kredens et al. (2020) realizaram uma revisão sistemática da literatura sobre ferramentas de compressão de dados genômicos verticais sem perdas de informação para genomas montados. Publicado na revista *PLOS ONE*, o estudo fornece uma análise abrangente das ferramentas disponíveis, destacando a necessidade de desenvolvimento contínuo em métodos de compressão genômica eficientes.

A seguir, apresentam-se as referências das publicações relacionadas às contribuições realizadas pelo autor deste trabalho:

KREDENS, et al. Genome Compression: An Image-Based Approach. In: RUTKOWSKI, Leszek et al. (Ed.). Artificial Intelligence and Soft Computing. Cham: Springer International Publishing, 2018. p. 240-249.

MARTINS, et al. An Approach to Compression of Genomic Data Based on Image File Format. In: 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC). 2018. p. 3274-3279.

KREDENS, et al. Vertical lossless genomic data compression tools for assembled genomes: A systematic literature review. PLOS ONE, v. 15, n. 5, p. 1-37, maio 2020.

5.4 Apoio Institucional

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Referências

AGARWALA, R. et al. Database resources of the national center for biotechnology information. *Nucleic Acids Research*, v. 44, n. D1, p. D7–D19, 01 2016. ISSN 0305-1048. Disponível em: <https://academic.oup.com/nar/article-lookup/doi/10.1093/nar/gkv1290>.

AHN, S.-M. et al. The first korean genome sequence and analysis: full genome sequencing for a socio-ethnic group. *Genome research*, Cold Spring Harbor Laboratory Press, v. 19, n. 9, p. 1622–1629, 09 2009. ISSN 1549-5469. Disponível também em: https://www.ncbi.nlm.nih.gov/pmc/PMC2752128/. Disponível em: <https://www.ncbi.nlm.nih.gov/pubmed/19470904>.

ALVES, F. et al. On-demand indexing for referential compression of dna sequences. *PLOS ONE*, Public Library of Science, v. 10, n. 7, p. e0132460, 07 2015. Disponível em: <https://doi.org/10.1371/journal.pone.0132460>.

ARRAM, J. et al. Fpga acceleration of reference-based compression for genomic data. In: *2015 International Conference on Field Programmable Technology (FPT)*. [S.l.: s.n.], 2015. p. 9–16. ISBN VO -.

BALDI, P. et al. Exploiting the past and the future in protein secondary structure prediction . *Bioinformatics*, v. 15, n. 11, p. 937–946, 11 1999. ISSN 1367-4803. Disponível em: <https://doi.org/10.1093/bioinformatics/15.11.937>.

BEAL ALIYA FARHEEN, D. A. R. Compressing genome resequencing data via the maximal longest factor. *3D Digital Imaging and Modeling, International Conference on*, IEEE Computer Society, Los Alamitos, CA, USA, p. 92–97, 2016.

BEAL, R. et al. A new algorithm for the LCS problem with application in compressing genome resequencing data. *BMC Genomics*, v. 17, n. 4, p. 544, 08 2016. ISSN 1471-2164. Disponível em: <https://doi.org/10.1186/s12864-016-2793-0>.

BEHZADI, B.; Le Fessant, F. Dna compression challenge revisited: A dynamic programming approach bt - combinatorial pattern matching. In: APOSTOLICO, A. et al. (Ed.). Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. p. 190–200. ISBN 978-3-540-31562-9.

BENESTY, J. et al. Pearson correlation coefficient. In: *Noise reduction in speech processing*. Berlin: Springer, 2009. p. 37–40.

BHATTACHARYYA, M.; BANDYOPADHYAY, S. Recent directions in compressing next generation sequencing data. *Curr. Bioinform.*, v. 7, n. 1, p. 2–6, 2012. ISSN 15748936.

BIJI, C. L.; NAIR, A. S. Benchmark dataset for whole genome sequence compression. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, IEEE, v. 14, n. 6, p. 1228–1236, 11 2017. ISSN 1545-5963. Disponível em: <https://ieeexplore.ieee.org/document/7470248/>.

BOOMSMA, D. I. et al. The genome of the netherlands: design, and project goals. *European journal of human genetics : EJHG*, Nature Publishing Group, v. 22, n. 2, p. 221–227, 02 2014. ISSN 1476-5438. Disponível em: <https://www.ncbi.nlm.nih.gov/pubmed/23714750https://www.ncbi.nlm.nih.gov/pmc/PMC3895638/>.

BRANDON, M. C. et al. Data structures and compression algorithms for genomic sequence data. *Bioinformatics (Oxford, England)*, Oxford University Press, v. 25, n. 14, p. 1731–1738, 07 2009. ISSN 1367-4811. Disponível em: <https://www.ncbi.nlm.nih.gov/pubmed/19447783https://www.ncbi.nlm.nih.gov/pmc/PMC2705231/>.

BROWN, C. G.; CLARKE, J. Nanopore development at oxford nanopore. *Nature Biotechnology*, Nature Publishing Group, a division of Macmillan Publishers Limited. All Rights Reserved., v. 34, p. 810, 08 2016. Disponível em: <https://doi.org/10.1038/nbt.3622http://10.0.4.14/nbt.3622>.

CANNANE, A.; WILLIAMS, H. E. General-purpose compression for efficient retrieval. *Journal of the American Society for Information Science and Technology*, Wiley-Blackwell, v. 52, n. 5, p. 430–437, 02 2001. ISSN 1532-2882. Disponível em: <https://doi.org/10.1002/1532-2890(2001)9999:9999{\%}3C::AID-ASI1084{\%}3E3.0.COhttp>.

CHAISSON, M. J. P. et al. Genetic variation and the de novo assembly of human genomes. *Nature reviews. Genetics*, v. 16, n. 11, p. 627–640, 11 2015. ISSN 1471-0064. Disponível em: <https://www.ncbi.nlm.nih.gov/pubmed/26442640https://www.ncbi.nlm.nih.gov/pmc/PMC4745987/>.

CHENG, K. et al. Compression of multiple dna sequences using intra-sequence and inter-sequence similarities. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, v. 12, n. 6, p. 1322–1332, 2015. ISSN 1545-5963 VO - 12.

CHENG, K. O. et al. Clustering-based compression for population dna sequences. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, p. 1, 2018. ISSN 1545-5963 VO -.

CHERN, B. G. et al. Reference based genome compression. *2012 IEEE Information Theory Workshop, ITW 2012*, p. 427–431, 2012.

CHOLLET, F. et al. *Keras*. GitHub, 2015. Disponível em: <https://keras.io>.

CHRISTLEY, S. et al. Human genomes as email attachments. *Bioinformatics*, v. 25, n. 2, p. 274–275, 01 2009. ISSN 1367-4803. Disponível em: <http://dx.doi.org/10.1093/bioinformatics/btn582>.

CLEARY, J.; WITTEN, I. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, v. 32, n. 4, p. 396–402, 1984. ISSN 0090-6778 VO - 32.

COCK, P. J. A. et al. The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. *Nucleic Acids Research*, v. 38, n. 6, p. 1767–1771, 12 2009. ISSN 0305-1048. Disponível em: <https://doi.org/10.1093/nar/gkp1137>.

COLABORATORY, G. *Colaboratory*. 2024. Acessado em 28 de feveriro de 2024. Disponível em: <https://colab.research.google.com>.

CONNELL, J. B. A huffman-shannon-fano code. *Proceedings of the IEEE*, v. 61, n. 7, p. 1046–1047, 1973. ISSN 0018-9219 VO - 61.

CONSORTIUM, . G. P. et al. A map of human genome variation from population-scale sequencing. *Nature*, v. 467, n. 7319, p. 1061–1073, 10 2010. ISSN 1476-4687. Disponível em: <https://www.ncbi.nlm.nih.gov/pubmed/20981092https://www.ncbi.nlm.nih.gov/pmc/PMC3042601/>.

CONSORTIUM, I. H. G. S. et al. Initial sequencing and analysis of the human genome. *Nature*, Macmillan Magazines Ltd., v. 409, p. 860, 02 2001. Disponível em: <https://doi.org/10.1038/35057062http://10.0.4.14/35057062https://www.nature.com/articles/35057062{\#}supplementary-informat>.

CONSORTIUM, U. et al. The uk10k project identifies rare variants in health and disease. *Nature*, v. 526, n. 7571, p. 82–90, 10 2015. ISSN 1476-4687. Disponível em: <https://www.ncbi.nlm.nih.gov/pubmed/26367797https://www.ncbi.nlm.nih.gov/pmc/PMC4773891/>.

COX, A. J. et al. Rlzap: Relative lempel-ziv with adaptive pointers bt - string processing and information retrieval. In: INENAGA, S. et al. (Ed.). Cham: Springer International Publishing, 2016. p. 1–14. ISBN 978-3-319-46049-9.

CROCHEMORE, M.; ILIE, L. Computing longest previous factor in linear time and applications. *Information Processing Letters*, v. 106, n. 2, p. 75–80, 2008. ISSN 0020-0190. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0020019007002979>.

CUI, W. et al. Compressing genomic sequences by using deep learning. In: FARKAŠ, I. et al. (Ed.). *Artificial Neural Networks and Machine Learning – ICANN 2020*. Cham: Springer International Publishing, 2020. p. 92–104. ISBN 978-3-030-61609-0.

DAI, W. et al. An adaptive difference distribution-based coding with hierarchical tree structure for dna sequence compression. *Proceedings. Data Compression Conference*, v. 2013, p. 371–380, 2013. ISSN 2375-0383. Disponível em: <https://www.ncbi.nlm.nih.gov/pubmed/26501129https://www.ncbi.nlm.nih.gov/pmc/PMC4617277/>.

DANECEK, P. et al. The variant call format and vcftools. *Bioinformatics (Oxford, England)*, Oxford University Press, v. 27, n. 15, p. 2156–2158, 08 2011. ISSN 1367-4811. Disponível em: <https://www.ncbi.nlm.nih.gov/pubmed/21653522https://www.ncbi.nlm.nih.gov/pmc/PMC3137218/>.

DANEK, A.; DEOROWICZ, S. Gtc: a novel attempt to maintenance of huge genome collections compressed. *bioRxiv*, Cold Spring Harbor Laboratory, 2017. Disponível em: <https://www.biorxiv.org/content/early/2017/08/29/131649>.

DANIEL, W. *Applied Nonparametric Statistics*. PWS-KENT Pub., 1990. (Duxbury advanced series in statistics and decision sciences). ISBN 9780534919764. Disponível em: <https://books.google.com.br/books?id=0hPvAAAAMAAJ>.

DEOROWICZ, S. et al. Genome compression: a novel approach for large collections. *Bioinformatics*, v. 29, n. 20, p. 2572–2578, 10 2013. ISSN 1367-4803. Disponível em: <http://dx.doi.org/10.1093/bioinformatics/btt460>.

DEOROWICZ, S. et al. Gdc 2: Compression of large collections of genomes. *Scientific Reports*, The Author(s), v. 5, p. 11565, 06 2015. Disponível em: <https://doi.org/10.1038/srep11565http://10.0.4.14/srep11565https://www.nature.com/articles/srep11565{\#}supplementary-informat>.

DEOROWICZ, S.; GRABOWSKI, S. Robust relative compression of genomes with random access. *Bioinformatics*, v. 27, n. 21, p. 2979–2986, 11 2011. ISSN 1367-4803. Disponível em: <http://dx.doi.org/10.1093/bioinformatics/btr505>.

DEOROWICZ, S.; GRABOWSKI, S. Data compression for sequencing data. *Algorithms for Molecular Biology*, v. 8, n. 1, p. 25, 11 2013. ISSN 1748-7188. Disponível em: <https://doi.org/10.1186/1748-7188-8-25>.

DEOROWICZ, S. et al. Comment on: "ergc: an efficient referential genome compression algorithm". *Bioinformatics (Oxford, England)*, Oxford University Press, v. 32, n. 7, p. 1115–1117, 04 2016. ISSN 1367-4811. Disponível em: <https://www.ncbi.nlm.nih.gov/pubmed/26615213https://www.ncbi.nlm.nih.gov/pmc/PMC4907388/>.

DU, Z. et al. Porting referential genome compression tool on loongson platform bt - parallel architecture, algorithm and programming. In: CHEN, G. et al. (Ed.). Singapore: Springer Singapore, 2017. p. 454–463. ISBN 978-981-10-6442-5.

DUPOND, S. A thorough review on the current advance of neural network structures. *Annual Reviews in Control*, v. 14, n. 14, p. 200–230, 2019.

DURBIN, R. Efficient haplotype matching and storage using the positional burrows-wheeler transform (pbwt). *Bioinformatics (Oxford, England)*, Oxford University Press, v. 30, n. 9, p. 1266–1272, may 2014. ISSN 1367-4811. Disponível em: <https://www.ncbi.nlm.nih.gov/pubmed/24413527https://www.ncbi.nlm.nih.gov/pmc/PMC3998136/>.

ELIAS, P. Universal codeword sets and representations of the integers. *IEEE Transactions on Information Theory*, v. 21, n. 2, p. 194–203, 1975. ISSN 0018-9448 VO - 21.

ELSEVIER. *Scopus [Internet]*. 2004. Disponível em: <http://www.scopus.com/>.

ELSEVIER. *Scopus Content Coverage Guide*. 2023. Disponível em: <https://beta.elsevier.com/products/scopus/content?trial=true>.

Encyclopaedia Britannica. *Machine Learning*. 2009. Acessado em 8 de janeiro de 2024. Disponível em: <https://www.britannica.com/technology/machine-learning>.

FAIRLEY, S. et al. The International Genome Sample Resource (IGSR) collection of open human genomic variation resources. *Nucleic Acids Research*, v. 48, n. D1, p. D941–D947, 10 2019. ISSN 0305-1048. Disponível em: <https://doi.org/10.1093/nar/gkz836>.

FAN, W. et al. Complementary contextual models with fm-index for dna compression. In: *2017 Data Compression Conference (DCC)*. [S.l.: s.n.], 2017. p. 82–91. ISBN 2375-0359 VO -.

FERRADA, H. et al. Relative lempel-ziv with constant-time random access bt - string processing and information retrieval. In: MOURA, E.; CROCHEMORE, M. (Ed.). Cham: Springer International Publishing, 2014. p. 13–17. ISBN 978-3-319-11918-2.

FERRAGINA, P. et al. An alphabet-friendly fm-index bt - string processing and information retrieval. In: APOSTOLICO, A.; MELUCCI, M. (Ed.). Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. p. 150–160. ISBN 978-3-540-30213-1.

FRIEDMAN, M. The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. *Journal of the American Statistical Association*, v. 32, n. 200, p. 675–701, 1937.

FRITZ, M. H.-Y. et al. Efficient storage of high throughput dna sequencing data using reference-based compression. *Genome Research*, v. 21, n. 5, p. 734–740, 2011. Disponível em: <http://genome.cshlp.org/content/21/5/734.abstract>.

FUKUSHIMA, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, v. 36, n. 4, p. 193–202, 1980. ISSN 1432-0770. Disponível em: <https://doi.org/10.1007/BF00344251>. Acesso em: [data de acesso]. Disponível em: <https://doi.org/10.1007/BF00344251>.

GHOLAMY, A. et al. *Why 70/30 or 80/20 Relation Between Training and Testing Sets: A Pedagogical Explanation*. [S.l.], 2018. Disponível em: <https://scholarworks.utep.edu/cs_techrep/1209>.

GIANCARLO, R. et al. Compressive biological sequence analysis and archival in the era of high-throughput sequencing technologies. *Briefings in Bioinformatics*, v. 15, n. 3, p. 390–406, may 2014. ISSN 1467-5463. Disponível em: <http://dx.doi.org/10.1093/bib/bbt088>.

GIANCARLO, R. et al. Textual data compression in computational biology: a synopsis. *Bioinformatics*, v. 25, n. 13, p. 1575–1586, jul 2009. ISSN 1367-4803. Disponível em: <http://dx.doi.org/10.1093/bioinformatics/btp117>.

GIANCARLO, R. et al. Textual data compression in computational biology: Algorithmic techniques. *Computer Science Review*, v. 6, n. 1, p. 1–25, 2012. ISSN 1574-0137. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1574013711000311>.

GLOROT, X.; BENGIO, Y. Understanding the difficulty of training deep feedforward neural networks. In: TEH, Y. W.; TITTERINGTON, M. (Ed.). *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Chia Laguna Resort, Sardinia, Italy: PMLR, 2010. (Proceedings of Machine Learning Research, v. 9), p. 249–256. Disponível em: <https://proceedings.mlr.press/v9/glorot10a.html>.

GOG, S. et al. From theory to practice: Plug and play with succinct data structures. In: *13th International Symposium on Experimental Algorithms, (SEA 2014)*. [S.l.: s.n.], 2014. p. 326–337.

GOLOMB, S. Run-length encodings (corresp.). *IEEE Transactions on Information Theory*, v. 12, n. 3, p. 399–401, 1966. ISSN 0018-9448 VO - 12.

GOYAL, M. et al. Deepzip: Lossless data compression using recurrent neural networks. In: *2019 Data Compression Conference (DCC)*. [S.l.: s.n.], 2019. p. 575–575.

GRABOWSKI, S.; KOWALSKI, T. M. MBGC: Multiple Bacteria Genome Compressor. *GigaScience*, v. 11, p. giab099, 01 2022. ISSN 2047-217X. Disponível em: <https://doi.org/10.1093/gigascience/giab099>.

GROUP, N. W. et al. Rfc 1952: Gzip file format specification version 4.3. *Distribution*, 1996.

GRUMBACH, S.; TAHI, F. Compression of dna sequences. In: *[Proceedings] DCC 93: Data Compression Conference*. [S.l.: s.n.], 1993. p. 340–350. ISBN VO -.

GRUMBACH, S.; TAHI, F. A new challenge for compression algorithms: Genetic sequences. *Information Processing & Management*, v. 30, n. 6, p. 875–886, 1994. ISSN 0306-4573. Disponível em: <http://www.sciencedirect.com/science/article/pii/0306457394900140>.

GUSFIELD, D. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. New York, NY, USA: Cambridge University Press, 1997. ISBN 0-521-58519-8.

HAYDEN, E. C. *Genome researchers raise alarm over big data*. 2015. Disponível em: <https://www.nature.com/news/genome-researchers-raise-alarm-over-big-data-1.17912>.

HEATHER, J. M.; CHAIN, B. The sequence of sequencers: The history of sequencing dna. *Genomics*, v. 107, n. 1, p. 1–8, 2016. ISSN 0888-7543. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0888754315300410>.

HIRSCHBERG, D. S.; LELEWER, D. A. Efficient decoding of prefix codes. *Commun. ACM*, ACM, New York, NY, USA, v. 33, n. 4, p. 449–459, 04 1990. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/77556.77566>.

HOCHREITER, S.; SCHMIDHUBER, J. Long Short-Term Memory. *Neural Computation*, v. 9, n. 8, p. 1735–1780, 11 1997. ISSN 0899-7667. Disponível em: <https://doi.org/10.1162/neco.1997.9.8.1735>.

HOLLEY, G. et al. Bloom filter trie: an alignment-free and reference-free data structure for pan-genome storage. *Algorithms for molecular biology : AMB*, BioMed Central, v. 11, p. 3, 04 2016. ISSN 1748-7188. Disponível em: <https://www.ncbi.nlm.nih.gov/pubmed/27087830https://www.ncbi.nlm.nih.gov/pmc/PMC4832552/>.

HOSSEINI, M. et al. *A survey on data compression methods for biological sequences*. 2016.

HUANG, Y.; LIANG, Y. A secure arithmetic coding algorithm based on integer implementation. In: *2011 11th International Symposium on Communications & Information Technologies (ISCIT)*. [S.l.: s.n.], 2011. p. 518–521. ISBN VO -.

HUANG, Z. et al. A privacy-preserving solution for compressed storage and selective retrieval of genomic data. *Genome research*, Cold Spring Harbor Laboratory Press, v. 26, n. 12, p. 1687–1696, 12 2016. ISSN 1549-5469. Disponível em: <https://www.ncbi.nlm.nih.gov/pubmed/27789525https://www.ncbi.nlm.nih.gov/pmc/PMC5131820/>.

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

IGLESIAS, A. I. et al. Haplotype reference consortium panel: Practical implications of imputations with large reference panels. *Human Mutation*, v. 38, n. 8, p. 1025–1032, 08 2017. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/humu.23247>.

IUBMB-IUPAC. International Union of Pure and Applied Joint Commission on Biochemical Nomenclature. *Pure & Applied Chemistry*, v. 56, n. 5, p. 595–624, 1984.

J. Craig Venter, Mark D. Adams, Eugene W. Myers, Peter W. Li, R. J. M. et al. The sequence of the human genome. *Science (New York, N.Y.)*, v. 291, n. February, p. 1304–1351, 2001. ISSN 00368075. Disponível em: <www.sciencemag.org>.

JOLLIFFE, I. T. *Principal Component Analysis*. Second. New York: Springer-Verlag, 2002. (Springer Series in Statistics).

KAHN, S. D. On the future of genomic data. *Science*, American Association for the Advancement of Science, v. 331, n. 6018, p. 728–729, 2011. ISSN 0036-8075. Disponível em: <http://science.sciencemag.org/content/331/6018/728>.

KITCHENHAM, B.; CHARTERS, S. *Guidelines for performing Systematic Literature Reviews in Software Engineering*. 2007.

KNOLL, B. *CMIX lossless data compression*. 2021. Disponível em: <http://www.byronknoll.com/cmix.html>.

KNUTH, D. E. *The Art of Computer Programming, Volume 3: (2Nd Ed.) Sorting and Searching*. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc., 1998. ISBN 0-201-89685-0.

KRAFT, F.; KURTH, I. Long-read sequencing in human genetics. *Medizinische Genetik*, v. 31, n. 2, p. 198–204, 2019. Disponível em: <https://doi.org/10.1007/s11825-019-0249-z>.

KREDENS, K. V. et al. Genome compression: An image-based approach. In: RUTKOWSKI, L. et al. (Ed.). *Artificial Intelligence and Soft Computing*. Cham: Springer International Publishing, 2018. p. 240–249. ISBN 978-3-319-91262-2.

KREDENS, K. V. et al. Vertical lossless genomic data compression tools for assembled genomes: A systematic literature review. *PLOS ONE*, Public Library of Science, v. 15, n. 5, p. 1–37, 05 2020. Disponível em: <https://doi.org/10.1371/journal.pone.0232942>.

KREFT, S.; NAVARRO, G. Lz77-like compression with fast random access. In: *Proceedings of the 2010 Data Compression Conference*. Washington, DC, USA: IEEE Computer Society, 2010. (DCC '10), p. 239–248. ISBN 978-0-7695-3994-2. Disponível em: <https://doi.org/10.1109/DCC.2010.29>.

KURUPPU, S. et al. Iterative dictionary construction for compression of large dna data sets. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, v. 9, n. 1, p. 137–149, 2012. ISSN 1545-5963 VO - 9.

KURUPPU, S. et al. Relative lempel-ziv compression of genomes for large-scale storage and retrieval bt - string processing and information retrieval. In: CHAVEZ, E.; LONARDI, S. (Ed.). Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 201–206. ISBN 978-3-642-16321-0.

KURUPPU, S. et al. Optimized relative lempel-ziv compression of genomes. *Conferences in Research and Practice in Information Technology Series*, v. 113, p. 91–98, 2011. ISSN 14451336.

KURUPPU, S. et al. Reference sequence construction for relative compression of genomes bt - string processing and information retrieval. In: GROSSI, R. et al. (Ed.). Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 420–425. ISBN 978-3-642-24583-1.

LARSSON, N. J.; MOFFAT, A. Off-line dictionary-based compression. *Proceedings of the IEEE*, v. 88, n. 11, p. 1722–1732, 2000. ISSN 0018-9219 VO - 88.

LECUN, Y.; BENGIO, Y. Convolutional networks for images, speech, and time-series. In: . [S.l.: s.n.], 1995.

LEVY, S. et al. The diploid genome sequence of an individual human. *PLoS biology*, Public Library of Science, v. 5, n. 10, p. e254–e254, 10 2007. ISSN 1545-7885. Disponível em: <https://www.ncbi.nlm.nih.gov/pubmed/17803354https://www.ncbi.nlm.nih.gov/pmc/PMC1964779/>.

LI, H. Bgt: efficient and flexible genotype query across many samples. *Bioinformatics*, v. 32, n. 4, p. 590–592, 2016. Disponível em: <http://dx.doi.org/10.1093/bioinformatics/btv613>.

LI, P. et al. Dna-compact: Dna compression based on a pattern-aware contextual modeling technique. *PloS one*, Public Library of Science, v. 8, n. 11, p. e80377–e80377, 11 2013. ISSN 1932-6203. Disponível em: <https://www.ncbi.nlm.nih.gov/pubmed/24282536https://www.ncbi.nlm.nih.gov/pmc/PMC3840021/>.

LIU, Y. et al. High-speed and high-ratio referential genome compression. *Bioinformatics*, v. 33, n. 21, p. 3364–3372, 11 2017. ISSN 1367-4803. Disponível em: <http://dx.doi.org/10.1093/bioinformatics/btx412>.

LIU, Y. et al. Allowing mutations in maximal matches boosts genome compression performance. *Bioinformatics*, v. 36, n. 18, p. 4675–4681, 06 2020. ISSN 1367-4803. Disponível em: <https://doi.org/10.1093/bioinformatics/btaa572>.

MANZINI, G.; RASTERO, M. A simple and fast dna compressor. *Software: Practice and Experience*, Wiley-Blackwell, v. 34, n. 14, p. 1397–1411, 08 2004. ISSN 0038-0644. Disponível em: <https://doi.org/10.1002/spe.619>.

MARTINS, J. V. et al. An approach to compression of genomic data based on image file format. In: *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. [S.l.: s.n.], 2018. p. 3274–3279.

MATSUGU, M. et al. Subject independent facial expression recognition with robust face detection using a convolutional neural network. *Neural Networks*, v. 16, n. 5, p. 555–559, 2003. ISSN 0893-6080. Advances in Neural Networks Research: IJCNN '03. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0893608003001151>.

MCANLIS, C.; HAECKY, A. *Understanding Compression: Data Compression for Modern Developers*. 1st. ed. [S.l.]: O'Reilly Media, Inc., 2016. ISBN 1491961538, 9781491961537.

MEDICINE, N. L. of. *PubMed Journals Currently Indexed*. 2023. Disponível em: <https://www.nlm.nih.gov/bsd/serfile_addedinfo.html>.

MEHTA, K.; GHRERA, S. P. Dna compression using referential compression algorithm. In: *2015 Eighth International Conference on Contemporary Computing (IC3)*. [S.l.: s.n.], 2015. p. 64–69.

MESELSON, M.; STAHL, F. W. The replication of dna in escherichia coli. *Proceedings of the National Academy of Sciences*, National Academy of Sciences, v. 44, n. 7, p. 671–682, 1958. ISSN 0027-8424. Disponível em: <https://www.pnas.org/content/44/7/671>.

MOFFAT, A. Implementing the ppm data compression scheme. *IEEE Transactions on Communications*, v. 38, n. 11, p. 1917–1921, 1990. ISSN 0090-6778 VO - 38.

MONTECUOLLO, F. et al. E2fm: an encrypted and compressed full-text index for collections of genomic sequences. *Bioinformatics*, v. 33, n. 18, p. 2808–2817, 09 2017. ISSN 1367-4803. Disponível em: <http://dx.doi.org/10.1093/bioinformatics/btx313>.

MYERS, E. W. Ano(nd) difference algorithm and its variations. *Algorithmica*, v. 1, n. 1, p. 251–266, 1986. ISSN 1432-0541. Disponível em: <https://doi.org/10.1007/BF01840446>.

MäKINEN, V. et al. Storage and retrieval of highly repetitive sequence collections. *Journal of computational biology : a journal of computational molecular cell biology*, 2010. ISSN 1557-8666.

MäKINEN, V. et al. Storage and retrieval of highly repetitive sequence collections. *Journal of computational biology: a journal of computational molecular cell biology*, v. 17 3, p. 281–308, 2010.

NAGARAJAN, N.; POP, M. Sequence assembly demystified. *Nature Reviews Genetics*, Nature Publishing Group, a division of Macmillan Publishers Limited. All Rights Reserved., v. 14, p. 157, 01 2013. Disponível em: <https://doi.org/10.1038/nrg3367http://10.0.4.14/nrg3367https://www.nature.com/articles/nrg3367{\#}supplementary-informat>.

NALBANTOGLU, O. U. et al. Data compression concepts and algorithms and their applications to bioinformatics. *Entropy (Basel, Switzerland)*, v. 12, n. 1, p. 34, 01 2010. ISSN 1099-4300. Disponível em: <https://www.ncbi.nlm.nih.gov/pubmed/20157640https://www.ncbi.nlm.nih.gov/pmc/PMC2821113/>.

NEMENYI, P. *Distribution-free Multiple Comparisons*. 263 p. Tese (Doutorado) — Princeton University, Princeton, NJ, 1962.

NEUMANN, T.; LEIS, V. Compiling database queries into machine code. *IEEE Data Eng. Bull.*, v. 37, n. 1, p. 3–11, 2014. Disponível em: <http://dblp.uni-trier.de/db/journals/debu/debu37.html#0001L14>.

NOVAK, A. M. et al. A graph extension of the positional burrows-wheeler transform and its applications bt - algorithms in bioinformatics. In: . Cham: Springer International Publishing, 2016. p. 246–256. ISBN 978-3-319-43681-4.

NUMANAGIć, I. et al. Comparison of high-throughput sequencing data compression tools. *Nature Methods*, Nature Publishing Group, a division of Macmillan Publishers Limited. All Rights Reserved., v. 13, n. 12, p. 1005–1008, 10 2016. ISSN 15487105. Disponível em: <https://doi.org/10.1038/nmeth.4037http://10.0.4.14/nmeth.4037https://www.nature.com/articles/nmeth.4037{\#}supplementary-informat>.

OCHOA, I. et al. idocomp: a compression scheme for assembled genomes. *Bioinformatics (Oxford, England)*, Oxford University Press, v. 31, n. 5, p. 626–633, 03 2015. ISSN 1367-4811. Disponível em: <https://www.ncbi.nlm.nih.gov/pubmed/25344501https://www.ncbi.nlm.nih.gov/pmc/PMC5855886/>.

OHLEBUSCH, E. et al. Cst++ bt - string processing and information retrieval. In: CHAVEZ, E.; LONARDI, S. (Ed.). Berlin, Heidelberg: Springer Berlin Heidelberg, 2010. p. 322–333. ISBN 978-3-642-16321-0.

OKANOHARA, D.; SADAKANE, K. Practical entropy-compressed rank/select dictionary. In: *Proceedings of the Meeting on Algorithm Engineering & Expermiments*.

Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2007. p. 60–70. Disponível em: <http://dl.acm.org/citation.cfm?id=2791188.2791194>.

PAPAGEORGIOU, L. et al. Genomic big data hitting the storage bottleneck. *EMBnet.journal*, v. 24, p. e910, 2018. ISSN 2226-6089. Epub 2018 Apr 19. Disponível em: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5958914>.

PAVLICHIN, D. S. et al. The human genome contracts again. *Bioinformatics*, v. 29, n. 17, p. 2199–2202, 09 2013. ISSN 1367-4803. Disponível em: <http://dx.doi.org/10.1093/bioinformatics/btt362>.

PAVLOV, I. *7-Zip: A File Archiver with High Compression Ratio*. https://www.7-zip.org/, 2023. Acessado em 20 de junho de 2023.

PEEL, A. et al. Collection-based compression using discovered long matching strings. In: *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*. New York, NY, USA: ACM, 2011. (CIKM '11), p. 2361–2364. ISBN 978-1-4503-0717-8. Disponível em: <http://doi.acm.org/10.1145/2063576.2063967>.

PINHO, A. J. et al. Green: a tool for efficient compression of genome resequencing data. *Nucleic acids research*, v. 40, n. 4, p. e27, 02 2012. ISSN 1362-4962. Disponível em: <https://academic.oup.com/nar/article/40/4/e27/2411074http://www.ncbi.nlm.nih.gov/pubmed/22139935http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC3287168>.

PRATAS, D. et al. Substitutional tolerant markov models for relative compression of dna sequences bt - 11th international conference on practical applications of computational biology & bioinformatics. In: FDEZ-RIVEROLA, F. et al. (Ed.). Cham: Springer International Publishing, 2017. p. 265–272. ISBN 978-3-319-60816-7.

PRATAS, D.; PINHO, A. J. A dna sequence corpus for compression benchmark. In: *Advances in Intelligent Systems and Computing*. [s.n.], 2019. v. 803, p. 208–215. ISBN 9783319987019. ISSN 21945357. Disponível em: <http://link.springer.com/10.1007/978-3-319-98702-6{_}.>

PRATAS, D. et al. Efficient compression of genomic sequences. In: *2016 Data Compression Conference (DCC)*. [S.l.: s.n.], 2016. p. 231–240. ISBN 1068-0314 VO -.

PRATAS, D. et al. Efficient compression of genomic sequences. In: *2016 Data Compression Conference (DCC)*. [S.l.: s.n.], 2016. p. 231–240. ISBN 1068-0314 VO -.

PROCHáZKA, P.; HOLUB, J. Compressing similar biological sequences using fm-index. *2014 Data Compression Conference*, p. 312–321, 2014.

PubMed Central (PMC). *PubMed [Internet]*. 2023. National Library of Medicine (US), 2023. Disponível em: <https://pubmed.ncbi.nlm.nih.gov>. Acesso em: [data de acesso].

PYTHON. 2024. <https://www.python.org/>. Acessado em 28 de feveriro de 2024.

QUANG, D.; XIE, X. Danq: a hybrid convolutional and recurrent deep neural network for quantifying the function of dna sequences. *Nucleic Acids Research*, v. 44, n. 11, p. e107–e107, 04 2016. ISSN 0305-1048. Disponível em: <https://doi.org/10.1093/nar/gkw226>.

REUTER, J. A. et al. High-throughput sequencing technologies. *Mol Cell*, v. 58, n. 4, p. 586–597, 05 2015. Disponível em: <https://www.ncbi.nlm.nih.gov/pubmed/26000844>.

RISSANEN, J. J. Generalized kraft inequality and arithmetic coding. *IBM Journal of Research and Development*, v. 20, n. 3, p. 198–203, 1976. ISSN 0018-8646 VO - 20.

ROSEN, B. E. et al. Adaptive range coding. *Advances in Neural Information Processing Systems*, v. 3, p. 486–492, 1991.

SAADA, B.; ZHANG, J. Dna sequences compression algorithms based on the two bits codation method. In: *Proceedings - 2015 IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2015*. [S.l.: s.n.], 2015. ISBN 9781467367981. ISSN 20780958.

SAHA, S.; RAJASEKARAN, S. Ergc: an efficient referential genome compression algorithm. *Bioinformatics (Oxford, England)*, Oxford University Press, v. 31, n. 21, p. 3468–3475, 11 2015. ISSN 1367-4811. Disponível em: <https://www.ncbi.nlm.nih.gov/pubmed/26139636https://www.ncbi.nlm.nih.gov/pmc/PMC4838057/>.

SAHA, S.; RAJASEKARAN, S. Nrgc: a novel referential genome compression algorithm. *Bioinformatics (Oxford, England)*, Oxford University Press, v. 32, n. 22, p. 3405–3412, 11 2016. ISSN 1367-4811. Disponível em: <https://www.ncbi.nlm.nih.gov/pubmed/27485445https://www.ncbi.nlm.nih.gov/pmc/PMC5939913/>.

SALOMON, D. *Data Compression: The Complete Reference*. 4th edition. ed. London, UK: Springer-Verlag London, 2007. v. 53. 1689–1699 p.

SALOMON, D.; MOTTA, G. *Handbook of Data Compression*. 5th. ed. [S.l.]: Springer Publishing Company, Incorporated, 2009. ISBN 1848829027, 9781848829022.

SAMEITH, K. et al. Iterative error correction of long sequencing reads maximizes accuracy and improves contig assembly. *Briefings in Bioinformatics*, v. 18, n. 1, p. 1–8, 01 2017. ISSN 1467-5463. Disponível em: <http://dx.doi.org/10.1093/bib/bbw003>.

SCHUSTER, M.; PALIWAL, K. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, v. 45, n. 11, p. 2673–2681, 1997.

SCHUSTER, S. C. Next-generation sequencing transforms today's biology. *Nature Methods*, Nature Publishing Group, v. 5, n. 1, p. 16–18, 12 2007. ISSN 1548-7091. Disponível em: <http://dx.doi.org/10.1038/nmeth1156>.

SCHWARTZ, E. S.; KALLICK, B. Generating a canonical prefix encoding. *Commun. ACM*, ACM, New York, NY, USA, v. 7, n. 3, p. 166–169, 03 1964. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/363958.363991>.

SHANNON, C. E. A mathematical theory of communication. *The Bell System Technical Journal*, v. 27, n. 3, p. 379–423, 1948.

SHI, W. et al. High efficiency referential genome compression algorithm. *Bioinformatics*, v. 35, n. 12, p. 2058–2065, 11 2018. ISSN 1367-4803. Disponível em: <https://doi.org/10.1093/bioinformatics/bty934>.

SHKARIN, D.; SUBBOTIN, D. *Data Compression Software*. 2006. Acessado em 8 de janeiro de 2024. Disponível em: <https://compression.ru/ds/>.

SILVA, M. et al. Efficient DNA sequence compression with neural networks. *GigaScience*, v. 9, n. 11, p. giaa119, 11 2020. ISSN 2047-217X. Disponível em: <https://doi.org/10.1093/gigascience/giaa119>.

STEINBISS, S.; KURTZ, S. A new efficient data structure for storage and retrieval of multiple biosequences. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, v. 9, n. 2, p. 345–357, 2012. ISSN 1545-5963 VO - 9.

STEPHENS, Z. D. et al. Big data: Astronomical or genomical? *PLOS Biology*, Public Library of Science, v. 13, n. 7, p. 1–11, 07 2015. Disponível em: <https://doi.org/10.1371/journal.pbio.1002195>.

TAN, L.; SUN, J. K-means clustering based compression algorithm for the high-throughput dna sequence. *2014 International Conference on Audio, Language and Image Processing*, p. 952–955, 2014.

WANDELT, S. et al. Trends in genome compression. *Current Bioinformatics*, v. 9, n. 3, p. 315–326, 2014. ISSN 15748936. Disponível em: <http://www.eurekaselect.com/openurl/content.php?genre=article{&}issn=1574-8936{&}volume=9{&}issue=3{}.>

WANDELT, S.; LESER, U. Adaptive efficient compression of genomes. *Algorithms for molecular biology : AMB*, BioMed Central, v. 7, n. 1, p. 30, 11 2012. ISSN 1748-7188. Disponível em: <https://www.ncbi.nlm.nih.gov/pubmed/23146997https://www.ncbi.nlm.nih.gov/pmc/PMC3541066/>.

WANDELT, S.; LESER, U. Fresco: Referential compression of highly similar sequences. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 10, n. 5, p. 1275–1288, 09 2013. ISSN 1545-5963. Disponível em: <http://dx.doi.org/10.1109/TCBB.2013.122>.

WANDELT, S.; LESER, U. Sequence factorization with multiple references. *PloS one*, Public Library of Science, v. 10, n. 9, p. e0139000–e0139000, 09 2015. ISSN 1932-6203. Disponível em: <https://www.ncbi.nlm.nih.gov/pubmed/26422374https://www.ncbi.nlm.nih.gov/pmc/PMC4589410/>.

WANG, C.; ZHANG, D. A novel compression tool for efficient storage of genome resequencing data. *Nucleic acids research*, Oxford University Press, v. 39, n. 7, p. e45–e45, 04 2011. ISSN 1362-4962. Disponível em: <https://www.ncbi.nlm.nih.gov/pubmed/21266471https://www.ncbi.nlm.nih.gov/pmc/PMC3074166/>.

WANG, R. et al. Deepdna: a hybrid convolutional and recurrent neural network for compressing human mitochondrial genomes. In: *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. [S.l.: s.n.], 2018. p. 270–274.

WETTERSTRAND, K. *DNA Sequencing Costs: Data from the NHGRI Genome Sequencing Program (GSP)*. 2022. Disponível em: <https://www.genome.gov/about-genomics/fact-sheets/DNA-Sequencing-Costs-Data>.

Wiselin Kiruba, E.; RAMAR, K. Enhancing security for gnome data using referential compression with symmetric cryptography schema. *Asian Journal of Information Technology*, 2016. ISSN 19935994.

WOLFE, K. H. et al. Clade- and species-specific features of genome evolution in the saccharomycetaceae. *FEMS yeast research*, Oxford University Press, v. 15, n. 5, p. fov035–fov035, 08 2015. ISSN 1567-1364. Disponível em: <https://www.ncbi.nlm.nih.gov/pubmed/26066552https://www.ncbi.nlm.nih.gov/pmc/PMC4629796/>.

WU, L. et al. Direct comparison of performance of single nucleotide variant calling in human genome with alignment-based and assembly-based approaches. *Scientific reports*, Nature Publishing Group UK, v. 7, n. 1, p. 10963, 09 2017. ISSN 2045-2322. Disponível em: <https://www.ncbi.nlm.nih.gov/pubmed/28887485https://www.ncbi.nlm.nih.gov/pmc/PMC5591230/>.

XIE, X. et al. Cogi: Towards compressing genomes as an image. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, v. 12, n. 6, p. 1275–1285, 2015. ISSN 1545-5963 VO - 12.

YAN, K.; ZHANG, D. Feature selection and analysis on correlated gas sensor data with recursive feature elimination. *Sensors and Actuators B: Chemical*, v. 212, p. 353–363, 2015. Disponível em: <http://dx.doi.org/10.1016/j.snb.2015.02.025>.

YAO, H. et al. Hrcm: An efficient hybrid referential compression method for genomic big data. *BioMed Research International*, Hindawi, v. 2019, p. 3108950, 2019. ISSN 2314-6133. Disponível em: <https://doi.org/10.1155/2019/3108950>.

ZELL, A. Simulation neuronaler netze. In: . [s.n.], 1994. Disponível em: <https://api.semanticscholar.org/CorpusID:47558162>.

ZHENG, X. et al. Seqarray—a storage-efficient high-performance data format for wgs variant calls. *Bioinformatics*, v. 33, n. 15, p. 2251–2257, 2017. Disponível em: <http://dx.doi.org/10.1093/bioinformatics/btx145>.

ZHOU, J.; TROYANSKAYA, O. G. Predicting effects of noncoding variants with deep learning-based sequence model. *Nature Methods*, v. 12, n. 10, p. 931–934, Oct 2015. ISSN 1548-7105. Disponível em: <https://doi.org/10.1038/nmeth.3547>.

ZHU, Z. et al. High-throughput dna sequence data compression. *Briefings in Bioinformatics*, v. 16, n. 1, p. 1–15, 2015. Disponível em: <http://dx.doi.org/10.1093/bib/bbt087>.

ZIV, J.; LEMPEL, A. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, v. 23, n. 3, p. 337–343, 1977. ISSN 0018-9448 VO - 23.

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. ISSN 0018-9448 VO - 24.

Glossário

Alfabeto Genômico diferente de {A, T, C, G} Alfabeto genômico constituído por símbolos ASCII que são diferentes das inicais das bases genéticas Adenina, Timina, Citosina e Guanina.. 48, 85

Alfabeto Genômico {A, T, C, G} Alfabeto genômico constituído pelas inicais das bases genéticas Adenina, Timina, Citosina e Guanina.. 73

Contig É um trecho contínuo da sequência genômica composto de bases A, C, G e T. Ele é formado juntando vários pequenos pedaços de DNA sobrepostos em uma sequência mais longa. Em palavras simples, contig é um conjunto de um conjunto de fragmentos de sequência.. 3

De Novo É o método aplicado para montagem do genoma quando nenhuma sequência do genoma de referência está disponível ou a referência disponível é de baixa qualidade. Os genomas que não foram sequenciados antes devem ser montados por meio da abordagem *de novo* após o sequenciamento.. 2, 3

FASTA Formato de arquivo baseado em texto para representar sequências de nucleótidos. Quando contém sequências distintas, é denominado multi-FASTA.. 2, 3, 14, 16, 18, 20, 21, 27, 34, 35, 84

FASTQ É um formato de arquivo baseado em texto para armazenar uma sequência de nucleotídeos e suas pontuações de qualidade correspondentes.. 2

Gap É uma lacuna que separa os contigs.. 3

Haplótipos Haplótipo é um conjunto de variações de DNA, ou polimorfismos, que tendem a ser herdados juntos. Um haplótipo pode se referir a uma combinação de alelos ou a um conjunto de polimorfismos de nucleotídeo único (SNPs) encontrados no mesmo cromossomo.. 2

Naïve-bit Encoding Simples substituição de cada símbolo do alfabeto genômico (A, C, G e T) por 2 *bits*, e.g., $A = 00$, $C = 01$, $G = 10$ e $T = 11$.. 73

Resequenciamento É o método aplicado para montagem do genoma quando uma sequência do genoma de referência está disponível. As leituras de sequenciamento são alinhadas de volta à referência para determinar a localização no genoma que a leitura específica melhor corresponde.. 2, 3

Scaffold Um scaffold é uma sequência genômica reconstruída a partir de clones de genoma completo sequenciados. Estruturalmente, um scaffold consiste em contigs e gaps (lacunas). Portanto, os scaffold são criados encadeando Contigs e separando-os por gaps.. 3

A

Anexo A

Este anexo oferece uma descrição detalhada do processo de extração de características realizado durante a fase exploratória do estudo focado na compressão de sequências genômicas, com ênfase na preservação integral da informação e na utilização de modelos preditivos. O objetivo primordial desta fase foi a identificação e avaliação de uma série de características com potencial para enriquecer tanto a análise quanto o processamento das sequências genômicas, facilitando a aprendizagem dos modelos sobre as relações informacionais intrínsecas às sequências. Para isso, foi realizada uma investigação que abrangeu desde características elementares até características complexas. Apesar da consideração inicial de um leque variado de características, a seleção final recaiu sobre uma abordagem de modelagem mais simplificada, dando preferência à codificação *one-hot* devido ao seu desempenho superior nos testes de acurácia. Assim, este documento expõe um exame das características avaliadas, detalhando o método de cálculo e a dimensão designada a cada uma dentro do vetor de características. A fase exploratória foi crucial para auxiliar as decisões sobre quais técnicas seriam adotadas ao longo do estudo, auxiliando na identificação de modelos preditivos que mostram maior eficácia para a predição do próximo símbolo.

A.1 Extração de características - Fase exploratória

As características investigadas durante a fase inicial do estudo, documentadas no Anexo A, refletem o intuito de abordar distintos aspectos das sequências

genômicas. Cada característica selecionada foi considerada com base em sua potencial contribuição para o processo de aprendizagem e previsão do modelo em relação ao próximo símbolo dentro dos dados. As características selecionadas para análise estão intrinsecamente relacionadas ao tamanho do contexto considerado, o qual desempenha um papel crucial na extração de informações relevantes das sequências genômicas. De maneira específica, a subsequência de contexto, definida por um comprimento padrão de 128 bases nitrogenadas (posição $i = 0$ até $i = 127$), serve como a unidade fundamental para a extração de características. Para cada subsequência de contexto, as características são extraídas e utilizadas para prever o símbolo que representa a base subsequente, designada como rótulo ou classe (por exemplo, a base na posição $i = 128$). Após a extração de características de uma dada subsequência, o processo avança uma base ($i = 1$ até $i = 128$), repetindo a extração de características para a nova subsequência de contexto e identificando a base seguinte ($i = 129$) como o novo rótulo. Este procedimento é aplicado de forma iterativa ao longo da sequência genômica até que se atinja seu término. Tal abordagem assegura que cada segmento da sequência genômica seja analisado, permitindo que o modelo preditivo aprenda a partir de um espectro amplo de contextos genômicos, o que é fundamental para a acurácia da previsão.

A listagem subsequente fornece um detalhamento das características avaliadas, especificando a dimensão alocada a cada uma no vetor de características:

- **Codificação One-Hot:** Dimensão = 4
- **Estatísticas:** Dimensão = 5
- **Frequências:** Dimensão = 1
- **Conteúdo GC:** Dimensão = 1
- **Frequência de *k-mers*:** Dimensão = kmer_loops
- **Entropia de Shannon:** Dimensão = 1
- **Transformada de Fourier:** Dimensão = 1
- **Cálculo de Skew:** Dimensão = 1

- **Palíndromos de *k-mer*:** Dimensão = palindrome_loops
- **Códons de Parada:** Dimensão = 1

Este conjunto de características foi selecionado para permitir uma análise abrangente e detalhada das sequências genômicas, embora, como mencionado anteriormente, a abordagem simplificada baseada unicamente na codificação *one-hot* tenha sido a escolhida para a modelagem final.

Codificação One-Hot A codificação *one-hot* é uma técnica essencial na representação de sequências genômicas para análises computacionais. Nesta abordagem, cada base nitrogenada da sequência genômica é representada por um vetor onde apenas um elemento é 'verdadeiro' (representado por 1) e todos os demais são 'falsos' (representados por 0). Esta representação garante que as bases nitrogenadas sejam tratadas de forma equitativa e sem qualquer referência de ordem ordinal ou hierárquica.

Na implementação específica adotada, a subsequência genômica é primeiramente selecionada. Cada base da sequência é então mapeada para um vetor correspondente, com dimensões definidas pelo comprimento da sequência (l) e pelo tamanho do alfabeto de bases (tamanho_do_alfabeto). Cada vetor tem um único elemento definido como 'verdadeiro', correspondente à posição da base no alfabeto.

Seja S uma sequência genômica de comprimento l e Alfabeto o conjunto de bases nitrogenadas, onde tamanho_do_alfabeto = |Alfabeto|. A codificação *one-hot* de S é uma matriz E de dimensões $l \times$ tamanho_do_alfabeto, onde:

$$E_{i,j} = \begin{cases} 1, & \text{se a } i\text{-ésima base de } S \text{ corresponde à } j\text{-ésima base em Alfabeto} \\ 0, & \text{caso contrário} \end{cases}$$

para $i = 1, 2, \ldots, l$ e $j = 1, 2, \ldots,$ tamanho_do_alfabeto.

Estatísticas da Subsequência Genômica A análise estatística de sequências genômicas fornece percepções sobre a composição e a variação das bases nitrogenadas. Nesta pesquisa, foi implementado um método para calcular estatísticas comuns, incluindo a média, mediana, desvio padrão, valor mínimo e valor máximo do número de ocorrências de cada base nitrogenada do alfabeto genômico

(A, C, G, T) na subsequência. Estas medidas estatísticas oferecem uma visão geral da distribuição das bases ao longo da subsequência, contribuindo para uma compreensão mais aprofundada das características genômicas.

A média é calculada como a média aritmética do número de ocorrências de cada base, enquanto a mediana é o valor central quando as ocorrências são ordenadas. O desvio padrão fornece uma medida da variação ou dispersão das ocorrências das bases. Os valores mínimo e máximo indicam, respectivamente, a menor e a maior ocorrência de qualquer base na sequência.

Seja S uma sequência genômica e cX o número de ocorrências da base X em S, onde X pertence ao conjunto de bases nitrogenadas $\{A, C, G, T\}$. As estatísticas da sequência são calculadas como segue:

- Média (μ): $\mu = \frac{cA+cC+cG+cT}{4}$
- Mediana ($\tilde{x}$): $\tilde{x}$ = mediana($[cA, cC, cG, cT]$)
- Desvio Padrão (σ): σ = desvio_padrão($[cA, cC, cG, cT]$)
- Mínimo (min): min = mínimo(cA, cC, cG, cT)
- Máximo (max): max = máximo(cA, cC, cG, cT)

As estatísticas calculadas são então replicadas para formar uma matriz de dimensões equivalentes ao comprimento da sequência (l), resultando em uma representação uniforme para toda a sequência.

Frequências das Bases Nitrogenadas A frequência das bases nitrogenadas em uma sequência genômica é uma métrica importante que reflete a prevalência de cada base ao longo da sequência. Nesta etapa da pesquisa, foi desenvolvido um método para calcular a frequência de cada símbolo (base nitrogenada) na sequência. Este cálculo dá um entendimento detalhado da composição genômica, sendo crucial para análises subsequentes. Para cada posição na sequência, a frequência da base correspondente é determinada e incluída no vetor de características.

Seja S uma sequência genômica de comprimento l, e f_X a frequência da base X em S, onde X pertence ao conjunto de bases nitrogenadas $\{A, C, G, T\}$. Para

cada posição i na sequência S, a frequência da base S_i é calculada. O vetor de frequências para a sequência é então definido como:

$$\text{Freq}(S) = [f_{S_1}, f_{S_2}, \ldots, f_{S_l}]$$

Onde f_{S_i} é a frequência da base na posição i da sequência. Este vetor de frequências é utilizado para representar a distribuição das bases ao longo da sequência genômica.

Conteúdo de GC O conteúdo de GC, definido como a proporção de guanina (G) e citosina (C) em uma sequência genômica, é um indicador na bioinformática. Este parâmetro é frequentemente associado a diversos aspectos biológicos, incluindo a estabilidade do DNA, pontos de fusão e características evolutivas. Na pesquisa, foi desenvolvido um método para calcular a proporção de GC em cada sequência genômica. Este cálculo fornece uma visão quantitativa da composição de GC, refletindo características importantes da sequência analisada.

Seja S uma sequência genômica de comprimento l, e cX o número de ocorrências da base X em S, onde X pertence ao conjunto de bases nitrogenadas $\{A, C, G, T\}$. O conteúdo de GC é calculado como a proporção total de guanina (G) e citosina (C) na sequência, definido por:

$$\text{Conteúdo\_GC} = \frac{cG + cC}{l}$$

Onde cG e cC são, respectivamente, o número de ocorrências de guanina e citosina na sequência. Este valor é replicado para formar um vetor de dimensões equivalentes ao comprimento da sequência (l), fornecendo uma representação constante do conteúdo de GC para toda a sequência.

Frequência de *k-mers* A frequência de *k-mers*, que são subsequências de comprimento k em uma sequência genômica, é uma métrica importante na bioinformática. Esta análise ajuda a identificar padrões repetitivos e características estruturais dentro da sequência. Na pesquisa, foi desenvolvido um método para calcular a frequência de cada *k-mer* possível ao longo da sequência genômica. Este processo permite a identificação de padrões frequentes e raros, que podem ser indicativos de regiões funcionais ou estruturais específicas na sequência.

Seja S uma sequência genômica de comprimento l e k o comprimento do *k-mer* a ser analisado. Para cada posição i em S, com $i \leq l - k$, um *k-mer* é

definido como a subsequência $S_{i:i+k}$. O número de ocorrências de cada *k-mer* é calculado e armazenado em um dicionário *kmer_counts*. A frequência de um *k-mer* em uma posição específica i é então dada por:

$$\text{Freq\_kmer}_i = \frac{\text{kmer\_counts}[S_{i:i+k}]}{l - k + 1}$$

Este processo resulta em um vetor que representa a frequência de cada *k-mer* ao longo da sequência genômica, oferecendo uma visão detalhada da distribuição e prevalência de padrões específicos de subsequências.

Entropia de *Shannon* A entropia de *Shannon* é uma métrica derivada da teoria da informação, usada para quantificar a incerteza ou a diversidade em sequências genômicas (SHANNON, 1948). Esta métrica é particularmente útil para avaliar a variabilidade na composição de bases da sequência, oferecendo *insights* sobre a complexidade genética. Na pesquisa, calculou-se a entropia de *Shannon* para a sequência genômica, considerando a frequência de cada base. Este cálculo fornece uma medida da diversidade informacional da sequência, sendo um indicador da sua heterogeneidade.

Seja S uma sequência genômica de comprimento l. Para cada base b em S, c_b denota o número de ocorrências da base. A probabilidade p_b de uma base b é então calculada como $p_b = \frac{c_b}{l}$. A entropia de Shannon $H(S)$ da sequência é dada por:

$$H(S) = -\sum_{b \in \{A,C,G,T\}} p_b \log_2(p_b)$$

Onde a soma é sobre todas as bases distintas na sequência. A entropia calculada é então replicada ao longo do vetor de características, proporcionando uma representação constante da diversidade informacional para toda a sequência.

Transformada de *Fourier* A transformada de *Fourier* é uma técnica matemática amplamente utilizada para decompor funções ou sinais em frequências. Na bioinformática, sua aplicação em sequências genômicas facilita a identificação de padrões periódicos e características estruturais. Na implementação desta pesquisa, a sequência genômica foi primeiro convertida em uma sequência numérica correspondente. Posteriormente, aplicou-se a Transformada de *Fourier* Discreta (DFT) para analisar as componentes de frequência da sequência. O

espectro de frequência obtido por meio da DFT revela informações sobre a periodicidade e as características frequenciais da sequência genômica.

Seja S uma sequência genômica convertida em uma sequência numérica S_{num} de comprimento l. A Transformada de Fourier Discreta (DFT) de S_{num} é dada por:

$$DFT(S_{\text{num}}) = \text{abs}(FFT(S_{\text{num}}))$$

Onde FFT representa o algoritmo de Transformada de *Fourier* Rápida. O espectro de frequência é obtido considerando a primeira metade do resultado da DFT, representado por Espectro = $DFT(S_{\text{num}})_{:l//2+1}$. O vetor de características final combina o espectro de frequência com as últimas posições da sequência numérica, proporcionando uma visão abrangente das características de frequência da sequência genômica.

Cálculo de *Skew* de Guanina-Citosina O *skew* de guanina-citosina é uma medida importante na genômica, utilizada para analisar a variação na composição de guanina (G) e citosina (C) ao longo de uma sequência genômica. Este indicador é calculado ao percorrer a sequência e ajustar um contador de *skew* baseado na presença de bases G e C. O *skew* aumenta com cada guanina e diminui com cada citosina encontrada, refletindo assim o desequilíbrio entre essas duas bases ao longo da sequência. Este método é particularmente útil para identificar regiões genômicas com características composicionais distintas.

Seja S uma sequência genômica de comprimento l. Define-se o *skew* de guanina-citosina em uma posição i da sequência, $\text{Skew}_i(S)$, como:

$$\text{Skew}_i(S) = \sum_{j=1}^{i} (1_{S_j=G} - 1_{S_j=C})$$

Onde $1_{S_j=X}$ é a função indicadora que retorna 1 se a base na posição j é X e 0 caso contrário. O cálculo começa com $\text{Skew}_0(S) = 0$ e itera sobre cada base da sequência, aumentando o *skew* para guanina (G) e diminuindo para citosina (C). O resultado é uma série que representa o *skew* acumulado em cada posição da sequência.

Anotação de Palíndromos Palíndromos genômicos são sequências de DNA que são lidas da mesma forma tanto na direção 5′ para 3′ quanto na direção

3′ para 5′. A identificação de palíndromos em sequências genômicas é crucial, pois essas estruturas podem desempenhar papéis significativos em processos biológicos, como a formação de estruturas secundárias do DNA. Na pesquisa, foi desenvolvido um método para anotar a presença de palíndromos de um determinado comprimento k na sequência genômica. Essa anotação ajuda a identificar regiões da sequência que possuem simetria palindrômica, o que pode indicar áreas de interesse biológico.

Seja S uma sequência genômica de comprimento l e k o comprimento do palíndromo a ser identificado. Para cada posição i em S, com $i \leq l - k$, um palíndromo é detectado se a subsequência $S_{i:i+k}$ é igual à sua sequência inversa. A anotação de palíndromos é realizada por meio de uma série A, onde:

$$A_i = \begin{cases} 1, & \text{se } S_{i:i+k} = (S_{i:i+k})^{\text{rev}} \\ 0, & \text{caso contrário} \end{cases}$$

para $i = 1, 2, \ldots, l$. Aqui, $(S_{i:i+k})^{\text{rev}}$ representa a subsequência $S_{i:i+k}$ invertida. Esta série A indica a presença de palíndromos ao longo da sequência genômica.

Identificação de Códons de Parada Códons de parada são trincas específicas de bases nitrogenadas no DNA que sinalizam o término da tradução de proteínas. A identificação desses códons é fundamental na análise genômica, pois eles marcam os limites funcionais das proteínas codificadas. Na pesquisa, desenvolveu-se um método para anotar a presença dos códons de parada "TAA", "TAG"e "TGA"na sequência genômica. Esta anotação é essencial para compreender a estrutura e a função dos genes, indicando as regiões onde a síntese de proteínas é finalizada.

Seja S uma sequência genômica de comprimento l. Os códons de parada considerados são "TAA", "TAG"e "TGA". Para cada posição i em S, com $i \leq l-2$, verifica-se se a subsequência de três bases $S_{i:i+3}$ corresponde a um códon de parada. A anotação dos códons de parada é realizada por mio de uma série C, onde:

$$C_i = \begin{cases} 1, & \text{se } S_{i:i+3} \text{ é um códon de parada} \\ 0, & \text{caso contrário} \end{cases}$$

para $i = 1, 2, \ldots, l$. Esta série C indica a presença de códons de parada ao longo da sequência genômica.

Métodos Automatizados de Seleção de características

Embora existam métodos automatizados para a extração e seleção de características, como a Eliminação Recursiva de Características (*Recursive Feature Elimination* - RFE) (YAN; ZHANG, 2015) e a Análise de Componentes Principais (*Principal Component Analysis* - PCA) (JOLLIFFE, 2002), optou-se por não empregar tais abordagens neste estudo. Essa decisão fundamenta-se no fato de que o conjunto de características considerado é relativamente reduzido, consistindo em dez características principais, conforme detalhado anteriormente, com dimensões variando de 1 até *kmer_loops* e *palindrome_loops*. Esse escopo permitiu a avaliação manual do impacto individual de cada característica no desempenho preditivo do modelo.

Ao testar cada característica isoladamente em comparação com a codificação *one-hot*, foi possível compreender diretamente a influência de cada uma na acurácia do modelo, sem a necessidade de algoritmos automatizados. Além disso, a simplicidade e o tamanho do conjunto de características não exigiram a aplicação de técnicas de redução dimensional ou seleção automática para prevenir sobreajuste ou atender a restrições computacionais. Portanto, a abordagem manual adotada revelou-se suficiente e adequada aos objetivos do estudo, permitindo uma análise controlada das características sem a introdução de complexidade adicional.

Padronização e Normalização dos Dados

Após a extração das características genômicas, procedeu-se com a padronização e a normalização dos dados. A padronização, ou *Z-score normalization*, transforma os dados de modo que sua distribuição tenha uma média de zero e um desvio padrão de um. Este processo é crucial para algoritmos de aprendizado de máquina que assumem que todos os atributos estão centrados em torno de zero e possuem variação na mesma escala. Seguidamente, aplicou-se a normalização *Min-Max*, que redimensiona os dados para um intervalo específico, geralmente entre zero e um. Esta técnica é útil para algoritmos sensíveis a variações na escala dos dados e ajuda a preservar as relações entre os valores originais, mantendo a estrutura dos dados.

Para a padronização, os dados são transformados seguindo a fórmula:

$$X_{\text{padronizado}} = \frac{X - \mu}{\sigma}$$

onde μ é a média e σ é o desvio padrão dos dados.

Para a normalização *Min-Max*, os dados são redimensionados conforme:

$$X_{\text{normalizado}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}}$$

onde X_{min} e X_{max} são, respectivamente, os valores mínimo e máximo dos dados.

Estes processos resultam em um conjunto de dados com características ajustadas, apto para análises preditivas e modelagem estatística.

Avaliação comparativa da acurácia entre características extraídas

Na Figura A.1, é apresentada a avaliação comparativa da acurácia do modelo com as camadas *CNN+LSTM* utilizando diferentes variantes da Codificação *One-Hot* (COH). A comparação é feita entre a aplicação da Codificação *One-Hot* de forma isolada e sua combinação com diversas características adicionais. O modelo base, *CNN+LSTM*, permanece constante em todas as variantes testadas, assegurando a consistência na comparação.

Os resultados mostram que a variante puramente baseada na Codificação *One-Hot* alcançou a maior acurácia 99, 65%, indicando que, neste contexto específico, a adição de características adicionais não resultou em melhorias significativas de desempenho. As variantes que combinaram a Codificação *One-Hot* com características como Entropia de *Shannon*, Códons de Parada, Transformada de *Fourier* e Estatísticas, embora tenham apresentado acurácias elevadas, não superaram a Codificação *One-Hot* em sua forma isolada. Estes resultados sugerem que a simplicidade da Codificação *One-Hot*, quando aplicada no contexto do modelo *CNN+LSTM*, é suficiente para alcançar um melhor desempenho, e a introdução de complexidade adicional pode não ser necessária ou até mesmo contraproducente.

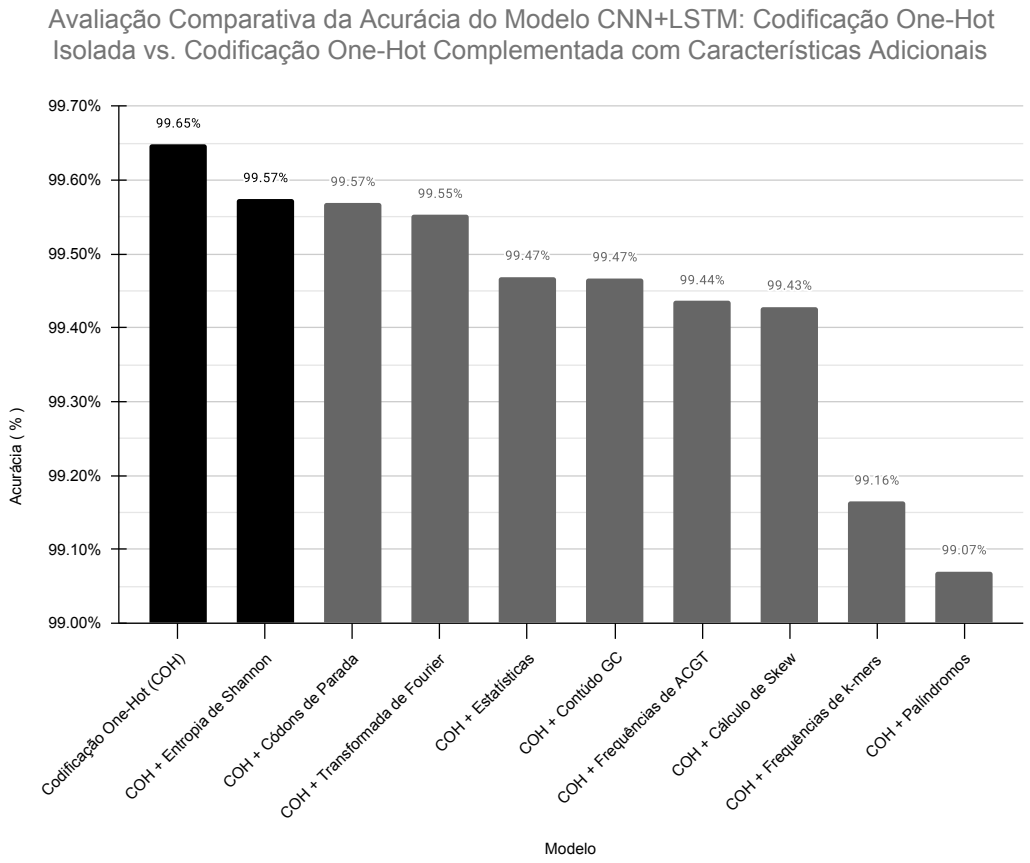

Figura A.1: Avaliação Comparativa da Acurácia do Modelo com as redes *CNN+LSTM*: Codificação *One-Hot* (COH) Isolada vs. Codificação *One-Hot* (COH) Complementada com Características Adicionais. A figura ilustra a acurácia alcançada pelo modelo *CNN+LSTM* utilizando a *COH* isolada em comparação com variações que incluem características adicionais - o autor.

B

Anexo B

Este anexo é dedicado à explanação detalhada de uma das técnicas de transformação dos dados utilizada em alguns experimentos. Especificamente, essa técnica aplica a codificação *Run-Length Encoding* (RLE) sobre a *bitstring* de predição. Inicialmente, apresenta-se uma visão geral da técnica RLE para dados binários, ressaltando sua aplicabilidade e vantagens no contexto de compressão de dados. A discussão subsequente detalha o procedimento de transformação, ilustrando como sequências de *bits* repetitivos podem ser convertidas em representações mais curtas, empregando uma máscara de *bits* de comprimento fixo baseado em um valor de expoente. Este método é particularmente vantajoso quando confrontado com cadeias de *bits* de extenso comprimento repetitivo, onde a aplicação da técnica RLE resulta em uma redução substancial no número de *bits* necessários para representar a informação original.

B.1 Transformação dos dados - Run-Length Encoding

Após a fase de predição em que é gerada a *bitstring* de predição, o passo seguinte é transformar cada cadeia de *bits* repetitivos, para diminuir a entropia, usando a técnica RLE para binário (cf. Figura B.1 e (cf. Algoritmo 2) (GOLOMB, 1966). Assim, quanto maior for o comprimento da cadeia repetitiva de *bits*, mais vantajosa é a transformação com a técnica RLE. Para isso, usa-se um marcador do *bit* ou cadeia de *bits* que está sendo tratado, seguido da quantidade de *bits*

que representa o expoente (cf. Algoritmo 1). Por exemplo, seja a base $b = 2$, o expoente $e = 5$, o comprimento (*run-length*) $c = 32$, e a sequência binária s de comprimento 67 é transformada da seguinte forma:

$s = 01111111111111111111111111111111100000000000000000000000000000000$

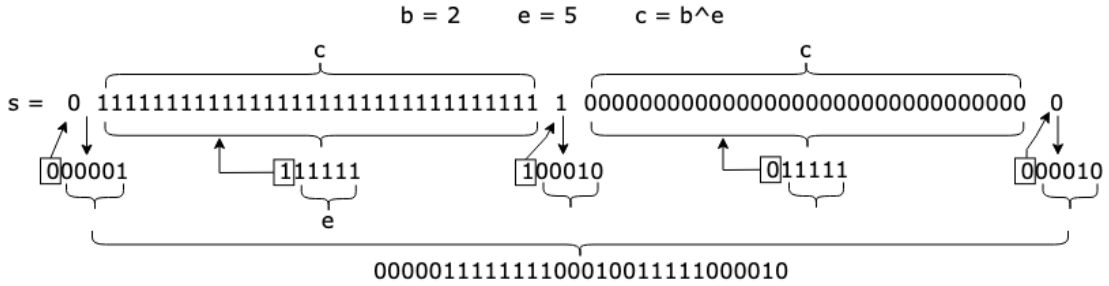

Figura B.1: Processo de transformação com *Run-Length Encoding* - o autor.

Como resultado da codificação na figura B.1 a *string* binária foi reduzida de 67 *bits* para 30 *bits*. Sempre que há uma cadeia repetitiva de *bits* de comprimento (*run-length*) definido, transforma-se para uma representação binária de comprimento menor, ou seja, valor da potência somado a 1 do marcador será a quantidade de *bits* ($e + 1 = 6$). Em contrapartida, a mesma representação é utilizada para cada *bit* nas posições em que não há cadeias repetitivas com comprimento igual ao comprimento (*run-length*). Por exemplo, se a cadeia repetitiva fosse de tamanho 31 e o comprimento (*run-length*) de tamanho 32, cada *bit* desta porção da cadeia seria transformado para 6 *(e+1) bits*. Assim, quanto mais preciso for o modelo de predição, maiores serão as porções repetitivas na cadeia de *bits*, e em consequência disso, maiores expoentes poderão ser utilizados para a codificação RLE.

B.2 Transformação dos dados - Arithmetic Coding

A codificação aritmética é uma técnica de compressão de dados que, ao contrário de métodos baseados em símbolos individuais, como *Huffman Coding*, opera sobre toda a sequência de dados como uma entidade única. Esta abordagem permite alcançar níveis de compressão próximos ao limite teórico da entropia de uma fonte de dados, representando a sequência de dados por meio de um único número decimal situado entre 0 e 1. A eficácia da codificação aritmética reside na sua capacidade de lidar com as probabilidades de ocorrência

de cada símbolo dentro da sequência, ajustando dinamicamente o intervalo de codificação com base na probabilidade acumulada dos símbolos processados.

Na aplicação da codificação aritmética à *bitstring* de predição, inicia-se com o intervalo $[0, 1)$ e divide-se este intervalo de maneira proporcional às probabilidades de ocorrência de 0 ou 1 na sequência. À medida que cada *bit* é processado, o intervalo é sucessivamente particionado, refletindo a probabilidade cumulativa dos *bits* até o momento. Assim, *bits* com maior probabilidade de ocorrência conduzem a um ajuste menos significativo do intervalo, enquanto aqueles com menor probabilidade resultam em ajustes mais substanciais. Este processo contínuo de partição do intervalo, à medida que cada *bit* é lido, converge para um número decimal específico que representa de forma única a sequência original.

Suponha-se que a *bitstring* 1111101011 represente as predições de próximos símbolos da sequência de bases nucleotídicas ATCGAGAA, de comprimento 8. Na posição de índice 5, o preditor falhou ao prever a base G, anotando, em vez disso, um marcador de erro de predição representado pelo *bit* 0 seguido de dois *bits* 10 para representação literal das bases, onde A=00, C=01, G=10, e T=11.

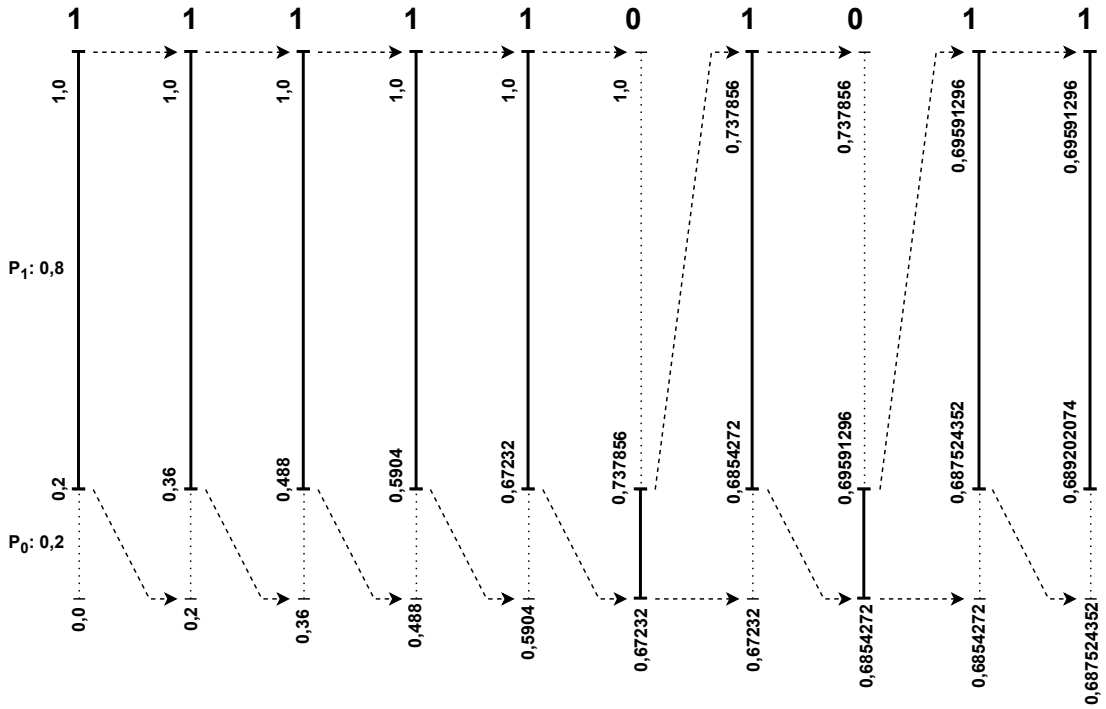

Figura B.2: Representação esquemática do processo de codificação aritmética aplicado à *bitstring* 1111101011, destacando a sequência de ajustes de intervalos com base nas probabilidades de ocorrência de 0 e 1 - o autor.

O processo de codificação aritmética para a *bitstring* de predição começa com o cálculo das probabilidades dos símbolos 0 e 1 ocorrerem na *bitstring* de predição. No caso da sequência 1111101011 de comprimento 10, o *bit* 0 pode aparecer duas vezes, enquanto o *bit* 1 pode ocorrer oito vezes. Assim, os *bits* 0 e 1 possuem as probabilidades de 20% e 80% de ocorrência, respectivamente. Após a determinação das probabilidades, o processo de codificação aritmética avança com a divisão do intervalo inicial $[0, 1]$ em subintervalos que refletem essas probabilidades (cf. Figura B.2). Portanto, cada *bit* subsequente da *bitstring* 1111101011 é processado, ajustando dinamicamente o intervalo de acordo com as probabilidades pré-estabelecidas. Para o primeiro *bit* 1, o intervalo é ajustado para $[0.2, 1.0]$, representando a probabilidade acumulada de 80% para o *bit* 1. Com a adição de cada novo *bit* 1, o intervalo é sucessivamente refinado, estreitando-se para refletir a sequência específica de *bits* sendo codificada.

Ao final do processo de alocação dos intervalos para a *bitstring* 1111101011, conforme ilustrado na Figura B.2, o intervalo correspondente ao último *bit* (1) está definido como $[0.689202074, 0.69591296]$. Para determinar o número dentro deste intervalo que possa ser representado com a menor quantidade de *bits*, é necessário primeiramente calcular o tamanho deste intervalo final. Esse cálculo é realizado pela diferença entre o limite superior e o limite inferior do intervalo, resultando em $0.69591296 - 0.689202074 = 0.006710886$, que define a extensão do intervalo final.

A partir da obtenção do valor da extensão do intervalo final, 0.006710886, o próximo passo é calcular o número mínimo de *bits* b necessário para representar um valor dentro deste intervalo específico. A quantidade de *bits* requerida pode ser aproximada pela fórmula:

$$b = \lceil \log_2(\frac{1}{tamanho\ do\ intervalo}) \rceil$$

onde $\lceil \cdot \rceil$ indica o arredondamento para o maior inteiro mais próximo. Este cálculo utiliza o inverso do tamanho do intervalo para estimar a quantidade de informação, em *bits*, necessária para representar um número que caiba unicamente dentro do intervalo especificado, permitindo assim a recuperação exata da *bitstring* original durante o processo de descompressão.

O tamanho final em *bits* da representação codificada da *bitstring* 1111101011 por meio da utilização da codificação aritmética é de aproximadamente 8 *bits*. Contrastando com o método de codificação *naive-bit*, o qual requer dois *bits* por

base para armazenar a sequência ATCGAGAA, seriam necessários $2 \times 8 = 16$ *bits*. Deste modo, a aplicação da codificação aritmética à *bitstring* de predição 1111101011 resulta em uma significativa redução do tamanho em *bits* da representação codificada, com uma economia de espaço de 50%. Quando comparado ao armazenamento direto em ASCII, onde cada símbolo requer 8 *bits*, para representar a sequência de oito caracteres ATCGAGAA, seriam necessários $8 \times 8 = 64$ *bits*. Assim, ao empregar a codificação aritmética, que comprime a sequência para aproximadamente 8 *bits*, a Economia de Espaço em relação ao armazenamento em ASCII é calculada como $\left(1 - \frac{8}{64}\right) \times 100 = 87.5\%$. Portanto, a utilização da codificação aritmética não apenas proporciona uma redução considerável do espaço necessário para armazenar a sequência em comparação com o método *naive-bit*, mas também oferece uma economia de espaço de 87.5% em relação ao armazenamento direto em ASCII.

Anexo C

O presente anexo destina-se a expor os algorítimos utilizados nos experimentos na fase de transformação dos dados. Assim, três técnicas distintas foram abordadas, cada uma com um objetivo espcífico para tranformar os dados na menor representação possível. As técnicas de transformação dos dados empregadas nesse trabalho foram: *Run-Length Encoding (RLE)*, *Arithmetic Encoding (ARE)*, e *Bitstring as Codebook (BAC)*.

A seção sobre Transformação dos dados - *Run-Length Encoding (RLE) binary-based* descreve os algoritmos dessa técnica que é particularmente eficaz em sequências com repetições consecutivas de um mesmo valor (ELIAS, 1975). A partir da *bitstring* de predição, a técnica RLE substitui as sequências repetitivas de comprimento *Run-Length* por uma sequência binária de tamanho fixo, reduzindo assim o espaço necessário para armazenar essas sequências.

Na subsequente seção, Transformação dos dados - *Arithmetic Encoding (ARE)* (RISSANEN, 1976) é apresentado os algoritmos da abordagem de codificação que, ao contrário do RLE, utiliza as probabilidades de ocorrência de cada símbolo (*bit*) da *bitstring* de predição para gerar um intervalo numérico único que representa toda a sequência. Este método permite uma compressão altamente eficiente, adaptando-se dinamicamente às características dos dados e alcançando taxas de compressão próximas ao limite teórico de *Shannon* para a entropia dos dados (SHANNON, 1948).

Por fim, a seção Transformação dos dados - *Bitstring as Codebook (BAC)* apresenta os algoritmos de uma abordagem que armazena, em formato binário, as posições deltas dos erros anotados na *bitstring* de predição. Esta técnica, embora

possa ser mais simples em sua implementação, mostrou-se eficaz nos testes de desempenho.

Os pseudocódigos apresentados nos algoritmos subsequentes são baseados na linguagem de programação *Python*, escolhida devido à sua ampla utilização no cenário atual de desenvolvimento de software (PYTHON, 2024).

C.1 Transformação dos dados - Run-Length Encoding (RLE) binary-based

Algoritmo 1 Cálculo do Expoente para Codificação *RLE*.

Require: Sequência binária $sequence$ com comprimento n
Ensure: Expoente $best\_exponent$

Inicialize $sequence\_lengths$ para armazenar comprimentos das sequências de '0' e '1'
$last\_bit \leftarrow$ primeiro bit de $sequence$
$count \leftarrow 1$
for $i \leftarrow 2$ **to** n **do**
    if o bit na posição i da $sequence$ é igual a $last\_bit$ **then**
        Incrementa $count$
    else
        Adiciona $count$ ao conjunto de comprimentos para $last\_bit$
        $last\_bit \leftarrow$ bit na posição i da $sequence$
        $count \leftarrow 1$
    end if
end for
Adiciona o último valor de $count$ ao conjunto de comprimentos para $last\_bit$
$best\_compression \leftarrow \infty$
$max\_exponent \leftarrow$ comprimento do maior número em $sequence\_lengths$ em bits
for $exponent \leftarrow 1$ **to** $max\_exponent$ **do**
    $compression \leftarrow 0$
    for all comprimentos $length$ em $sequence\_lengths$ **do**
        $chunks \leftarrow$ divisão inteira de $length$ por $2^{exponent}$
        $remainder \leftarrow length \mod 2^{exponent}$
        $compression \leftarrow compression + chunks \times (1 + exponent)$
        if $remainder > 1$ **then**
            $compression \leftarrow compression + 1 + exponent$
        end if
    end for
    if $compression < best\_compression$ **then**
        $best\_compression \leftarrow compression$
        $best\_exponent \leftarrow exponent$
    end if
end for
return $best\_exponent$

Algoritmo 2 Codificação *RLE*

Require: Sequência binária $bitstring$ com comprimento n

Ensure: Sequência binária codificada $encoded\_bitstring$ e expoente $exponent$

$exponent \leftarrow$ calculate_exponent($bitstring$)
$cont \leftarrow 0$
$last\_symbol \leftarrow$ primeiro bit de $bitstring$
$encoded\_bitstring \leftarrow$ string vazia
for $i \leftarrow 2$ **to** n **do**
    if $last\_symbol \neq bitstring[i]$ **or** $cont = 2^{exponent}$ **then**
        Adicione $last\_symbol$+Bin($cont-1, exponent$) a $encoded\_bitstring$
        $cont \leftarrow 1$
    end if
    Incremente $cont$
    $last\_symbol \leftarrow bitstring[i]$
end for
Adicione $last\_symbol$ + Bin($cont - 1, exponent$) a $encoded\_bitstring$
if $encoded\_bitstring[1]$ é '-' **then**
    Remova '-' de $encoded\_bitstring$ e preponha '1'
else
    Preponha '0' a $encoded\_bitstring$
end if
$prefix \leftarrow$' 00' + $bitstring[0]$
if $bitstring[0] = bitstring[1]$ **then**
    $prefix \leftarrow$' 1' + $bitstring[0]$ + $bitstring[1]$
end if
return $prefix + encoded\_bitstring, exponent$

Algoritmo 3 Decodificação *RLE*

Require: Sequência binária codificada $encoded\_bitstring$ e expoente $exponent$
Ensure: Sequência binária decodificada $decoded\_bitstring$

$prefix\_tmp \leftarrow$ primeiros 3 bits de $encoded\_bitstring$
$encoded\_bitstring \leftarrow$ remove os primeiros 3 bits de $encoded\_bitstring$
if primeiro bit de $encoded\_bitstring$ é '1' **then**
    Insira um '-' após o primeiro bit de $encoded\_bitstring$
    Remove o primeiro bit de $encoded\_bitstring$
else
    Remove o primeiro bit de $encoded\_bitstring$
end if
$max\_pow \leftarrow exponent + 1$
$decoded\_bitstring \leftarrow$ string vazia
Inicialize $sequence\_split$ como lista vazia
for $i \leftarrow 0$ **to** length of $encoded\_bitstring$ **step** max_pow **do**
    Adiciona a subsequence de $encoded\_bitstring$ começando em i com comprimento max_pow a $sequence\_split$
end for
for all $item$ em $sequence\_split$ **do**
    if $item$ tem comprimento maior que 1 **then**
        $decoded\_bitstring \leftarrow decoded\_bitstring$+ primeiro bit de $item$ repetido int($item[1:], 2$) vezes
    end if
end for
$prefix \leftarrow$ terceiro bit de $prefix\_tmp$
if primeiro bit de $prefix\_tmp$ é '1' **then**
    $prefix \leftarrow$ últimos 2 bits de $prefix\_tmp$
end if
return $prefix + decoded\_bitstring$

C.2 Transformação dos dados - Arithmetic Encoding (ARE)

Algoritmo 4 Calcula Frequências Acumuladas

Require: Um dicionário $freq$ com as frequências dos símbolos

Ensure: Um dicionário $frequencies$ com as frequências acumuladas

```
1: function CumulativeFrequencies(freq)
2:     frequencies ← {}
3:     total ← 0
4:     for b em sorted(freq.keys()) do
5:         frequencies[b] ← total
6:         total ← total + freq[b]
7:     end for
8:     return frequencies
9: end function
```

Algoritmo 5 Codificador Aritmético

Require: Uma sequência de bytes $sequence\_bytes$, Um dicionário $frequencies$ com as frequências dos símbolos

Ensure: Um dado codificado $encoded\_data$ e um fator de escala $scaling\_factor$

function ArithmeticEncoder($sequence\_bytes, frequencies$)
    $accumulated\_freq \leftarrow$ CumulativeFreq($frequencies$)
    $base \leftarrow sum(frequencies.values())$
    $lower \leftarrow 0$
    $frequency\_product \leftarrow 1$
    for $byte$ **em** $sequence\_bytes$ **do**
        $lower \leftarrow lower \times base + accumulated\_freq[byte] \times frequency\_product$
        $frequency\_product \leftarrow frequency\_product \times frequencies[byte]$
    end for
    $upper \leftarrow lower + frequency\_product$
    $scaling\_factor \leftarrow 0$
    while $frequency\_product \geq radix$ **do**
        $frequency\_product \leftarrow frequency\_product // radix$
        $scaling\_factor \leftarrow scaling\_factor + 1$
    end while
    $encoded\_data \leftarrow (upper - 1) // (radix^{scaling\_factor})$
    return $encoded\_data, scaling\_factor$
end function

Algoritmo 6 Preparador da Bitstring

Require: Uma bitstring $bitstring$, Um alfabeto $alphabet$

Ensure: Uma sequência codificada $encoded\_seq$, Um fator de escala $scaling\_factor$, Frequências $frequencies$

1: **function** BitstringPrepare($bitstring, alphabet$)
2: $char\_to\_int \leftarrow \{char : i$ **para** $i, char$ **em** $enumerate(set(alphabet))\}$
3: $int\_sequence \leftarrow [char\_to\_int[char]$ **para** $char$ **em** $bitstring]$
4: $bytes\_sequence \leftarrow bytes(int\_sequence)$
5: $frequencies \leftarrow dict(Counter(bytes\_sequence))$
6: $encoded\_seq, scaling\_factor \leftarrow$ Encoder($bytes\_sequence, frequencies$)
7: **return** $encoded\_seq, scaling\_factor, frequencies$
8: **end function**

Algoritmo 7 Decodificador Aritmético

Require: Um dado codificado enc, um fator de escala $scaling\_factor$, um dicionário $frequencies$ com as frequências dos símbolos

Ensure: Uma sequência de bytes decodificada

function ArithmeticDecoder($enc, scaling\_factor, frequencies$)
    $enc \leftarrow enc \times radix^{scaling\_factor}$
    $base \leftarrow sum(frequencies.values())$
    $cf \leftarrow$ CumulativeFreq($frequencies$)
    $dict\_from\_cf \leftarrow \{v : k$ para cada k, v em $cf.items()\}$
    $last\_char \leftarrow$ nulo
    for i de 0 até $base$ **do**
        if i em $dict\_from\_cf$ **then**
            $last\_char \leftarrow dict\_from\_cf[i]$
        else if $last\_char$ não é nulo **then**
            $dict\_from\_cf[i] \leftarrow last\_char$
        end if
    end for
    $decoded \leftarrow$ bytearray vazio
    for i de $base - 1$ até 0 **do**
        $pow\_base \leftarrow base^i$
        $quotient \leftarrow enc // pow\_base$
        $char \leftarrow dict\_from\_cf[quotient]$
        $freq\_val \leftarrow frequencies[char]$
        $cum\_freq\_val \leftarrow cf[char]$
        $remainder \leftarrow (enc - pow\_base \times cum\_freq\_val) // freq\_val$
        $enc \leftarrow remainder$
        Adiciona $char$ ao final de $decoded$
    end for
    return $bytes(decoded)$
end function

Algoritmo 8 Restaurador da Bitstring

Require: Uma sequência codificada $encoded\_seq$, um fator de escala $scaling\_factor$. Também são necessários um dicionário $frequencies$ com as frequências dos símbolos e um alfabeto $alphabet$.

Ensure: Uma bitstring decodificada

1: **function** BitstringRestore($encoded\_seq$, $scaling\_factor$, $frequencies$, $alphabet$)
2: $decoded\_bytes \leftarrow$ Decoder($encoded\_seq, scaling\_factor, frequencies$)
3: $decoded\_int\_sequence \leftarrow [int(byte)$ para cada $byte$ em $decoded\_bytes]$
4: $int\_to\_char \leftarrow \{i : char$ para cada $i, char$ em $enumerate(set(alphabet))\}$
5: $bitstring \leftarrow$ concatenação de $int\_to\_char[i]$ para cada i em $decoded\_int\_sequence$
6: **return** $bitstring$
7: **end function**

C.3 Transformação dos dados - Bitstring as Codebook (BAC)

Algoritmo 9 Converte Bitstring para Codebook

Require: Uma bitstring $bitstring$

Ensure: Uma representação de codebook como uma string, onde cada linha representa uma entrada do codebook

```
1:  function BitstringToCodebook(bitstring)
2:      data ← []
3:      index ← 0
4:      last_delta ← 0
5:      error_length ← 0
6:      length ← length(bitstring)
7:      while index < length do
8:          symbol ← bitstring[index]
9:          if symbol =' 1' then
10:             index ← index + 1
11:         else
12:             delta ← index − last_delta − error_length
13:             last_delta ← index
14:             error ← bitstring[index + 1 : index + 3]
15:             delta_binary ← format(delta,' b')
16:             data.append(list(str(delta_binary) + error))
17:             index ← index + 3
18:             error_length ← 3
19:         end if
20:     end while
21:     result ←
22:     for item em data do
23:         result ← result +" .join(item) + "\n"
24:     end for
25:     return result
26: end function
```

Algoritmo 10 Converte Codebook para Bitstring

Require: Um conjunto de dados de codebook $codebook\_data$ e o comprimento da bitstring original $bitstring\_length$

Ensure: Uma bitstring reconstruída a partir do codebook

1: **function** CodebookToBitstring($codebook\_data, bitstring\_length$)
2: $bitstring \leftarrow$ " "
3: **for** $entry$ em $codebook\_data$ **do**
4: **if** $isinstance(entry, list)$ **then**
5: $entry \leftarrow$" $.join(map(str, entry))$
6: $delta\_bin, error \leftarrow entry[:-2], entry[-2:]$
7: $delta \leftarrow int(delta\_bin, 2)$
8: $bitstring \leftarrow bitstring +' 1' \times delta +' 0' + error$
9: **end if**
10: **end for**
11: $bitstring \leftarrow bitstring +' 1' \times (bitstring\_length - length(bitstring))$
12: **return** $bitstring$
13: **end function**