TÉCNICAS PARA VISUALIZAÇÃO PROGRESSIVA DE SÉRIES TEMPORAIS DE DADOS MULTIVARIADOS AGREGADOS AD-HOC

Lucas Barcellos Oliveira

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Claudio Esperança

Rio de Janeiro
Agosto de 2025

TÉCNICAS PARA VISUALIZAÇÃO PROGRESSIVA DE SÉRIES TEMPORAIS DE DADOS MULTIVARIADOS AGREGADOS AD-HOC

Lucas Barcellos Oliveira

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Orientador: Claudio Esperança

Aprovada por: Prof. Claudio Esperança
Prof. Geraldo Zimbrão da Silva
Prof.ª Asla Medeiros e Sá

RIO DE JANEIRO, RJ – BRASIL
AGOSTO DE 2025

Barcellos Oliveira, Lucas

Técnicas para visualização progressiva de séries temporais de dados multivariados agregados ad-hoc/Lucas Barcellos Oliveira. – Rio de Janeiro: UFRJ/COPPE, 2025.

XIII, 94 p.: il.; 29, 7cm.

Orientador: Claudio Esperança

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2025.

Referências Bibliográficas: p. 89 – 94.

1. Visualização de dados. 2. Transmissão progressiva. 3. Séries temporais. I. Esperança, Claudio. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

A qualquer pessoa com quem
compartilho esta jornada.

Agradecimentos

Gostaria de agradecer a todos que, de alguma forma, contribuíram para a conclusão deste grande desafio. Muitos, de maneira indireta e muitas vezes sem saber, contribuíram para a realização deste trabalho. Essas contribuições sutis, mas significativas, lembram-nos de que o aprendizado e o crescimento são processos coletivos, onde cada um desempenha um papel, mesmo que pequeno, na jornada de outro.

Agradeço ao meu orientador, Claudio Esperança, por sua orientação precisa e paciente ao longo da minha trajetória acadêmica.

Obrigado a minha família e a meus amigos pelo incondicional apoio, incentivo e companhia no decorrer dessa jornada.

A dedicação e o apoio de todos foram essenciais para o desenvolvimento deste trabalho e para o meu crescimento acadêmico, profissional e pessoal.

Por fim, agradeço às instituições, como a UFRJ, que são pilares fundamentais na construção do conhecimento e referências para a sociedade. Que possamos sempre valorizar e defender essas instituições, reconhecendo seu papel vital na construção de um futuro mais justo e sustentável.

A todos, meu mais profundo e genuíno obrigado!

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

TÉCNICAS PARA VISUALIZAÇÃO PROGRESSIVA DE SÉRIES TEMPORAIS DE DADOS MULTIVARIADOS AGREGADOS AD-HOC

Lucas Barcellos Oliveira

Agosto/2025

Orientador: Claudio Esperança

Programa: Engenharia de Sistemas e Computação

Esta dissertação aborda as diversas técnicas para transmissão progressiva e carregamento progressivo de dados de séries temporais no contexto de visualização de grandes volumes de dados (*big data*). São levantadas e discutidas as principais abordagens para implementar a progressividade em ferramentas de visualização desse tipo, incluindo o uso de amostragens, transformadas e inteligência artificial. Adicionalmente, um recorte dessas técnicas é implementado e analisado segundo critérios como responsividade e acurácia a partir de conjuntos de dados reais e sintéticos.

Dado o crescente valor e uso de aplicações de *big data*, o estudo de técnicas de visualização progressiva representa um importante caminho para a concepção e o desenvolvimento de ferramentas interativas, escaláveis e capazes de oferecer informação a usuários de maneira rápida e significativa através da Internet. Assim, este trabalho explora reflexões relevantes sobre os requisitos e a viabilidade de sistemas de visualização que ofereçam uma experiência de uso fluida e eficaz, respondendo de forma veloz a perguntas feitas sobre grandes volumes de dados e suas evoluções ao longo do tempo.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

TECHNIQUES FOR PROGRESSIVE VISUALIZATION OF AD-HOC-AGGREGATED MULTIVARIATE TIME SERIES

Lucas Barcellos Oliveira

August/2025

Advisor: Claudio Esperança

Department: Systems Engineering and Computer Science

This dissertation explores the various techniques for progressive transmission and loading of time series data in the context of big data visualization. The main approaches to implementing progressiveness in visualization tools of this kind are discussed, including the use of sampling, transformations and artificial intelligence. Additionally, a subset of these techniques is implemented and analyzed according to criteria such as responsiveness and accuracy using real and synthetic data.

Given the increasing value and use of big data applications, the study of progressive visualization techniques represents an important path for the design and development of interactive, scalable tools capable of providing information to users quickly and meaningfully over the Internet. Thus, this work poses relevant considerations regarding the requirements and feasibility of visualization systems that offer a smooth and effective user experience, expeditiously responding to questions about large volumes of data and their evolution over time.

Sumário

Lista de Figuras **x**

Lista de Tabelas **xii**

1 Introdução **1**

2 Revisão Bibliográfica **5**

2.1 Séries temporais 5

2.2 Visualizações de dados multidimensionais 5

2.3 Visualização de dados de séries temporais 7

2.4 Progressividade em visualização de dados 9

3 Estratégias para visualização progressiva de séries temporais **20**

3.1 Abordagem sem progressividade 20

3.2 Progressividade por amostragem 21

3.3 Progressividade por transformadas 21

3.4 Progressividade por aprendizado de máquina 25

4 Arquitetura e solução experimental **28**

4.1 Soluções para transmissão assíncrona de dados 29

4.2 Desenvolvimento de APIs 31

4.3 Uso prático 33

4.4 Arcabouço experimental 35

5 Experimentação **41**

5.1 Conjunto de dados 41

5.2 Resultados 44

5.2.1 Abordagem não progressiva 45

5.2.2 Abordagem por amostragem 47

5.2.3 Abordagem por transformadas 59

5.2.4 Abordagem por Redes Neurais 67

5.3 Discussão dos resultados 73

5.4 Escalabilidade 81
5.5 Limitações 82

6 Conclusões **85**
6.1 Trabalhos Futuros 86

Referências Bibliográficas **89**

Lista de Figuras

1.1 Comparação de dois níveis de detalhamento vistos no OpenStreetMap. À esquerda, nível de detalhamento menor. À direita, nível de detalhamento maior quando realizado zoom. 2
1.2 Captura de tela da ferramenta exploratória de visualização de dados Amplia Saúde exibindo uma comparação de séries temporais sobre nascimentos entre os municípios de Rio de Janeiro, São Paulo e Brasília. 4

2.1 Consulta utilizada para agregação M4 conforme descrito por Jugel et al. (2014) 8
2.2 Estrutura do sistema baseado em amostragem proposto por Ding et al. (2016) 11
2.3 Comparação do fluxo convencional e do fluxo progressivo proposto por Li e Ma (2019) 12
2.4 Comparação dos tempos de recepção dos dados entre sistemas de visualização com carregamento progressivo (PVA), sistemas instantâneos (IVA) e sistemas monolíticos (MVA) por Angelini et al. (2018) [1] . . 14
2.5 Classificação visual de algoritmos de compressão para séries temporais segundo Chiarot e Silvestri (2022) 16

4.1 Diagrama da arquitetura proposta 29

5.1 Série temporal do atributo G1 com agrupamento pela média 42
5.2 Série temporal do atributo S1 com agrupamento pela média 43
5.3 Gráfico comparativo entre tempos e erros de algoritmos no cálculo de médias diárias da variável S1 testados com conjunto de 100.000 linhas e 10% de amostragem 55
5.4 Diagrama comparativo do tipo "box plot" de erros de algoritmos testados com conjunto de 100.000 linhas 56
5.5 Gráfico da série temporal da média diária geral do preço de fechamento das ações gerado por 10% de amostragem utilizando KNN sobre dados reais 58

5.6 Gráfico do erro quadrático médio da série reconstruída via FFT a partir do conjunto de dados sintéticos com 100.000 linhas em relação à quantidade de coeficientes utilizados em escala *symlog* 61
5.7 Gráfico do erro quadrático médio da série reconstruída via FFT a partir do conjunto de dados sintéticos com 100.000.000 linhas em relação à quantidade de coeficientes utilizados em escala *symlog* . . . 62
5.8 Gráfico do volume de dados transmitido em relação à quantidade de coeficientes da FFT utilizados para dados sintéticos 63
5.9 Gráfico do erro quadrático médio da série reconstruída via FFT e do volume dos coeficientes transmitidos a partir do conjunto de dados real em relação à quantidade de coeficientes utilizados 64
5.10 Gráfico do erro quadrático médio da série reconstruída a partir do conjunto de dados sintético de 100.000 linhas em relação à *wavelet* utilizada 66
5.11 Diagrama da arquitetura da rede neural implementada 68
5.12 Série temporal da média diária da variável S1 com filtro $N2 < 20$ gerada pela rede neural com erro quadrático médio de 211,730 68
5.13 Comparação das medições relativas à rede neural com dados sintéticos em relação ao número de nós das camadas ocultas 70
5.14 Volume dos dados transmitidos vs número de nós das camadas ocultas da rede neural 71
5.15 Série temporal da média dos preços de fechamento com aplicação do filtro $open < 1579$ gerada pela rede neural com erro quadrático médio de 129,122 73
5.16 Comparação entre séries reconstruídas a partir da inversa da FFT com 2, 10 e 50 coeficientes e série exata da média geral do preço de fechamento com dados reais 76
5.17 Comparação entre séries reconstruídas a partir da inversa da transformada de *wavelets* de Daubechies com diferentes números de coeficientes e níveis de detalhe para a média geral dos preços de fechamento do conjunto de dados reais 77
5.18 Comparação geral entre técnicas testadas de acordo com diferentes métricas 78

Lista de Tabelas

5.1 Tabela de medições relativas ao caso-base sem progressividade para o cálculo da série temporal da média de S1 com conjunto de dados fictícios 47
5.2 Tabela de tempos, cobertura e erro médio quadrático (MSE) da média diária de S1 por porcentual amostrado para o conjunto de 100.000 linhas 48
5.3 Tabela de tempos, cobertura e erro médio quadrático (MSE) por porcentual amostrado com agrupamento por data da média diária da variável S1 para o conjunto de 100.000 linhas 49
5.4 Tabela de tempos, cobertura e erro médio quadrático (MSE) da média diária de S1 por porcentual amostrado via KNN para o conjunto de 100.000 linhas 51
5.5 Tabela de tempos, cobertura e erro médio quadrático (MSE) da média diária da variável S1 por porcentual amostrado via DBSCAN para o conjunto de 100.000 linhas 52
5.6 Tabela de tempos, cobertura e erro médio quadrático (MSE) da média diária de S1 por porcentual amostrado via OPTICS para o conjunto de 100.000 linhas 53
5.7 Quadro comparativo entre algoritmos com amostragem de 10% sobre a variável S1 para um conjunto de dados de 100.000 linhas 54
5.8 Quadro comparativo entre algoritmos com amostragem de 10% sobre a variável S1 para um conjunto de dados de 100.000.000 de linhas . . 54
5.9 Quadro comparativo da geração da série temporal referente à variável S1 através do algoritmo KNN com amostragem de 10% sob diferentes filtros 55
5.10 Quadro comparativo entre algoritmos com amostragem de 10% sobre o conjunto de dados reais para as médias diárias do preço de fechamento das ações 58
5.11 Tabela de medições de decomposições FFT para médias diárias das variáveis G1 e S1 para 100.000 linhas 60

5.12 Tabela de medições de decomposições FFT para médias diárias das variáveis G1 e S1 para 100.000.000 linhas 60
5.13 Tabela de medições de decomposições FFT com dados reais relativas à média diária do preço de fechamento das ações 64
5.14 Tabela de medições de decomposições com *wavelets* de Daubechies para variáveis G1 e S1 para 100.000 linhas 65
5.15 Tabela de medições de decomposições com *wavelets* de Daubechies para variáveis G1 e S1 para 100.000.000 linhas 65
5.16 Tabela de medições de decomposições com *wavelets* de Daubechies para dados reais referentes à média diária do preço de fechamento de todas as ações 66
5.17 Tabela de medições da rede neural com dados sintéticos considerando a média diária da variável S1 e filtro $N2 < 50$ 69
5.18 Tabela de medições de erros da rede neural com dados reais considerando o preço médio de fechamento de todas as ações 72
5.19 Matriz de *trade-offs* entre técnicas de visualização progressiva 79

Capítulo 1

Introdução

Com a disseminação de novas tecnologias e o aumento no número de dispositivos e aplicações gerando e consumindo dados, a boa interpretação desses dados é imprescindível para extrair deles valor e conhecimento.

O objeto de estudo do campo de visualização de dados concentra-se em traduzir dados brutos em visuais que façam parte de uma interface de mais fácil compreensão e obtenção de informações. Mas, para além desse desafio, essa área do conhecimento também se depara com conjuntos de dados cada vez mais volumosos. O recrudescimento dos grandes volumes de dados, comumente referidos como *Big Data*, sejam oriundos de numerosas e diversas fontes de dados com o advento da Internet das Coisas (*Internet of Things*, ou IoT) ou das interações sempre crescentes nas redes sociais, provocou um célere desenvolvimento de novas técnicas de computação distribuída e progressiva para a extração de estatísticas e a modelagem desses dados.

Assim como novas técnicas foram criadas para a rápida análise desses grandes conjuntos de dados, novas técnicas precisam existir para visualizá-los. A visualização de dados no contexto de *Big Data* mostra-se tanto mais desafiadora quanto mais necessária para interpretação e geração de *insights* tornando o que seria um conjunto de dados grande demais para ser consumido em algo não apenas humanamente legível como também intuitivamente compreensível.

Em especial, para visualizações web, em que dados precisam ser transmitidos através da Internet, esse grande volume de dados pode gerar longos tempos de espera para o usuário final.

Um exemplo comum de aplicação que se faz presente em nosso cotidiano e ilustra esse problema é o carregamento progressivo de informações geográficas e georreferenciadas em mapas online, como visto na Figura 1.1. Com base no nível atual de zoom, mais (ou menos) detalhes são exibidos. Níveis de zoom menores, que exibem países e estados, por exemplo, não exigem a transmissão e exibição de informações sobre ruas e comércio local. Já níveis de zoom mais altos devem incluir essas e outras informações de possível interesse dos usuários.

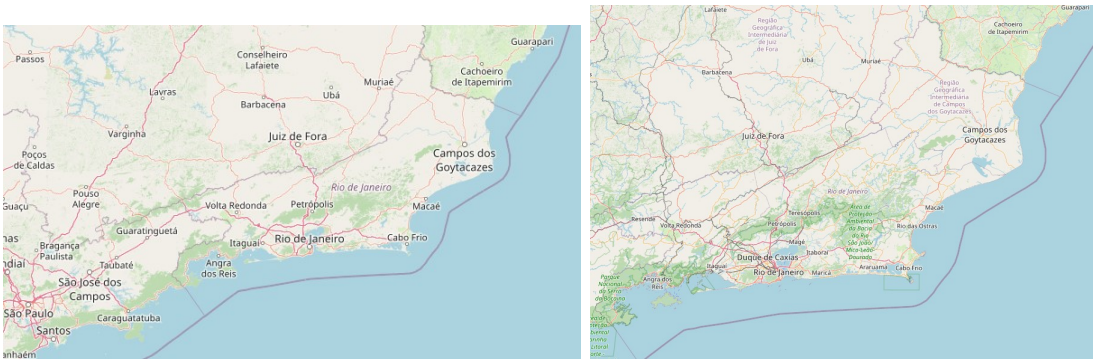


Figura 1.1: Comparação de dois níveis de detalhamento vistos no OpenStreetMap. À esquerda, nível de detalhamento menor. À direita, nível de detalhamento maior quando realizado zoom.

Na prática, o mesmo conjunto de dados é pré-processado em diferentes níveis de detalhamento (em inglês, *Level of Detail*, e comumente referidos como *LoD*). Essa técnica permite uma economia dos recursos a serem enviados através da rede em troca de um maior processamento por parte do servidor antes de enviar as informações.

Mapas não são o único tipo de aplicação a se beneficiar desses e outros tipos de técnica. Na realidade, mapas compõem apenas um de uma diversidade de visualizações a fazerem uso de grandes volumes de dados, que demandam capacidades de interação e acessíveis pela Internet. Por exemplo, visualizações de dados de saúde, como o painel de vigilância de COVID-19 da University of Virginia [2] e visualizações de dados de redes sociais, como o sentiment viz: Tweet Sentiment Visualization da North Carolina State University [3], abarcam uma variedade de visuais gráficos, como gráficos de barras e de linhas, e oferecem diversas agregações e filtragens, como recortes temporais e médias móveis.

Para esses outros tipos de visualização, outras técnicas para otimização do carregamento dos dados podem ser aplicadas, visando reduzir o tempo de espera do usuário para obter as informações desejadas tanto na primeira exibição da tela quanto em resposta a suas interações. Inclui-se em tal tempo o processamento dos dados feito pelo servidor, a transmissão através da rede e o processamento do lado do cliente.

Neste trabalho, terá principal enfoque o uso de técnicas que otimizem o carregamento de visualizações de dados que tenham como base dados de natureza temporal. Esse tipo de dado nos permite explorar certas características únicas a esse formato, como suas ergodicidades e autocorrelações. Algoritmos pensados para séries temporais podem se fazer valer dessas características para comprimir as informações ou transmitir progressivamente alguns pontos da série, por exemplo.

A transmissão progressiva, a principal técnica a ser discutida neste trabalho, se refere ao envio parcial dos dados de modo a gerar uma imagem preliminar rapi-

damente para o usuário enquanto aguarda o envio do restante da informação [4]. Parâmetros como o tamanho dos blocos e o critério de escolha da informação enviada preemptivamente devem ser ajustados para otimizar a experiência do usuário, o que já representa um desafio por si só.

A motivação para o estudo desse tema surge no contexto do projeto "Amplia Saúde: Observatório dos períodos pré- e perinatal" apoiado pela *Bill & Melinda Gates Foundation* através do programa *Grand Challenges Explorations*. Utilizando dados do Sistema de Informações sobre Nascidos Vivos (SINASC) e do Sistema de Informações sobre Mortalidade (SIM), inseridos no ecossistema do Departamento de Informática do Sistema Único de Saúde (DATASUS), o Amplia Saúde oferece uma ferramenta exploratória de visualização de dados sobre natalidade e saúde materna e neonatal para pesquisadores de áreas de saúde e para outros interessados por esse tema [5].

A ferramenta também é capaz de visualizar os dados de poluição do ar (na forma da concentração de material particulado menor que 2,5 micrômetros, chamado de MP2.5), extraídos do Sistema de Informações Ambientais Integrado à Saúde (SISAM) do Instituto Nacional de Pesquisas Espaciais (INPE). Isso viabiliza a exploração de diferentes recortes e cruzamentos de dados e a geração de *insights* sobre a evolução da saúde materna e infantil no decorrer dos anos e os possíveis impactos da poluição sobre esses indicadores. Uma tela exibindo essas séries temporais de dados sobre nascimentos pode ser vista na Figura 1.2.

Os sistemas SINASC e SIM oferecem informação a nível de microdados, em que cada linha da base de dados representa um nascimento ou um óbito. No intervalo de tempo estudado, abrangendo os anos de 2012 a 2019 completos, essas bases compõem dezenas de milhares de registros que precisam ser agregados por município e por unidade da federação e filtrados por diferentes atributos de acordo com a necessidade de exploração do usuário.

Sem o devido pré-processamento e técnicas de *caching*, os tempos de espera para consultas seriam demasiadamente longos, especialmente para cidades muito populosas como São Paulo. E, para além dessas otimizações, ao empregar envios progressivos dos dados, tais tempos poderiam ser reduzidos ainda mais, o que viabilizaria agregações a nível nacional.

Apesar da abundante pesquisa nos temas de transmissão progressiva e visualização de dados temporais, poucas publicações exploram a interseção entre os dois temas. Trata-se de um vasto tópico a ser estudado e com variadas aplicações em nosso dia-a-dia. Como será discutido nos capítulos a seguir, a transmissão progressiva envolve não apenas a técnica de segmentar e enviar os dados progressivamente como também a renderização desses dados à medida que chegam ao cliente, sem sacrificar a fluidez, a clareza e a acurácia da visualização.

Figura 1.2: Captura de tela da ferramenta exploratória de visualização de dados Amplia Saúde exibindo uma comparação de séries temporais sobre nascimentos entre os municípios de Rio de Janeiro, São Paulo e Brasília.

O restante desta dissertação é organizado da seguinte forma: no segundo capítulo, a seguir, serão discutidas produções nesta temática, constituindo um breve panorama sobre o assunto, seu desenvolvimento ao longo dos anos e o estado-da-arte hoje. No terceiro capítulo, será apresentada a metodologia proposta para ideação, implementação e avaliação de técnicas de carregamento progressivo em visualizações de dados. Já no quarto capítulo, são discutidos os principais resultados obtidos da experimentação com diferentes técnicas em termos de tempo de resposta e erro médio das aproximações. Por fim, no quinto e último capítulo, são exibidas as conclusões deste trabalho, recapitulando o que foi exposto e endereçando possíveis trabalhos futuros.

Capítulo 2

Revisão Bibliográfica

2.1 Séries temporais

Consideraremos, para este trabalho, a conceituação de séries temporais como sequências de valores mensurados em intervalos de tempo iguais [6]. Em especial, desejamos obter séries temporais de funções de agregação, como médias e máximas, ou ainda outras estatísticas de segunda ordem, como variâncias e correlações, de diferentes colunas de conjuntos de microdados e de grande cardinalidade.

A modelagem de séries temporais é uma área de estudo consolidada, com diversas abordagens matemáticas e estatísticas disponíveis. Modelos clássicos de aprendizagem de máquina, como médias móveis integradas autorregressivas (ARIMA, do inglês *Autoregressive Integrated Moving Average*), são amplamente utilizados para capturar padrões lineares em séries temporais. Por outro lado, técnicas mais recentes, como redes neurais recorrentes (RNNs, do inglês *recurrent neural networks*) e memória de curto prazo longa (LSTM, do inglês *Long Short-Term Memory*), têm se mostrado eficazes para lidar com séries temporais mais complexas e não lineares.

Esling e Agon (2012) sintetizam as principais definições e técnicas empregadas no estudo de séries temporais, como a segmentação, predição e detecção de anomalias [7].

2.2 Visualizações de dados multidimensionais

O estudo de visualizações de dados multidimensionais tem se mostrado essencial no contexto de análise de grandes volumes de dados, permitindo a exploração e interpretação de informações complexas de maneira mais intuitiva e acessível. Dados multidimensionais, por sua natureza, apresentam desafios únicos, uma vez que envolvem múltiplas variáveis que precisam ser representadas de forma simultânea e compreensível.

Esse processo exige a aplicação de técnicas que vão além da simples representação gráfica. É necessário considerar aspectos como a escolha adequada de gráficos, a redução de dimensionalidade para dados complexos, a interatividade para personalização das análises e a otimização do carregamento para garantir uma experiência fluida ao usuário. Além disso, a visualização deve ser capaz de destacar padrões, tendências e anomalias, transformando o que seria um conjunto de dados ininteligível em algo acessível e significativo.

Uma das abordagens mais comuns para lidar com essa complexidade é o uso de técnicas de redução de dimensionalidade, como a Análise de Componentes Principais (PCA, do inglês *Principal Component Analysis*) e o *Embedding* de Vizinhos Estocásticos t-Distribuídos (t-SNE, do inglês *t-Distributed Stochastic Neighbor Embedding*). Essas técnicas permitem projetar dados de alta dimensionalidade em espaços de menor dimensão, preservando, na medida do possível, as relações entre os dados originais. No entanto, a escolha da técnica mais adequada depende do contexto da aplicação e das características dos dados analisados.

Além disso, a utilização de gráficos interativos tem ganhado destaque como uma ferramenta poderosa para a exploração de dados multidimensionais. Ferramentas como gráficos de dispersão com múltiplas dimensões representadas por cores, tamanhos ou formas, bem como matrizes de correlação e gráficos paralelos, são amplamente empregadas para identificar padrões, tendências e anomalias nos dados. A interatividade permite que os usuários ajustem filtros, explorem diferentes perspectivas e obtenham insights mais profundos, adaptando a visualização às suas necessidades específicas.

Dzemyda, Kurasova e Zilinskas (2012) propõem aplicações de redes neurais treinadas sobre conjuntos de dados multidimensionais para estimar a projeção de novos dados e visualizá-los em um plano bidimensional. Outra proposta é, por exemplo, feita por Xu, Xu e Chow (2009) com o algoritmo PolSOM, uma variante do algoritmo SOM (*self-organizing map* ou mapa auto-organizável) utilizando coordenadas polares para a visualização dos dados.

Os mapas auto-organizáveis são uma classe de redes neurais que se destacam na análise e visualização de dados complexos, especialmente em contextos onde a redução de dimensionalidade é necessária. Desenvolvidos por Teuvo Kohonen na década de 1990, os SOMs utilizam um processo de aprendizado não supervisionado para organizar dados em uma grade bidimensional, preservando as relações topológicas dos dados originais [8].

A estrutura de um SOM é composta por uma rede de neurônios, onde cada neurônio é associado a um vetor de pesos que representa uma posição no espaço de entrada. Durante o treinamento, os neurônios competem para se tornarem os melhores representantes dos dados apresentados. Tais neurônios denotam *clusters*

de dados com grande similaridade.

Uma das principais vantagens dos SOMs é sua capacidade de projetar dados de alta dimensionalidade em um espaço de menor dimensão, o que é particularmente útil em aplicações que envolvem grandes volumes de dados (*big data*). Essa projeção não apenas facilita a visualização, mas também permite a identificação de agrupamentos e tendências que poderiam ser difíceis de discernir em representações mais complexas. No entanto, os resultados do uso dessa arquitetura são fortemente dependentes da escolha minuciosa de hiperparâmetros adequados durante a fase de treinamento.

Barreto (2007) aborda a aplicação de variações do modelo SOM para séries temporais e, em suas conclusões, destaca que não existem arquiteturas desse tipo para séries temporais de dados multivariados [9].

2.3 Visualização de dados de séries temporais

A visualização de dados para séries temporais é uma prática essencial na análise de informações que evoluem ao longo do tempo, permitindo que padrões, tendências e anomalias sejam identificados de forma clara e intuitiva. Dados temporais frequentemente carregam uma complexidade inerente que pode ser difícil de interpretar em formatos tabulares ou textuais, tornando a representação gráfica uma ferramenta indispensável para analistas e pesquisadores.

O gráfico de linhas é uma das técnicas mais tradicionais e amplamente utilizadas para a visualização de séries temporais. Ele conecta pontos de dados em ordem cronológica, destacando a evolução de uma variável ao longo do tempo. Essa abordagem é particularmente eficaz para identificar tendências de longo prazo, flutuações sazonais e mudanças abruptas. Em contextos como finanças, saúde e meteorologia, gráficos de linhas são frequentemente empregados para análises detalhadas, como a variação de preços de ações ou a evolução de indicadores climáticos.

Além dos gráficos de linhas, outras representações visuais têm sido desenvolvidas para atender às demandas de análises mais sofisticadas. Gráficos de barras empilhadas, por exemplo, são úteis para representar a composição de categorias ou grupos dentro de uma série temporal, permitindo uma análise comparativa mais detalhada.

Outro recurso valioso na análise de séries temporais é a decomposição, que separa os dados em componentes como tendência, sazonalidade e ruído. Essa técnica permite uma compreensão mais profunda dos fatores que influenciam as variações temporais, destacando elementos estruturais e padrões subjacentes. A decomposição é especialmente útil em análises que buscam isolar efeitos específicos ou prever comportamentos futuros.

Técnicas mais avançadas, como *heatmaps* temporais, também têm ganhado des-

taque na visualização de séries temporais. Esses gráficos utilizam uma grade de cores para representar a intensidade dos dados ao longo do tempo, sendo particularmente eficazes para identificar padrões sazonais e comportamentos recorrentes em grandes volumes de dados. Gráficos de dispersão animados, por sua vez, permitem a análise simultânea de múltiplas variáveis, mostrando sua evolução ao longo do tempo de forma dinâmica e contextualizada.

Por fim, a escolha da técnica de visualização deve ser guiada pelo objetivo da análise e pelo público-alvo. Visualizações eficazes de séries temporais são aquelas que conseguem traduzir a complexidade dos dados em representações claras, intuitivas e informativas.

Fang, Xu e Jian (2019) traçam um panorama da pesquisa no campo de visualização de séries temporais levantando as ferramentas que endereçam esse desafio de diferentes formas [10]. Alguns projetos obtiveram relativo sucesso ao explorar a representação de dados temporais com alta dimensionalidade, utilizando animações e o formato de coordenadas paralelas (em inglês, *parallel coordinates*) e codificando dados em atributos como cor e opacidade. Entretanto, pouco foi estudado em termos de otimizações para reduzir os tempos de carregamento e permitir sistemas de visualização interativos, em que o usuário interage com a ferramenta e observa reflexos em tempo real de suas ações.

Já Jugel et al. (2014) apresentam uma técnica de redução de dimensionalidade para agregação de dados temporais buscando menores latências mesmo em grandes volumes de dados [11]. Para isso, são computados dois pares de mínimos e máximos em intervalos de tempo equidistantes que são mapeados na largura de um pixel no espaço de visualização, como na *query* exibida na Figura 2.1.

```
SELECT t,v FROM Q JOIN
(SELECT round($w*(t-$t1)/($t2-$t1)) as k, --define key
       min(v) as v_min, max(v) as v_max, --get min,max
       min(t) as t_min, max(t) as t_max  --get 1st,last
       FROM Q GROUP BY k) as QA          --group by k
ON k = round($w*(t-$t1)/($t2-$t1))       --join on k
   AND (v = v_min OR v = v_max OR        --&(min|max|
       t = t_min OR t = t_max)           --  1st|last)
```

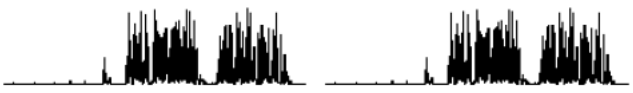


Figura 2.1: Consulta utilizada para agregação M4 conforme descrito por Jugel et al. (2014)

Outro trabalho com propósito similar é Martin e Quach (2016), porém com o diferencial de permitir ao usuário escolher dentre diferentes algoritmos implementados qual será utilizado para a otimização na geração da série temporal [12]. Os autores implementaram em sua ferramenta algoritmos de escalonamento multidimensional, diferenças de séries temporais e clusterização por metadados.

O resultado desse estudo é um projeto altamente flexível, permitindo ao usuário a escolha dos algoritmos mais adequados no contexto específico de sua exploração dos dados, bem como permite o carregamento de diferentes conjuntos de dados sem necessidade de um pré-processamento ou treinamento prévio. No entanto, tal ferramenta foi desenvolvida como um executável *standalone* e não foi pensada para um sistema de visualização de dados a ser consumido por um navegador web como é feito por grandes públicos atualmente.

Por fim, no campo do design da informação, muitos estudos já foram conduzidos com base em entrevistas com usuários para determinar as melhores formas de visualizar dados temporais, tendo como foco uma apreensão mais veloz e eficiente da informação exibida. Nesse contexto, trabalhos como Saraiya, Lee e North (2005) apontam que, para conjuntos de dados multidimensionais, a visualização de uma série temporal de um atributo de cada vez levou a uma melhor experiência de usuário, permitindo que tarefas de análise e comparação envolvendo tais dados fossem concluídas mais rapidamente e com maior acurácia [13].

Dessa forma, é estabelecida a recomendação de construir sistemas que permitam a visualização de séries temporais de maneira independente entre si, desencorajando, por exemplo, a estratégia de exibir séries temporais referentes a diferentes atributos em um mesmo espaço visual empregada nas ferramentas de visualização mais clássicas abordadas aqui. Apesar do enfoque mais direcionado à tecnologia que viabiliza esse tipo de ferramenta de visualização no escopo desta dissertação, o alinhamento às boas práticas de design pode – e deve – servir como princípio norteador nas escolhas de implementação, podendo influenciar na arquitetura técnica do sistema de visualização proposto.

2.4 Progressividade em visualização de dados

Podemos esboçar um breve panorama sobre o surgimento e o desenvolvimento do carregamento progressivo de dados na Web e entender como se dá sua evolução até hoje.

Gilbert e Brodersen (1999), em um dos primeiros trabalhos a tocar em tal assunto, ainda no ano de 1999, cunham o termo "progressive delivery" para se referir a um carregamento parcial e mais rápido de um recorte dos dados totais a serem transmitidos de modo a oferecer ao usuário uma experiência mais interativa enquanto espera o carregamento total de uma página [4]. Este trabalho não aborda o contexto de visualização de dados em si, mas é um dos textos fundacionais para a disseminação dessa técnica nos anos seguintes.

Para endereçar o escopo a ser tratado neste trabalho, é fundamental entender o contexto em que se encontra atualmente o estado-da-arte no tema de visualização

progressiva.

É possível dividir as abordagens para a resolução do problema proposto em dois tipos distintos: utilizando compressão para reduzir o tempo de transmissão dos dados através da rede e utilizando técnicas de amostragem para obter uma abordagem representativa da totalidade dos dados a partir de uma parte deles. A primeira tem maior enfoque na redução do tempo de transmissão dos dados, enquanto a segunda visa reduzir o tempo de processamento dos dados no servidor.

As técnicas de compressão podem ser subdivididas em vários outros tipos, de acordo com a perda de informação associada à compressão ou com os tipos de dados a serem comprimidos, por exemplo [14]. Os algoritmos utilizados por essas técnicas evoluíram muito nas últimas décadas, numa constante busca aos que melhor se adaptam às mídias e meios de suas épocas, como a compressão em tempo real para *streaming* de dados [15] [16] [17] [18] [19].

Em especial, já no contexto delineado para este trabalho, destaca-se a técnica utilizada por Lehmann e Jung (2014) para visualização de grandes volumes de dados de natureza temporal e em múltiplas resoluções [20]. Por sua vez, Sengupta e Kasabov (2017) discutem *frameworks* para a codificação (método de compressão) que mantenham padrões encontrados em séries temporais não comprimidas, garantindo que os dados comprimidos ainda sejam reconhecíveis pelos usuários [21].

No campo da visualização de dados, os trabalhos mais prevalentes que discutem a compressão de dados aplicam a compressão a imagens biomédicas de volumes [22] [23] e a malhas topográficas [24].

Já uma popular escolha arquitetural em sistemas de visualização de dados é o uso do chamado *query processor*, ou "processador de consultas" em tradução livre. São geralmente associados a sistemas de gerenciamento de banco de dados (SGBDs), sendo um componente essencial responsável por interpretar e executar consultas feitas pelos usuários. Segundo a definição da Encyclopedia of Database Systems de 2009, ele transforma essas consultas, que podem incluir operações de filtragem, agregação e ordenação, em planos de execução eficientes [25].

Entre seus principais benefícios, destaca-se a eficiência no processamento de consultas, uma vez que o *query processor* é capaz de interpretar e otimizar as requisições feitas pelos usuários, transformando-as em chamadas que reduzem o tempo necessário para processar grandes volumes de dados. Outra vantagem é sua adaptabilidade a diferentes arquiteturas. O *query processor* pode ser configurado para operar tanto no servidor quanto no cliente, dependendo das necessidades do sistema, o que permite otimizar a performance e a responsividade em diferentes contextos.

Ding et al. (2016) propõem uma abordagem baseada em amostragem [26]. Em resumo, as requisições de agregação recebidas como entrada do sistema são processadas em um *query processor*, que calcula a agregação desejada para uma amostra

randomicamente selecionada do total dos dados, e fornece como saída uma estimativa do resultado real efetuado sobre o conjunto completo de linhas da base de dados.

É importante destacar que, em geral, a amostragem geralmente não pode ser pré-computada, já que os dados a serem visualizados dependem diretamente das operações de filtragem e agregação realizadas pelo usuário. Isso exige que o *query processor* seja capaz de gerar planos de execução adaptáveis, considerando as mudanças nas consultas e otimizando o processamento para fornecer resultados parciais rapidamente.

Além disso, a localização do *query processor*, seja no servidor ou no cliente, desempenha um papel crucial nas soluções de visualização progressiva. Se o processamento ocorrer no servidor, a carga de trabalho pode ser distribuída, mas isso pode resultar em latências maiores na entrega dos dados ao cliente. Por outro lado, um *query processor* no cliente pode oferecer respostas mais rápidas, mas pode ser limitado pela capacidade de processamento local e pela quantidade de dados que pode ser manipulada. Portanto, as soluções para visualização progressiva devem considerar esses aspectos para otimizar a experiência do usuário e a eficiência do sistema.

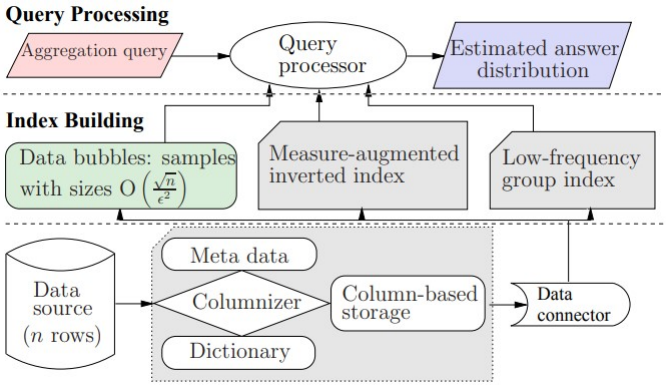


Figura 2.2: Estrutura do sistema baseado em amostragem proposto por Ding et al. (2016)

O processamento também pode ser efetuado sobre um índice previamente computado, que fornece a informação de que linhas possuem maior probabilidade de serem sorteadas para compor o cálculo aproximado.

Li e Ma (2019) propõem um sistema de visualização de dados paralelizado, progressivo e portátil chamado "P5", do nome *Portable Progressive Parallel Processing Pipelines for Interactive Data Analysis and Visualization* [27]. P5 é um sistema de visualização para a Web, com sua arquitetura baseada em um *query processor* localizado no servidor. Ele oferece uma gramática declarativa para a construção de visualizações de dados utilizando tal sistema e é construído sobre um trabalho

anterior dos mesmos autores intitulado “P4”, adicionando a ele a funcionalidade de carregamento progressivo.

Dentre as declarações possíveis, destaca-se a possibilidade de optar entre três “modos” de processamento dos dados: automático, semi-automático e manual. No funcionamento automático, logo após o processamento e a renderização de uma partição dos dados, a partição seguinte começa a ser processada. Já no modo semi-automático, o processamento analítico e a renderização dos dados ocorrem de forma separada e, conforme as partições são processadas uma após a outra, a renderização das partições deve ser atualizada manualmente através de um sinal explícito recebido no *pipeline*. Por fim, no modo manual, tanto o processamento quanto a renderização são feitos com comandos explicitamente.

Essa arquitetura é especialmente útil em aplicações em que se deseja permitir uma customização do aspecto progressivo da visualização ou que a atualização dos elementos visuais esteja sujeita a algum outro aspecto sistêmico ou a uma limitação de hardware.

No navegador, assim como é costumeiro para aplicações web, a ferramenta foi desenvolvida em JavaScript. A progressividade do *framework* P5 faz uso de um shader WebGL responsável pela acumulação dos dados que são recebidos pelo cliente e enviados pelo servidor executado em NodeJS que acessa o conjunto de dados armazenado em um banco de dados MySQL. A transmissão baseia-se na API (Interface de Programação de Aplicação, do inglês *Application Programming Interface*) referida como FileAPI, construída para transmissão de arquivos em HTML5, além do protocolo HTTP e de WebSockets.

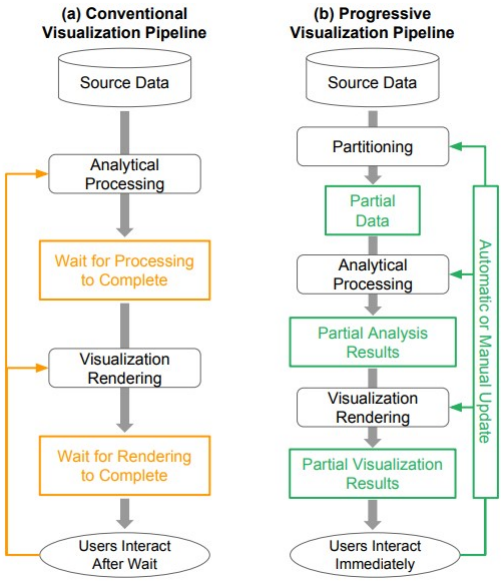


Figura 2.3: Comparação do fluxo convencional e do fluxo progressivo proposto por Li e Ma (2019)

Compressão e amostragem não são opostos. São, na verdade, conceitos mais similares do que podem aparentar. Essas e outras técnicas compõem um espectro, sendo possível pensar em amostragem como uma forma de compressão e vice-versa.

Já nos aprofundando mais no escopo deste trabalho, podemos analisar também publicações com maior enfoque em dados de natureza temporal. Ulmer, Sessler e Kohlhammer (2019) propõem um sistema de visualização para a Web especializado na análise de capturas de pacotes de dados de redes de computadores [28]. Essa prática, muito utilizada no campo de cibersegurança, envolve a inspeção e interação de milhares de pacotes, incluindo etapas de filtragem e agregação.

Em especial, a visão "timeline" dialoga de forma próxima com os objetivos deste trabalho. Ulmer, Sessler e Kohlhammer utilizam um gráfico de barras com a variável temporal no eixo X e o número de pacotes no eixo Y. Dessa forma, conforme novos pacotes são enviados do servidor para o cliente, as barras crescem de tamanho verticalmente, indicando que o processo de carregamento progressivo está em andamento.

É possível, também, selecionar um intervalo temporal para filtragem dos dados. Os autores consideraram que o número ideal de barras a serem exibidas simultaneamente é por volta de 50 e comentam sobre a inviabilidade técnica por parte dos navegadores web de exibir cada pacote individualmente.

É importante notar as diferenças entre otimizações em nível de *rendering* e otimizações em nível de processamento. O primeiro caso, focado no *rendering*, é dedicado a exibir os dados o mais rápido possível, conforme chegam ao cliente, não se preocupando em otimizar a forma como são processados e enviados do servidor. Já o segundo caso trabalha sobre como o servidor pode gerar uma resposta mais rápida e, idealmente, garantir que a tal resposta, aproximada e/ou parcial, seja representativa do resultado final.

O trabalho de Ulmer, Sessler e Kohlhammer, por exemplo, implementa melhorias a nível de *rendering*, buscando representar mais rapidamente os dados disponíveis no momento na tela para consumo do usuário. No entanto, não existem garantias de que, para um instante qualquer no processo de carregamento progressivo, a forma da curva mostrada na tela será similar a sua forma final. Isso eventualmente pode levar a uma interpretação equivocada dos dados por parte dos usuários.

Angelini et al. (2018) fazem uma revisão dos fundamentos e das principais características encontradas em ferramentas de análise visual progressivas [1]. O artigo faz uma distinção entre os sistemas de visualização de dados progressivos (chamados pelos autores de *Progressive Visual Analytics*, ou PVA) e outras implementações.

Os autores também definem os chamados sistemas monolíticos (*Monolithic Visual Analytics*, ou MVA), em que todo o processamento necessário dos dados é feito de maneira bloqueante para a visualização e, quando concluído, enviado ao cliente

de uma só vez e os sistemas instantâneos (*Instanaeous Visual Analytics*, ou IVA), em que o pré-processamento dos dados já fora realizado e armazenado antes mesmo da requisição do usuário e enviados muito rapidamente quando demandado, em um funcionamento análogo a um sistema de *caching* em uma solução de banco de dados.

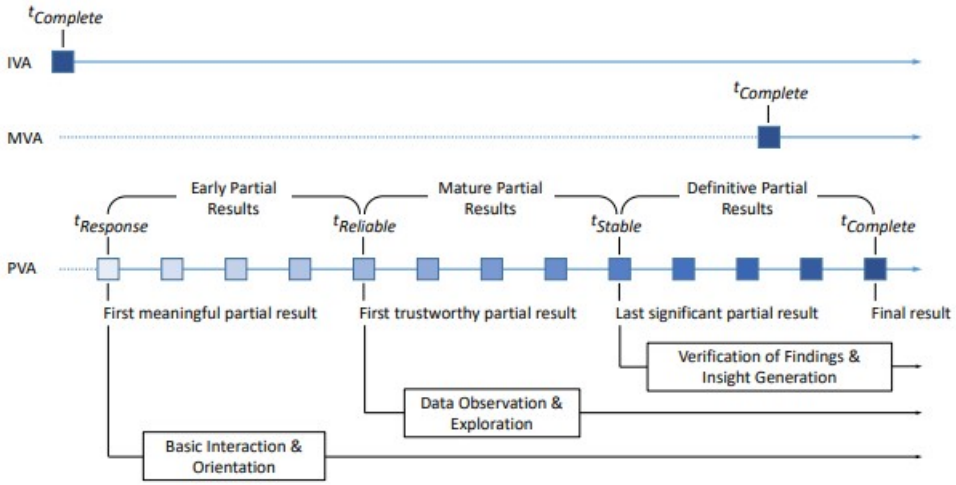


Figura 2.4: Comparação dos tempos de recepção dos dados entre sistemas de visualização com carregamento progressivo (PVA), sistemas instantâneos (IVA) e sistemas monolíticos (MVA) por Angelini et al. (2018) [1]

Ao levantar os requisitos propostos para visualizações de dados progressivas em outras oito publicações, o artigo busca caracterizar esse tipo de visualização considerando uma abordagem orientada a processos, dividindo os requisitos em quatro categorias: requisitos de dados, requisitos de processamento, requisitos de visualização e requisitos de interação.

Foi constatado que os requisitos se referem mais frequentemente a aspectos de visualização, que abarca o processo de apresentar elementos visuais dinamicamente e de forma incremental; de processamento, que engloba todo o tipo de implementação, execução e controle de computações de natureza progressiva para a manipulação dos dados; e suas interseções com a interação, que concerne aspectos da percepção e cognição humanas ao interagir com ferramentas digitais, como as limitações de tempo de resposta para oferecer uma experiência considerada fluida para um usuário.

Já em uma análise orientada ao usuário, os usuários elencam nove requisitos, cada um com uma recomendação. A seguir, encontram-se listados os requisitos propostos pela revisão:

- Prover resultados parciais antecipadamente.
- Prover resultados parciais maduros.
- Prover resultados parciais definitivos.

- Prover uma sucessão de resultados.
- Suportar a parametrização da progressão.
- Suportar o julgamento de resultados parciais.
- Suportar o monitoramento da sucessão de resultados.
- Suportar formas de influenciar a progressão.
- Suportar o tratamento de progressões flutuantes.

Em suma, o trabalho de Angelini et al. oferece um importante panorama das principais características que compõem uma visualização de dados progressiva [1]. Não apenas prevalentes, tais pontos são essenciais para a construção de uma ferramenta de visualização progressiva que atenda às expectativas dos usuários, colhendo os benefícios do carregamento não bloqueante dos dados e superando os desafios que os resultados parciais podem oferecer à tomada de decisão.

O trabalho recomenda que (1) os resultados parciais sejam entregues ao usuário prontamente, ainda mantendo seu significado e a interatividade, (2) que os resultados parciais possam "amadurecer", reduzindo a incerteza associada a eles e comunicando o grau de confiabilidade da informação exibida, e (3) que os resultados parciais convirjam para um resultado definitivo, deixando claro o tempo para completar a entrega progressiva.

Em termos de renderização, a disposição dos elementos visuais na tela deve ser atualizada à medida que os dados são recebidos e complementados. Isso deixa claro o aspecto progressivo e a resposta às interações do usuário com a visualização. Além disso, a complexidade visual deve ser minimizada e mantida de forma consistente entre os diferentes layouts e telas que compõem o sistema de visualização.

Um dos principais desafios encontrados por esse tipo de visualização é garantir a estabilidade do carregamento e a transparência ao usuário de que os dados são carregados progressivamente e de como suas interações impactam no *pipeline* de visualização. Segundo Angelini et al., isso pode ser alcançado através da combinação entre elementos estáticos e dinâmicos, trabalhando o posicionamento e as animações do que está em fase de carregamento e do que já se encontra consolidado [1].

O uso de compressão e carregamento progressivo pode ser pensado de maneira conjunta. Por exemplo, alguns algoritmos de compressão se adequam melhor ao envio contínuo de dados. Uma das mais célebres técnicas com essa característica é a codificação de Huffman adaptativa, implementada pelos algoritmos FGK (Faller-Gallager-Knuth) [29] e Vitter [30].

Diferentemente da abordagem mais tradicional em que todo o conjunto de dados é percorrido para o cálculo das frequências e criação de um dicionário, como a codifi-

cação de Huffman original e a codificação aritmética [31], as codificações adaptativas permitem que tal processamento seja feito à medida que os dados são transmitidos pela rede. Ao dispensar o conhecimento do conjunto completo de dados para posterior compressão, os algoritmos que implementam essa codificação viabilizam que a transmissão dos dados – e, em nosso contexto, uma transmissão progressiva – possa ser iniciada de maneira imediata, reduzindo o tempo de espera para o início do recebimento dos dados e da consequente renderização dos resultados parciais para consumo do usuário.

Muitas técnicas possuem suas respectivas versões adaptativas e aplicáveis ao contexto de transmissão progressiva. No campo das codificações aritméticas, por exemplo, a *context-adaptive binary arithmetic coding* (CABAC), ou codificação aritmética binária adaptativa ao contexto, é uma codificação adaptativa utilizada para transmissão em tempo real de arquivos de vídeo nos padrões H.264/MPEG-4 AVC e HEVC [32]. Essas são técnicas sem perdas (ou *lossless*) em que a informação original consegue ser recuperada de forma exatamente idêntica ao estado anterior à compressão, em contraste com as codificações com perdas (*lossy*).

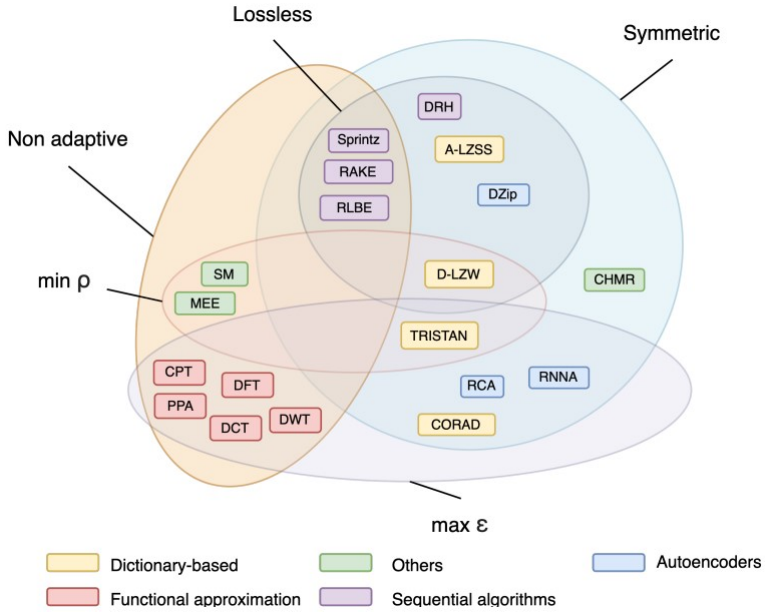


Figura 2.5: Classificação visual de algoritmos de compressão para séries temporais segundo Chiarot e Silvestri (2022)

No entanto, as taxas de compressão e transmissão encontram um limite associado à entropia, uma medida de desordem e incerteza, do conjunto de dados. Estudos de Teoria da Informação nos permitem, inclusive, relacionar a entropia às flutuações e irregularidades de uma série temporal [33].

Existem, também, as técnicas de compressão com perdas, ou *lossy*. Com elas, as saídas comprimidas não carregam toda a informação presente no original e, por isso, o processo inverso de descompressão para reobter este original com exatidão é

impossível [34]. Assim, podemos ver a compressão com perdas como uma forma de gerar aproximações da entrada.

É de grande importância mencionar que diferentes métodos de compressão podem ser aplicados no contexto de uma aplicação de transmissão progressiva, com diferentes benefícios e desvantagens. Por exemplo, certas técnicas *lossy* permitem saber com precisão o que foi perdido durante o processo. Somente assim a descompressão da saída somada a essa informação adicional permite a recuperação do arquivo original. Tais formas de compressão diferem daquelas em que busca-se unicamente obter informação descomprimida suficientemente parecida com a original, baseada nas limitações da percepção humana.

Utilizando algoritmos de compressão *lossy* como o referido anteriormente, em um sistema de visualização de dados com transmissão progressiva podemos enviar antecipadamente a saída comprimida obtida e em seguida enviar a diferença para a informação original. Assim, uma versão preliminar dos dados pode ser renderizada rapidamente enquanto a versão exata é gerada e reconstruída depois. Idealmente, a informação transmitida primeiro seria aquela considerada "mais importante", ou mais representativa dos dados exatos e que inclui suas características (ou *features*) mais distintivas. Desse modo, busca-se garantir que o usuário visualize informações que efetivamente permitam uma análise e tomada de decisão acurada tão logo quanto possível e sem interrupções que comprometam uma experiência fluida em suas interações com a ferramenta.

Ao tratar a técnica de compressão dessa forma, sua aplicação torna-se muito similar a técnicas de amostragem, em que uma amostra dos dados é selecionada e enviada antecipadamente [35]. Neste caso, o algoritmo para selecionar uma amostra representativa e que dê melhor suporte ao usuário para ser renderizada primeiro faz um papel análogo ao de comprimir os dados de maneira *lossy* de modo a carregar a maior parte da informação em um recorte menor dos dados.

Em especial, no contexto de séries temporais, o uso da compressão com perdas pode trazer, ainda, outros benefícios [34]. Por exemplo, a compressão de séries temporais pode ser utilizada como um método de remoção de ruído. E, para isso, é possível estabelecer um intervalo de erro máximo tolerável da série reconstruída em relação à original [36] [37].

Também é possível se valer da natureza temporal dos dados para embasar a escolha dos ditos pontos mais representativos. Considerando o uso comum desse tipo de informação, geralmente exposta em um gráfico de linhas, e a importância da escolha de escalas adequadas para a representação dos dados em um espaço restrito como o de uma tela, máximos e mínimos locais e globais passam a ser uma escolha natural de pontos-chave a serem preservados na compressão [6].

Nesse caso, torna-se essencial conhecer o objeto de estudo no qual o usuário está

interessado. Diferentes casos de uso podem demandar a escolha de diferentes pontos na série. Enquanto mínimos e máximos podem ser bons pontos-chave para a maioria das séries temporais, informações como o período e a distância entre mínimos ou máximos podem ser mais relevantes caso seja estudada uma série com fortes padrões de sazonalidade.

Esta abordagem traz, no entanto, um importante ponto de atenção. É aceitável que o tempo total de transmissão progressivo, incluindo processamento adicional como o de compressão, seja superior ao de um envio "monolítico", isso é, de dados enviados como um único objeto de maneira bloqueante. Para os usuários da aplicação de visualização, os ganhos de usabilidade associados à obtenção mais rápida da renderização parcial e de sua atualização em resposta a interações podem compensar o tempo total para carregamento do conjunto de dados exatos.

Mais ainda, essa responsividade permite que os usuários avaliem rapidamente os dados transmitidos e ajustem filtragens e recortes na ferramenta para exibir exatamente as informações desejadas. Assim, torna-se possível economizar tempo e recursos computacionais, evitando a visualização de dados menos relevantes. Como discutido anteriormente, esse é um dos requisitos previstos por Angelini et al. [1] no que tange ao suporte a formas de influenciar a transmissão. Essa capacidade pode tornar o tempo total de utilização da ferramenta como um todo, incluindo todas as explorações feitas pelo usuário, menor no método progressivo quando comparado ao envio completo de sucessivos monólitos de dados na abordagem tradicional.

Surge, então, o desafio adicional de conhecer até que ponto tempos totais maiores de transmissão progressiva são mais vantajosos que os de um carregamento monolítico. Essa resposta, no entanto, é menos trivial do que pode aparentar. Esse limiar pode variar – e muito – de aplicação para aplicação. Diferentes dados, gráficos e, acima de tudo, tarefas realizadas pelo usuário podem requerer diferentes graus de responsividade e pré-computação.

Por exemplo, mapas pensados para exploração requerem tempos de resposta menores e um uso mais frequente e otimizado da transmissão progressiva do que mapas para visualização de rotas pré-definidas. Dessa forma, os benefícios da implementação de técnicas de transmissão progressiva não possuem delimitações claras mas, na verdade, estendem-se sobre um espectro que abarca desde aplicações cuja progressividade não agrega valor à experiência dos usuários até aquelas em que o envio progressivo de dados é desejável para seu uso.

Em muitos casos, respostas aproximadas são suficientes para a tomada de decisão. Ao considerar a resolução das telas em que a visualização será renderizada, o resultado exato possivelmente não precisará sequer ser computado pois pequenas cotas de erro já não seriam discerníveis pela visão humana em tais condições. Ou ainda, a exemplo do modo semi-automático descrito por Li e Ma (2019), caso uma

inspeção mais acurada seja realizada, uma nova requisição pode ser feita ao servidor pelo cliente, reduzindo a cota de erro tolerável e assim sucessivamente até obter o resultado exato ou com precisão suficientemente alta [27].

Finalmente, as técnicas de transmissão progressiva podem ser utilizadas com outros tipos de otimização, como *caching*. Seja para os diferentes níveis de detalhe ou para as diferentes cotas de erro, o resultado do processamento do lado do servidor pode ser armazenado de forma a reduzir o tempo de resposta para requisições futuras semelhantes. Adicionalmente, níveis de detalhe mais grosseiros podem utilizar menos dados, valendo-se apenas daqueles dados necessários para consultas naquele nível. Essa abordagem é utilizada, por exemplo, nas aplicações de mapas já citadas, em que consultas com um nível de zoom da escala de países e continentes não demandam o carregamento e a renderização de dados de ruas e outras informações que não são relevantes naquele contexto. Isso permite uma pré-filtragem que reduz o recorte dos dados a serem processados em outras etapas do *pipeline* de transmissão e renderização progressivas, conferindo maior agilidade à ferramenta.

Portanto, a tradução de dados brutos em visuais eficazes não é apenas uma questão técnica, mas também um exercício de design e compreensão das necessidades do público-alvo. É um campo que exige a integração de conhecimentos em ciência de dados, computação, design de interfaces e até mesmo psicologia cognitiva, para garantir que as informações sejam apresentadas de forma clara, interativa e impactante.

Capítulo 3

Estratégias para visualização progressiva de séries temporais

Com base nos insumos fornecidos pela análise da literatura realizada, é possível propor uma nova arquitetura cliente-servidor a ser implementada por um sistema de visualização de dados que incorpora os benefícios do paradigma progressivo, buscando reduzir o tempo de espera para resposta a requisições do usuário e viabilizando uma interação mais fluida. Esta arquitetura baseia-se na geração de aproximações da série temporal original em um curto intervalo de tempo e de maneira suficientemente acurada para permitir uma tomada de decisão básica pelo usuário. Ademais, podemos organizar as diferentes técnicas aplicáveis a esse contexto em uma taxonomia esboçada a seguir.

3.1 Abordagem sem progressividade

O método mais facilmente implementado e que serve de caso base para nossa análise é a agregação de todas as linhas da base de dados pelo servidor e envio da série temporal gerada para o cliente. Os pontos da série podem ser representados e transmitidos pela rede como pares de coordenadas, no caso bidimensional, ou, tendo em vista que os dados podem ser multivariados, com várias coordenadas ou um vetor multidimensional no formato $(t, x_1, x_2, ..., x_n)$, sendo t o tempo. Essa implementação "ingênua" ainda não incorpora nenhuma técnica para otimização do tempo de resposta ou dos recursos envolvidos.

Apesar de capturar todos os padrões e nuances presentes dos dados devido ao uso do conjunto em sua totalidade, esse é também o método mais computacionalmente custoso, levando a tempos de espera mais elevados em seu processamento. Isso inviabiliza seu uso em ferramentas de visualização exploratórias e interativas, com entradas frequentes do usuário, que exploram um grande conjunto de dados, como

aplicações relacionadas a *big data*.

Ainda assim, espera-se que o carregamento progressivo da visualização gradativamente convirja para o resultado exato e idêntico àquele computado sem técnicas de progressividade. Considerando um servidor com suporte a paralelismo, é possível, por exemplo, implementar a progressividade ao manter a geração da série temporal exata sendo executada em paralelo enquanto prévias da série temporal são estimadas e enviadas assincronamente através das técnicas a serem discutidas a seguir.

3.2 Progressividade por amostragem

As técnicas de amostragem consistem na seleção de um recorte dos dados para ser utilizado no cálculo de uma série temporal aproximada. Em seguida, as agregações são computadas apenas sobre esse subconjunto reduzido de dados, ou amostra. Isso permite uma drástica redução do tempo de processamento no lado do servidor, especialmente em bases de dados muito volumosas, ao evitar a necessidade de processar a totalidade dos dados. A progressividade por amostragem é particularmente útil em cenários onde a velocidade de resposta é crucial, como em sistemas interativos de visualização de dados.

Uma das principais vantagens do uso de amostragem é sua facilidade de implementação e flexibilidade em incorporar diferentes algoritmos de seleção da amostra. A abordagem mais simples consiste em uma amostragem puramente randômica sobre todas as linhas do conjunto de dados. Embora essa técnica seja muito eficiente em termos de tempo de execução, existem grandes limitações em termos de representatividade, especialmente em conjuntos de dados com distribuições não uniformes.

Para superar tais complicações, outras técnicas mais avançadas podem ser empregadas. A amostragem agrupada, por exemplo, organiza os dados em grupos baseados em intervalos de tempo, como meses ou dias, antes de realizar a seleção randômica. Essa abordagem garante uma cobertura mais uniforme do período analisado, reduzindo o risco de se observar lacunas temporais na série gerada, e é particularmente útil em séries temporais em que a continuidade dos dados é essencial para a análise, como em estudos de padrões sazonais. Além disso, a progressividade por amostragem pode ser combinada com outras técnicas, como métodos de interpolação para o preenchimento de eventuais descontinuidades nos dados amostrados.

3.3 Progressividade por transformadas

Outra maneira de aproximar uma série temporal utilizando menos recursos computacionais envolve a aplicação de transformadas e suas inversas. Assim, podemos

levar a série temporal a um domínio mais adequado, que permita seu processamento e transmissão de maneira mais eficiente.

Uma possível transformada a ser utilizada é a transformada de Fourier, definida como $\mathcal{F}(f(t)) = \int_{-\infty}^{\infty} f(t)e^{-i\omega t}dt$, sendo ω real, capaz de transformar o domínio do tempo em um domínio da frequência. Neste caso, não buscamos reduzir o volume de dados a ser agregado para gerar os valores da série temporal, mas sim reduzir a forma de descrever a série temporal em si. Por isso, esta abordagem é especialmente útil em séries com numerosos intervalos de tempo – por exemplo, séries com valores para cada dia de um recorte de várias décadas – e em situações em que a rede represente um gargalo para a comunicação entre cliente e servidor e, consequentemente, para a transmissão dos dados. Enquanto na abordagem anterior buscamos reduzir o tempo de processamento do servidor mas seguimos enviando a mesma quantidade de informação pela rede, no caso da compressão por transformadas de Fourier alcançamos uma redução no tamanho da informação transmitida pela rede.

A implementação baseada em transformadas de Fourier consiste em realizar a decomposição harmônica da série temporal, expressando-a no domínio das frequências. Qualquer série temporal integrável pode ser expressa por uma combinação de senoides de diferentes frequências e amplitudes, que correspondem às componentes harmônicas do sinal, ou seja, múltiplos inteiros de uma frequência fundamental. Esse processo é também chamado de decomposição da série temporal em suas frequências. Analogamente, em muitos casos, tal processo pode ser perfeitamente revertido e, a partir de um conjunto de amplitudes e frequências, podemos obter a série temporal original, sem introduzir qualquer erro.

É importante destacar que, em sinais não periódicos ou que apresentam descontinuidades, a reconstrução pode ser exata somente com um número infinito de coeficientes, sendo impossível em uma aplicação real. A soma de um número finito de senos e cossenos pode levar à introdução de artefatos, como o chamado "fenômeno de Gibbs" ou o "vazamento espectral" (em inglês, *"spectral leakage"*), que podem requerer outras técnicas matemáticas para sua mitigação [38]. Ainda assim, no entanto, é possível alcançar um número de componentes suficiente de tal forma que o erro inserido na série reconstruída em relação à original seja muito pequeno. Em especial, pode ser arbitrariamente baixo a ponto de não ser percebido pelo olho humano na visualização, dada a resolução da tela.

Na arquitetura proposta, a série temporal computada pelo servidor, após a agregação dos dados multivariados seja em conjunto ou não com as técnicas de amostragem já discutidas, é decomposta através de um algoritmo de transformada rápida de Fourier (FFT, sigla em inglês para *Fast Fourier Transform*), uma versão do algoritmo baseado em decomposição de matrizes e que reduz a complexidade assintótica da computação da transformada de $O(n^2)$ para $O(n\log(n))$ [39]. A FFT é

um algoritmo capaz de computar uma transformada de Fourier numericamente de forma muito rápida, dado que tanto a função original quanto sua transformada sejam discretizadas (conjunto de técnicas conhecidas como "Transformada de Fourier Discreta" ou DFT, do inglês *"Discrete Fourier Transform"*. Assim, a transformada de Fourier é expressa como

$$X_k = \sum_{n=0}^{N-1} x_n e^{-2\pi i k \frac{n}{N}}, \tag{3.1}$$

para uma sequência $\langle x_0 \dots x_{N-1} \rangle$ de comprimento N, e tem como inversa

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{2\pi i k \frac{n}{N}}, \tag{3.2}$$

conforme descrito por Cooley e Tukey (1965) no chamado algoritmo Cooley-Tukey [40].

A saída do algoritmo é uma lista de amplitudes e frequências que descrevem a série original, chamados de coeficientes. Em vez das coordenadas x e y dos pontos que compõem a série temporal, são transmitidos os coeficientes calculados do servidor para o cliente. No lado cliente, é executada a inversa da transformada rápida de Fourier, que toma como entrada a lista de coeficientes e reconstrói a série temporal para ser renderizada e exibida ao usuário.

Uma consequência interessante e de grande utilidade para este estudo é o fato de que as senoides com maiores amplitudes são as que descrevem o comportamento mais geral da série, como tendências e sazonalidades, enquanto as de menor amplitude são menos significativas para a expressão da série, exprimindo desvios menores e resíduos. Assim, uma vez realizada a decomposição da série em suas frequências, podemos descartar as componentes de menor amplitude e transmitir pela rede, do servidor ao cliente, apenas as mais representativas. Ao ser reconstruída pelo cliente, a série obtida é uma aproximação próxima da série original.

Como uma alternativa à abordagem de Fourier, as transformadas de *wavelet* oferecem uma análise mais sofisticada para séries temporais que exibem características de relevância local [41]. Enquanto a transformada de Fourier decompõe um sinal em senoides de duração infinita, perdendo toda a informação sobre a localização temporal dos eventos de frequência, a transformada de *wavelet* utiliza funções de curta duração que são escalonadas e transladadas. Isso permite uma análise localizada em um ponto intermediário do espectro tempo-frequência, capturando não só "quais" frequências estão presentes, como também "quando" elas ocorrem.

Assim como as FFT utilizam uma combinação de senoides de diferentes frequências, a transformada de *wavelet* utiliza uma combinação de uma função pré-definida

chamada *wavelet-mãe* $\psi(t)$ com diferentes translações no tempo e em diferentes escalas, calculadas por

$$\psi_{a,b} = \frac{1}{\sqrt{a}}\psi(\frac{t-b}{a}) \tag{3.3}$$

sendo $a > 0$ o parâmetro de escala e $-\infty < b < \infty$ o parâmetro de deslocamento no tempo.

A *wavelet-mãe* escolhida, desde que atenda o requisito de ter energia finita, pode possuir diferentes formatos e propriedades, aderentes a diferentes tipos de necessidade e aplicação, pertencendo às chamadas famílias de *wavelets*. Uma das mais conhecidas é a família de *wavelets* de Daubechies, amplamente utilizada para compressão de imagens devido a sua ortogonalidade (a função e sua transformada são sempre ortogonais entre si) e por maximizar o número de *vanishing points*, ou seja, de pontos em que a função assume o valor 0. A notação "dbN" refere-se à função dessa família com N pontos desse tipo.

Na prática, para fins computacionais, utiliza-se a transformada de *wavelet* discreta (conhecida, em inglês, por "DWT"). Através dela, o sinal é decomposto em múltiplos níveis, gerando em cada um deles coeficientes de "aproximação" (de baixa frequência, representando a tendência geral) e de "detalhe" (de alta frequência, representando as variações rápidas e localizadas). No contexto da visualização progressiva, essa decomposição hierárquica é particularmente vantajosa. Pode-se transmitir inicialmente apenas os coeficientes de aproximação do nível mais grosseiro para renderizar uma versão de baixa resolução da série. Subsequentemente, os coeficientes de detalhe dos níveis sucessivos são enviados para refinar progressivamente a visualização, adicionando nuances e eventos localizados até que a representação completa seja alcançada.

A DWT demonstra uma notável eficiência computacional. Ela oferece uma complexidade de tempo que se aproxima de $O(N)$, dado que o tamanho do filtro seja muito menor que o tamanho do sinal), o que a distingue favoravelmente em comparação com a transformada rápida de Fourier (FFT), cuja complexidade é de $O(NlogN)$. Enquanto a FFT, que emprega as mesmas funções de base da transformada de Fourier discreta (DFT), realiza uma divisão de frequência com intervalos uniformes, a DWT adota uma divisão logarítmica. Tal escolha confere à DWT uma vantagem significativa em termos de recursos computacionais necessários para o processamento de grandes volumes de dados.

3.4 Progressividade por aprendizado de máquina

Uma terceira abordagem para a geração de séries temporais acuradas e de maneira rápida para dados multivariados e volumosos consiste na utilização de técnicas de aprendizado de máquina, parte do campo do conhecimento de Inteligência Artificial (IA). Cada vez mais discutida e em constante e extremamente veloz evolução, a IA abarca um conjunto de conceitos relacionados à capacidade de detectar padrões e se adaptar programaticamente. É possível valer-se dessa valiosa ferramenta para otimizar o processamento e a transmissão de dados de séries temporais para serem visualizados.

Em particular, redes neurais são modelos computacionais formados por uma ou mais camadas de "neurônios", aludindo (ainda que apenas conceitualmente) ao funcionamento do sistema nervoso de seres vivos. Esses neurônios recebem como entrada as saídas de neurônios da camada anterior, multiplicando-as por diferentes pesos, somando-as e aplicando uma função denominada "função de ativação", como a ReLU (do inglês, *Rectified Linear Unit* ou Unidade Linear Retificada) definida como $f(x) = max(0, x)$ e uma das mais populares utilizadas para tal fim. Assim, redes neurais podem ser compreendidas como abstrações de intricados sistemas algébricos com capacidades mais robustas de identificar e replicar padrões.

Nesta arquitetura, será realizado de maneira prévia o treinamento de uma rede neural que recebe como entrada a variável a ser visualizada e a expressão a ser aplicada como filtro dos dados e fornece como saída os pontos da série temporal. Como parte do treinamento, ocorrido na inicialização do servidor, são geradas várias combinações possíveis de entradas e computadas as respectivas séries temporais exatas. Após uma sequência de ciclos de treinamento, também chamados de épocas, a rede neural torna-se capaz de criar rapidamente séries temporais aproximadas para futuras entradas.

Essa abordagem, devido a sua capacidade de abstração e detecção de padrões mais sofisticada e aplicável, permite a identificação de correlações nos dados que poderiam passar de maneira despercebida por seres humanos. Isso pode garantir que sazonalidades, tendências e *features* pontuais apareçam corretamente nas séries temporais em diferentes filtros e recortes aplicados pelo usuário. Ademais, o uso de redes neurais permite uma outra otimização: os parâmetros (pesos e *bias*) das diversas conexões da rede podem ser transmitidos uma última vez do servidor ao cliente, que pode reconstruir a rede neural localmente. Dessa forma, as séries temporais estimadas podem ser geradas inteiramente pelo cliente, minimizando ainda mais atrasos ocasionados pela latência da rede. No entanto, é importante considerar a necessidade de mais recursos computacionais do lado do servidor para o treinamento inicial.

A escolha do número de nós em uma rede neural, que determina sua sofisticação, apresenta vantagens e desvantagens que impactam diretamente o treinamento, a transmissão dos coeficientes e a complexidade da reconstrução. Redes neurais mais sofisticadas, com um maior número de nós e pesos, têm a capacidade de modelar relações complexas e capturar padrões sutis nos dados. Essa complexidade pode resultar em um desempenho superior em tarefas de previsão, especialmente em cenários onde os dados apresentam variabilidade significativa.

Entretanto, o aumento no número de pesos também implica em um maior tempo de treinamento. Redes mais complexas demandam mais iterações para convergir, o que pode resultar em um custo computacional elevado e em um tempo de treinamento mais longo. Além disso, a necessidade de um conjunto de dados maior e mais diversificado para evitar *overfitting* se torna evidente, uma vez que redes mais profundas tendem a se ajustar excessivamente aos dados de treinamento se não forem devidamente regularizadas.

Overfitting é um fenômeno ocorrido quando um modelo de aprendizado de máquina se ajusta excessivamente aos dados de treinamento, capturando não apenas os padrões gerais como desejado, mas também os ruídos e variações específicas desses dados. Isso faz com que o modelo tenha um desempenho inferior ao ser aplicado a novos dados, pois ele não generaliza bem para situações fora do conjunto de treinamento. Em outras palavras, o modelo se torna altamente especializado nos dados de treinamento, mas perde a capacidade de lidar com dados não vistos, comprometendo sua eficácia em cenários reais.

Ademais, a transmissão dos coeficientes de uma rede neural mais sofisticada pode acarretar um aumento no tempo de transmissão. O envio de um maior número de pesos do servidor ao cliente pode resultar em latências significativas, especialmente em ambientes com largura de banda limitada. Essa questão é crítica em aplicações em tempo real. Em contrapartida, redes neurais menos sofisticadas, com um número reduzido de pesos, apresentam vantagens em termos de velocidade de treinamento e transmissão. O tempo necessário para treinar a rede é menor, e a quantidade de dados a ser transmitida é reduzida, o que pode ser benéfico em cenários onde a latência é uma preocupação. No entanto, essa simplicidade pode vir à custa da capacidade de abstração, resultando em um desempenho inferior em tarefas que exigem uma compreensão mais profunda dos dados.

A complexidade da reconstrução também é afetada pelo número de pesos. Redes mais complexas podem exigir um processamento mais intensivo para a reconstrução das estimativas, enquanto redes mais simples podem ser mais rápidas e eficientes nesse aspecto. Esse processamento ocorre justamente no lado do cliente, em que geralmente existem menos recursos computacionais disponíveis se comparado com o servidor.

Portanto, a escolha entre redes mais ou menos sofisticadas deve ser cuidadosamente ponderada, levando em consideração o equilíbrio entre a capacidade de modelagem, o tempo de treinamento, a latência na transmissão e a complexidade da reconstrução, de modo a atender às necessidades específicas da aplicação em questão.

Capítulo 4

Arquitetura e solução experimental

No escopo deste trabalho, foram escolhidas três técnicas de transmissão progressiva para serem implementadas e testadas utilizando um grande volume de dados de natureza temporal a serem visualizados em um navegador web. Dessa forma, busca-se realizar uma comparação entre os diferentes métodos quanto ao tempo de espera para o carregamento dos resultados parciais e totais, suscitando uma discussão das principais vantagens e desvantagens de cada um deles.

A escolha dessas três técnicas de transmissão progressiva para serem implementadas e testadas neste trabalho foi motivada pela necessidade de explorar diferentes abordagens para resolver o problema de implementar a progressividade e reflete a intenção de cobrir um espectro amplo de possibilidades. Cada técnica representa um modo distinto de abordar os desafios associados à visualização progressiva com base em proposições semelhantes existentes na literatura da área, permitindo uma análise comparativa que contribui para identificar as vantagens e limitações de cada solução. Assim, ilustram-se melhorias de tempo e acurácia possíveis feitas no processamento do lado do servidor, otimizações na transmissão dos dados através da rede e aplicações de aprendizado de máquina e inteligência artificial para o atingimento do comportamento progressivo por meio de aproximações geradas no lado do cliente.

Essa diversidade de abordagens é essencial para compreender como diferentes estratégias podem impactar aspectos como tempo de resposta, qualidade das estimativas e experiência do usuário. Ao testar modos variados de endereçar o problema, busca-se, além de avaliar a eficácia de cada técnica, identificar oportunidades de combinação ou aprimoramento que possam resultar em soluções mais robustas e eficientes, como ilustrado na Figura 4.1.

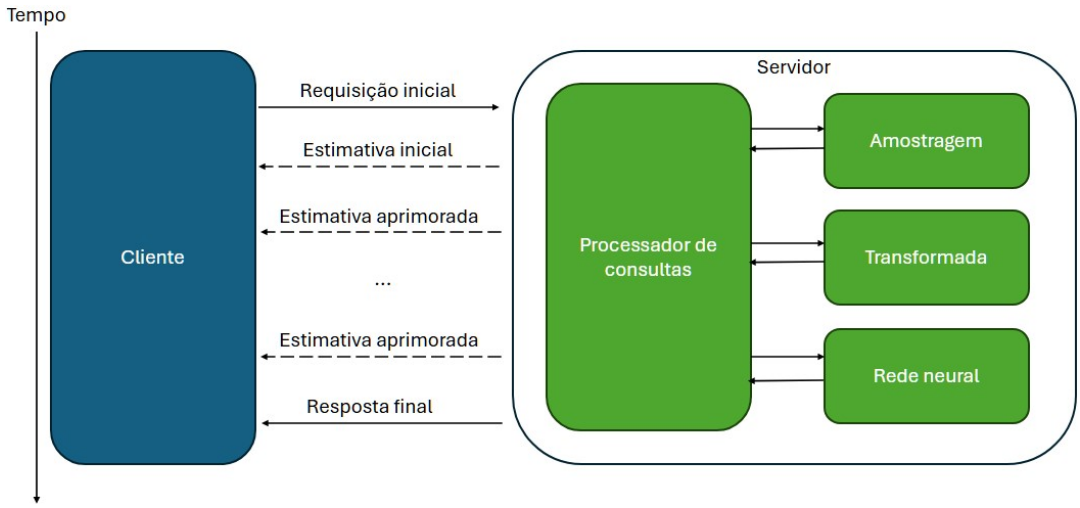


Figura 4.1: Diagrama da arquitetura proposta

4.1 Soluções para transmissão assíncrona de dados

Oficialmente proposta em 2003 e atualizada em 2010, a *Web Server Gateway Interface* (WSGI) é uma interface padronizada entre servidores e aplicações web desenvolvidos na linguagem de programação Python [42] [43], muito empregada nesse tipo de aplicação devido ao seu grande ecossistema de bibliotecas e módulos publicamente disponíveis e constantemente mantidos pela comunidade crescente e ativa de desenvolvedores ao seu redor. Criada com o propósito de padronizar a comunicação entre servidores e aplicações por eles executadas, a WSGI permitiu que aplicações não precisassem mais ser desenvolvidas para um cliente em específico, contribuindo para uma maior flexibilidade e interoperabilidade na implantação em ambiente de produção e uma implementação mais agnóstica e descomplicada.

No entanto, por design, o modelo WSGI prevê o uso de requisições síncronas, bloqueando a conexão até o envio de uma resposta. Além disso, essa interface foi pensada para conexões simples e curtas, tratadas uma de cada vez, dificultando a implementação de *streaming* de dados. Devido à sua idade, a WSGI também não possui suporte para tecnologias mais recentes que poderiam ser úteis no contexto deste trabalho, como, por exemplo, a tecnologia *WebSocket* [44].

Assim, em 2018, surge o conceito de *Asynchronous Server Gateway Interface* (ASGI), proposta como sucessora da WSGI [45]. Especialmente aplicável para projetos baseados em requisições assíncronas, a ASGI conta com suporte a WebSockets, é facilmente escalável e é definida a partir de um escopo, uma função *async send* para envio de mensagens ao cliente e uma função *async receive* para recebimento de mensagens vindas do cliente.

Neste trabalho, a implementação foi realizada utilizando a interface ASGI, especificamente através do servidor Uvicorn (versão 0.27.0), devido à sua capacidade

de lidar eficientemente com requisições assíncronas, característica fundamental para o desenvolvimento de um sistema de visualização progressiva. A escolha do ASGI em detrimento do WSGI se justifica pela necessidade de manter múltiplas conexões simultâneas e permitir a transmissão contínua de dados parciais ao cliente sem bloqueios, como evidenciado no Código-Fonte 4.1.

O funcionamento da transmissão assíncrona permite um padrão de comunicação em que o servidor processa parcialmente os dados solicitados e os envia ao cliente sem esperar pela conclusão total do processamento. De tal forma, o cliente pode começar a renderizar os dados assim que os primeiros resultados são recebidos, melhorando significativamente a experiência do usuário em comparação com abordagens síncronas, utilizando JavaScript com capacidades de manipulação de *streams*, permitindo o processamento e a renderização dos dados à medida que chegam.

Para viabilizar a avaliação empírica do protótipo também no lado cliente, foi desenvolvida uma aplicação web de apoio construída com HTML5, CSS3 e JavaScript ES6. A visualização das séries temporais geradas pelos diferentes algoritmos foi implementada utilizando a biblioteca D3.js (*Data-Driven Documents*) em sua versão 7.8.0, escolha justificada por sua capacidade de manipulação eficiente de estruturas baseadas em dados e pelo conjunto robusto de primitivas de visualização que oferece. Tal biblioteca permitiu a renderização dinâmica e atualização progressiva dos gráficos de séries temporais à medida que novos dados eram recebidos do servidor, implementando diferentes comportamentos para cada técnica analisada.

A ferramenta funciona como um painel de experimentação: o usuário pode selecionar qualquer um dos *endpoints* expostos pela API e, em seguida, definir interativamente os parâmetros de cada algoritmo (percentual de amostra, número de coeficientes etc.). Além da interface gráfica, o código inclui funções *wrapper* JavaScript que encapsulam as chamadas REST e as rotinas de reconstrução (por exemplo, expansão do espectro via simetria conjugada ou inferência dos pesos da rede). As funções *wrapper* expõem interfaces padronizadas que abstraem as diferenças entre os algoritmos subjacentes, permitindo que desenvolvedores adotem facilmente qualquer uma das técnicas sem necessidade de conhecimento aprofundado sobre sua implementação interna. Dessa forma, a aplicação web cumpre duas finalidades: ser um ambiente controlado de experimentação sistemática para este trabalho e oferecer um kit básico reutilizável para adoção das estratégias progressivas em cenários reais.

Código-Fonte 4.1: Trecho do código-fonte para instanciação do servidor ASGI com 4 processos concorrentes e utilizando *asyncio*

```
import uvicorn
from fastapi import FastAPI
app = FastAPI()
if __name__ == "__main__":
```

```
uvicorn.run(app, host="0.0.0.0", port=8000, workers=4,
            loop="asyncio", timeout_keep_alive=120)
```

Essa abordagem assíncrona forma a base sobre a qual as APIs de visualização progressiva foram desenvolvidas, permitindo a criação de *endpoints* específicos para cada técnica de processamento progressivo implementada.

4.2 Desenvolvimento de APIs

Para a requisição dos dados a serem visualizados, foi construída uma *Application Programming Interface* (API), estabelecendo rotas de modo que o cliente possa estabelecer uma conexão com o servidor de dados através de um *Uniform Resource Locator* (URL) que contém os parâmetros necessários para a descrição da consulta desejada. Cada tipo de "caminho" que permite uma determinada consulta ao servidor é chamado de *endpoint* e cada *endpoint* pode acionar diferentes lógicas para a geração da resposta pelo servidor.

Uma API é um conjunto de regras e protocolos que permite que diferentes sistemas de software se comuniquem e interajam entre si. Os recursos disponibilizados através da API podem ser consumidos por qualquer outro programa através da Internet, desde um executável programado para esse fim ou diretamente através do navegador web, por exemplo.

Neste caso, foram definidos diferentes métodos que extraem a função de agregação e os filtros aplicados sobre os dados diretamente da URL, processam e compõem a resposta a ser enviada de volta. Já o consumo dessa API é feito por meio de uma página web que realiza as requisições e as exibe na tela utilizando HTML e JavaScript.

Em um primeiro momento, no primeiro *endpoint* criado, a API foi implementada de forma que o cliente seja responsável por fornecer um número entre 0 e 1 que represente o grau de precisão desejado para a resposta. Esse valor é utilizado como parâmetro do algoritmo de amostragem randômica utilizado para gerar uma resposta aproximada.

Subsequentemente, foram criados diferentes *endpoints*, dedicando um para cada técnica de compressão e transmissão implementadas. Para cada *endpoint*, seus próprios parâmetros foram definidos de acordo com o algoritmo por ele acionado.

Todas as respostas geradas pelo servidor são estruturadas em formato JSON (*JavaScript Object Notation*) antes de serem serializadas e transmitidas. Isso permite a reconstrução do JSON original no lado do cliente e mantém os diferentes atributos e aninhamentos do objeto transmitido.

Por exemplo, para requisições de aproximações da série temporal calculadas por amostragem simples, pode ser utilizada a URL:

```
http://127.0.0.1:8000/sample/N1/mean/10/N2<40
```

Nela, observa-se os parâmetros abaixo descritos:

- *sample*: seleciona algoritmo de amostragem simples;
- $N1$: seleciona atributo N1 (primeira variável normal) do conjunto de dados sintético;
- *mean*: seleciona agregação por média aritmética;
- 10: seleciona tamanho da amostra de 10% da quantidade total de linhas do conjunto de dados;
- $N2 < 40$: aplica filtro nas linhas do conjunto de dados, selecionando apenas casos em que variável N2 é menor que 40. (Na prática, é transmitida como "$N2\%3C40$" devido à codificação dos caracteres em UTF-8 realizada automaticamente pelo navegador.)

A resposta transmitida pelo servidor é gerada no formato a seguir:

```
"[{\"date\":1325376000000,\"N1\":74.4496948365},
{\"date\":1325462400000,\"N1\":72.9055368621},
{\"date\":1325548800000,\"N1\":75.21584396},
..."
```

com uma *timestamp* representando o número de segundos decorridos desde a *epoch* Unix.

A principal justificativa para a opção pelo formato JSON (JavaScript Object Notation) para a serialização dos dados reside na sua nativa e direta compatibilidade com o ambiente de execução do cliente, desenvolvido em JavaScript. Embora formatos binários como o Protocol Buffers (Protobuf) ofereçam uma redução significativa no volume de dados transmitidos e maior velocidade de *parsing*, sua adoção implicaria em uma camada adicional de complexidade, tanto no servidor quanto no cliente [46]. Seria necessária a definição de esquemas (*schemas*) pré-estabelecidos e a inclusão de bibliotecas específicas no lado do cliente para a decodificação dos dados.

A capacidade de converter a resposta da API em um objeto manipulável com uma única instrução (como `response.json()`) simplifica sobremaneira o desenvolvimento no lado do cliente, eliminando a necessidade de bibliotecas de serialização adicionais. Essa sinergia é ainda mais evidente ao se considerar que grande parte das bibliotecas de visualização de dados para a web, incluindo a D3.js utilizada neste

projeto e outras como Vega-Lite, são projetadas para consumir dados no formato JSON de maneira otimizada.

Ademais, a natureza textual e legível do JSON oferece vantagens significativas durante as fases de desenvolvimento e depuração, permitindo a inspeção direta das respostas nas ferramentas de desenvolvedor do navegador por sua melhor legibilidade humana. Embora formatos binários ofereçam maior compactação, considerou-se que a simplicidade de implementação e a interoperabilidade no ecossistema web eram mais prementes para o escopo deste trabalho. Vale ressaltar, ainda, que a desvantagem da verbosidade do JSON pode ser, em grande parte, mitigada pelas técnicas de compressão aplicadas transparentemente pelo próprio protocolo HTTP (como Gzip ou Brotli), que reduzem significativamente o volume de dados efetivamente trafegado na rede [47].

4.3 Uso prático

Antes de adentrar na análise quantitativa das métricas de desempenho, é fundamental contextualizar como cada uma das estratégias de progressividade — amostragem, transformadas e aprendizado de máquina — seria implementada em um cenário de produção e como isso impactaria a experiência do usuário de um sistema de visualização de dados.

Amostragem: No que concerne às abordagens por amostragem, sua execução ocorre majoritariamente em tempo real, após a requisição do usuário. Porém, pode haver ou não um processo anterior de pré-computação em preparação para a consulta do usuário, poupando tempo na geração da resposta em troca de um tempo de inicialização maior uma única vez pelo servidor quando realizado o carregamento inicial do conjunto de dados a ser oferecido para visualização. Não há, para os métodos de amostragem mais simples, necessidade de qualquer pré-processamento ou treinamento prévio, o que lhes confere máxima flexibilidade. Em contrapartida, para as técnicas baseadas em agrupamento, como KNN, DBSCAN e OPTICS, seus *clusters* podem já se encontrar pré-computados, evitando que seja executado "on-the-fly" sobre o subconjunto de dados resultante da consulta do usuário.

Transformadas: A estratégia fundamentada em transformadas matemáticas, como FFT e *wavelets*, adota um fluxo de trabalho distinto, cujo benefício primário não é a redução do tempo de processamento no servidor, mas sim a drástica redução do volume de dados transmitido pela rede. Ao receber uma consulta, o servidor realiza a agregação completa para calcular a série temporal exata. Somente então, aplica a transformada sobre esta série completa, enviando

ao cliente apenas um subconjunto dos coeficientes mais significativos para a reconstrução de uma versão aproximada.

Redes neurais: Por fim, a utilização de redes neurais representa o paradigma mais complexo em termos de preparação, mas que oferece a experiência interativa mais fluida. Este paradigma introduz uma dicotomia clara entre um custo computacional inicial e um desempenho em tempo real. A etapa de pré-processamento consiste em um treinamento intensivo e offline do modelo, o que pode demandar de dezenas de segundos a minutos, a depender da complexidade da rede e dos dados. Uma vez treinado, contudo, o tempo de inferência para gerar uma série estimada a partir dos parâmetros de uma nova consulta é extremamente baixo, da ordem de milissegundos. Na prática, isto se desdobra em duas possíveis implementações: a inferência pode ocorrer no servidor, com respostas quase instantâneas, ou, de forma mais avançada, os parâmetros do modelo podem ser enviados ao cliente uma única vez. Embora isso represente um download inicial de volume considerável, todas as consultas subsequentes são processadas localmente no navegador, eliminando a latência de rede e permitindo uma exploração de dados com fluidez máxima.

Portanto, a seleção da estratégia ideal depende intrinsecamente do caso de uso e dos recursos disponíveis. A amostragem oferece flexibilidade e rapidez para explorações iniciais; as transformadas mostram-se superiores em cenários com restrições de rede; e as redes neurais, apesar do custo inicial elevado, justificam-se em ambientes de análise intensiva, nos quais o investimento em treinamento é amortizado por uma interatividade superior em sessões de exploração prolongadas. Já estratégias que envolvem pré-processamento, como algoritmos de amostragem baseados em agrupamento ou o treinamento de redes neurais, mostram-se particularmente vantajosas para conjuntos de dados de natureza estática, cujas atualizações não ocorrem em tempo real ou com alta frequência.

Convém ressaltar que a viabilidade de cada uma das estratégias discutidas é fortemente influenciada pela natureza do conjunto de dados, especificamente se é estático (imutável) ou dinâmico (sujeito a modificações e atualizações com o passar do tempo). As estratégias que dependem de um pré-processamento intensivo, como o treinamento de redes neurais, demonstram sua máxima eficácia em cenários com dados estáticos. Nestes casos, o custo computacional elevado é incorrido uma única vez e amortizado por múltiplas consultas subsequentes de alta performance. Em um ambiente com dados dinâmicos, como em visualizações que devem refletir dados gerados em tempo real, tais abordagens tornam-se menos práticas, uma vez que o modelo, por exemplo, se tornaria obsoleto, exigindo recálculos frequentes e computacionalmente dispendiosos.

Por outro lado, técnicas que operam puramente em tempo real, como a amostragem simples, são inerentemente mais adaptáveis a dados dinâmicos. Para o caso de amostragens baseadas em agrupamentos, como o KNN, o uso de dados mais voláteis pode ser bem-sucedido, tendo em vista que novos dados podem ser classificados de acordo com os *clusters* previamente formados e adicionados ao respectivo agrupamento mais próximo através de um processo de inferência. No entanto, pode ser necessário realizar o recálculo dos agrupamentos com o passar do tempo. A adição de novos dados ao conjunto pode alterar a distribuição estatística do conjunto em um fenômeno conhecido como *concept drift* [48]. Assim, os *clusters* originais podem deixar de ser representativos e a classificação de novos pontos pode começar a perder precisão, tornando um recálculo periódico inevitável.

4.4 Arcabouço experimental

O objetivo central desta fase experimental é investigar, de forma sistemática, o desempenho e as características das diferentes estratégias de progressividade quando aplicadas à agregação ad-hoc de dados multivariados.

Para tanto, a avaliação foi delineada para responder às seguintes questões de pesquisa:

- Qual o impacto das diferentes abordagens progressivas no tempo de processamento do servidor necessário para gerar uma representação inicial da série temporal, em comparação com uma abordagem não progressiva?
- Como varia a acurácia das aproximações geradas por cada técnica progressiva em seus estágios iniciais e como ela converge para o resultado exato?
- Qual é o *trade-off* entre a redução no tempo de resposta e o erro de aproximação introduzido por cada técnica, considerando diferentes parametrizações de cada método, como por exemplo o percentual de amostragem, o número de coeficientes da FFT, ou a complexidade da rede neural?
- De que forma as características do conjunto de dados, como volume, cardinalidade após filtragem ou distribuição dos atributos, influenciam a eficácia relativa de cada estratégia progressiva?

Para responder a estas questões, o protótipo foi submetido a uma série de testes controlados, utilizando tanto conjuntos de dados sintéticos, com propriedades conhecidas, quanto dados reais, para aferir a aplicabilidade em cenários práticos. As principais variáveis de desempenho e qualidade mensuradas em cada experimento incluem:

- Tempo de Processamento no Servidor: Aferido para quantificar a carga computacional de cada algoritmo na geração da série temporal, diretamente relacionado à experiência do usuário e à responsividade das interações.
- Erro Quadrático Médio (MSE): Utilizado como métrica principal para avaliar a fidelidade da série temporal aproximada em relação à série temporal exata, fundamental para a confiabilidade na tomada de decisão do usuário.
- Cobertura Temporal: Especificamente para as técnicas de amostragem, esta métrica avalia a proporção de intervalos de tempo distintos presentes na amostra selecionada.
- Volume de Dados Transmitidos: Particularmente relevante para técnicas como a transformada de Fourier, onde se busca reduzir a quantidade de informação enviada ao cliente.

As seções subsequentes detalharão a configuração do ambiente experimental, os conjuntos de dados utilizados, as especificidades da implementação das APIs para cada técnica e, finalmente, apresentarão e discutirão os resultados obtidos para cada abordagem, culminando em uma análise comparativa e na identificação das limitações do estudo.

A avaliação de desempenho temporal das diferentes estratégias implementadas no servidor considerou, primariamente, o tempo de processamento da CPU. Para esta finalidade, os tempos foram medidos através do método `time.process_time()` da linguagem de programação Python. Conforme sua especificação, este método permite contabilizar somente o tempo decorrido em que o processador está ativamente executando as instruções do programa, desconsiderando períodos em que o processo aguarda operações de leitura e escrita ou é colocado em espera pelo escalonador do sistema operacional. A escolha por esta métrica visou isolar a eficiência computacional intrínseca de cada algoritmo, minimizando a influência de fatores externos à lógica implementada, como a carga de outros processos em execução na máquina de testes, permitindo assim uma comparação mais estável e reprodutível do custo de processamento puro no *backend*.

No entanto, reconhece-se que, em uma arquitetura cliente-servidor como a explorada neste trabalho, a métrica que melhor reflete a experiência do usuário é o tempo total decorrido (também conhecido como tempo de parede ou *wall-clock time*) entre a submissão de uma interação e a efetiva apresentação da resposta visual. Este tempo total engloba não apenas o processamento no servidor (`time.process_time()` acrescido de esperas, leituras e escritas), mas também as latências de transmissão de dados pela rede e o tempo de processamento e renderização no lado do cliente. Embora uma análise detalhada do tempo de resposta ponta a ponta em diferentes cenários

de rede e configurações de cliente não tenha sido o foco primário dos experimentos comparativos entre os algoritmos de servidor aqui descritos, que visavam otimizar especificamente o componente de processamento de dados no *backend*, a importância do tempo total é indiscutível para a avaliação holística de sistemas interativos.

Ademais, é relevante considerar o impacto de técnicas como *prefetching* (pré-busca de dados) e *caching* (armazenamento em memória de resultados frequentemente acessados ou recentemente computados) sobre as métricas de tempo de resposta. O *prefetching* procura antecipar as necessidades do usuário, carregando dados ou mesmo resultados parciais de consultas prováveis antes de uma requisição explícita. Por sua vez, o *caching* permite que consultas idênticas ou substancialmente similares, já processadas anteriormente, sejam atendidas com latência drasticamente reduzida, ao se evitar a recomputação completa dos dados. Ambas as estratégias, quando efetivamente implementadas, podem melhorar substancialmente o tempo total percebido pelo usuário e a fluidez da interação.

É crucial notar, entretanto, que no contexto de visualização exploratória de grandes volumes de dados multivariados com agregações ad-hoc, como o foco desta dissertação, a aplicabilidade de *caching* e *prefetching* exaustivos é inerentemente limitada. O vasto número de combinações de possíveis filtros, recortes temporais e funções de agregação que o usuário pode aplicar torna inviável a pré-computação e o armazenamento em cache de todos os resultados potenciais. Embora resultados para certos níveis de detalhe ou para consultas mais frequentes possam ser estrategicamente cacheados, a natureza exploratória e dinâmica da interação do usuário frequentemente levará a consultas inéditas, cujos resultados não se encontram pré-calculados, reforçando a necessidade das técnicas de processamento progressivo e aproximação aqui investigadas para garantir a interatividade desejada.

Já o erro quadrático médio foi escolhido por permitir comparações entre resultados que envolvam erros positivos ou negativos e em séries temporais com quantidades diferentes de pontos. O cálculo do erro quadrático médio foi feito comparando cada período de tempo da série estimada pelo algoritmo com a série temporal calculada de maneira exata por meio da abordagem clássica.

Foi definida também a métrica nomeada "cobertura" como a razão entre a quantidade de intervalos de tempo (no caso destas experimentações, dias) efetivamente selecionada pela amostragem e o total de intervalos de tempo disponíveis no conjunto de dados. Por exemplo, para um conjunto de dados com dados que abrangem 1000 dias em que o algoritmo de amostragem selecionou uma amostra com dados referentes a 800 dias, teremos uma cobertura de 80%.

Essa métrica busca mensurar a capacidade de um algoritmo de capturar possíveis picos, vales e descontinuidades em uma série temporal, isto é, desvios pontuais que podem ser de significativa relevância para o usuário. Assim, um maior valor

de cobertura permite uma maior probabilidade de capturar tais variações agudas além dos padrões sazonais que já poderiam ser observados em valores de cobertura menores.

Para a implementação dos testes, foi usada a linguagem de programação Python, em sua versão 3.10.11, além das seguintes bibliotecas listadas abaixo:

- FastAPI 0.109.0, utilizada para a definição da API;
- Uvicorn 0.27.0, utilizada para a criação do servidor ASGI;
- Pandas 2.2.3, utilizada para manuseio dos conjuntos de dados;
- NumPy 2.2.1, utilizada para diferentes operações vetoriais;
- PyTorch 2.5.1+cu124, com suporte a aceleração por hardware baseada em CUDA, utilizada para criação e uso de redes neurais;
- Scikit-Learn 1.6.0, utilizada para diversas técnicas de aprendizado de máquina;
- SciPy 1.15.0, utilizada para a geração dos conjuntos de dados sintéticos.

A escolha pela linguagem Python se fundamenta em sua ampla coleção de bibliotecas e módulos disponíveis para o campo de Ciência de Dados e para o desenvolvimento de sistemas web. Seu uso permitiu um processo de codificação e experimentação mais célere.

Para cada abordagem testada, foram realizadas 100 medições e extraídas as médias e desvios-padrão para cada métrica descrita. Esse procedimento permite reduzir o impacto de eventuais variações no escalonamento dos recursos computacionais a nível do processador e no uso da memória. No caso dos algoritmos que utilizam geradores de números randômicos, em especial nas técnicas baseadas em amostragem aleatória, a escolha por utilizar a média de várias execuções do algoritmo permite reduzir eventuais vieses e inconsistências ocasionadas por acaso. O fluxo de execução de um experimento, em linhas gerais, pode ser visto em pseudocódigo em Algoritmo 1.

O código-fonte do *software* desenvolvido para a execução dos experimentos se encontra em um repositório da plataforma *GitHub* disponível no endereço https://github.com/lucasbarcellosoliveira/progressive. O código está disponível em um repositório público, permitindo que outros pesquisadores e desenvolvedores acessem, revisem e utilizem as implementações discutidas. Ademais, os arquivos disponibilizados através do repositório oferecem suporte para utilização junto à ferramenta Docker, facilitando a configuração e a execução do ambiente de desenvolvimento e tornando os experimentos mais acessíveis, replicáveis e isoláveis.

Algoritmo 1 Execução de experimento com uma dada técnica

```
tempos ← {∅}
mse ← {∅}
coberturas ← {∅}
for i ← 1 até 100 do
    t_inicio ← tempo_atual
    saida ← func_testada ()
    t_fim ← tempo_atual
    tempos_i ← t_fim − t_inicio
    referencia ← func_exata ()
    dif ← 0
    cobertura ← 0
    for j ← 1 até tamanho(referencia) do
        if j em saida then
            dif ← dif + (saida_j − referencia_j)^2
            cobertura ← cobertura + 1
        end if
    end for
    mse_i ← dif/tamanho(referencia)
    coberturas_i ← cobertura/tamanho(referencia)
end for
escreva media(tempos)
escreva desvio_padrao(tempos)
escreva media(mse)
escreva desvio_padrao(mse)
escreva media(coberturas)
escreva desvio_padrao(coberturas)
```

É pertinente ressaltar que, embora as questões de pesquisa supracitadas orientem a investigação empírica detalhada, certos comportamentos e relações de desempenho podem ser, em certa medida, antecipados com base nos princípios fundamentais das técnicas em estudo. A experimentação sistemática se justifica e se torna crucial não para constatar o evidente, mas para quantificar essas relações, explorar interações complexas e validar o comportamento das abordagens em cenários não triviais e com dados de características variadas.

Por exemplo, pode-se deduzir que a abordagem não progressiva, ao processar a integralidade dos dados para cada requisição, invariavelmente fornecerá o resultado exato, com erro quadrático médio nulo, mas também demandará maior tempo de processamento em comparação com as técnicas progressivas, especialmente ao lidar com grandes volumes de dados. Já nas técnicas baseadas em amostragem, um aumento no percentual de dados amostrados tenderá a resultar em uma aproximação mais acurada da série temporal real (menor MSE), porém implicará um custo computacional maior e, consequentemente, um tempo de resposta mais elevado no servidor. Inversamente, amostras menores serão processadas mais rapidamente, mas com potencial detrimento da fidelidade.

Para a abordagem por transformada de Fourier, espera-se que a utilização de um maior número de coeficientes na reconstrução da série temporal no cliente resulte em menor erro de aproximação, aproximando-se da série original, ao custo de um maior volume de dados a serem transmitidos e de um processamento ligeiramente maior na etapa de reconstrução. O valor da experimentação, portanto, reside não em simplesmente confirmar tais tendências gerais, mas em determinar a magnitude precisa desses efeitos, identificar os pontos ótimos de operação para diferentes contextos de dados e requisitos de aplicação (como o limiar aceitável de erro versus tempo de resposta), e, crucialmente, compreender o comportamento comparativo das diversas técnicas quando submetidas às dinâmicas de filtragem e agregações ad-hoc que caracterizam a exploração interativa de dados.

Capítulo 5

Experimentação

O presente capítulo detalha o conjunto de experimentos conduzidos para avaliar o protótipo de visualização progressiva de séries temporais, cuja arquitetura e técnicas implementadas foram descritas no Capítulo 4.

5.1 Conjunto de dados

De modo a testar as diferentes abordagens para computar e entregar as requisições solicitadas por um usuário, torna-se necessário aplicar tais técnicas sobre um conjunto de dados volumoso e multidimensional que ofereça um cenário realista comparado a uma aplicação real desse tipo. Por outro lado, visto que certas técnicas podem obter melhores resultados – em termos de valores aproximados mais acurados – para dados com certas características, para os experimentos conduzidos neste trabalho também é importante realizar testes sobre dados com diferentes distribuições em diferentes dimensões, com ordens de grandeza distintas, enviesados ou não, bem como outras variações que podem ser notadas após a implementação e realização de testes.

Por exemplo, técnicas baseadas em amostragens selecionadas randomicamente do conjunto de dados podem apresentar melhores resultados em variáveis com distribuições balanceadas, sem grande viés. Já abordagens baseadas em algoritmos de agrupamento combinados podem levar a aproximações de séries com graus variados de precisão a depender dos filtros selecionados pelo usuário, visto que certas filtragens podem afetar de forma diferente cada *cluster* gerado.

Assim, considerando que a obtenção de variadas bases de dados reais de grandes dimensões representaria um desafio excessivamente grande para a realização dos experimentos propostos, optou-se pela geração de conjuntos de dados de forma procedural. Esse tipo de dado gerado algoritmicamente também é comumente chamado de "dado sintético" e é muito utilizado no contexto de *big data*, de aprendizado de máquina e em aplicações que virão a lidar com dados sensíveis [49]. Com dados

sintéticos, faz-se possível testar melhor as aptidões e limites de algoritmos a serem estudados, já que se possui o controle total sobre a forma como os dados estão distribuídos e suas estatísticas, como média, variância, *skewness* e outras, desde o momento de sua geração. Além disso, passa a ser possível gerar diferentes conjuntos de dados com diferentes características conforme desejado e quantas vezes for necessário.

No contexto deste trabalho, foram definidos os seguintes atributos para o conjunto de dados sintéticos:

- N1, com distribuição normal de média 75 e variância 20;
- N2, com distribuição normal de média 25 e variância 20;
- U1, com distribuição uniforme entre 0 e 100;
- G1, definida de forma a apresentar padrões periódicos crescentes mês a mês com descontinuidades acrescida de um ruído aleatório, como ilustrado na Figura 5.1;
- S1, composta por uma senoide de forma a apresentar ciclos sazonais ao longo do tempo, acrescida de um ruído aleatório, como ilustrado na Figura 5.2.

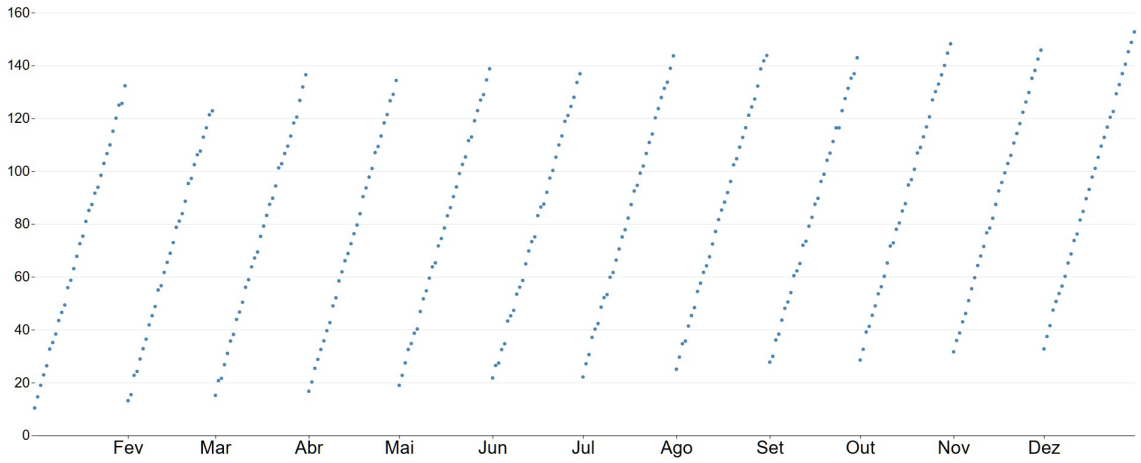


Figura 5.1: Série temporal do atributo G1 com agrupamento pela média

As colunas do conjunto de dados foram construídas utilizando a biblioteca SciPy para geração das distribuições mencionadas. O conjunto de dados resultante foi composto por 100.000.000 linhas, representando observações individuais, considerando os limites de memória disponível para a realização dos testes. Já um segundo conjunto composto de apenas 100.000 linhas foi gerado para testes preliminares necessários para o ajuste de parâmetros das diferentes técnicas e ajustes finos que podem se valer de maior celeridade. Para cada uma dessas observações, foi atribuída uma data aleatória compreendida no intervalo entre 01/01/2012 e 31/12/2012

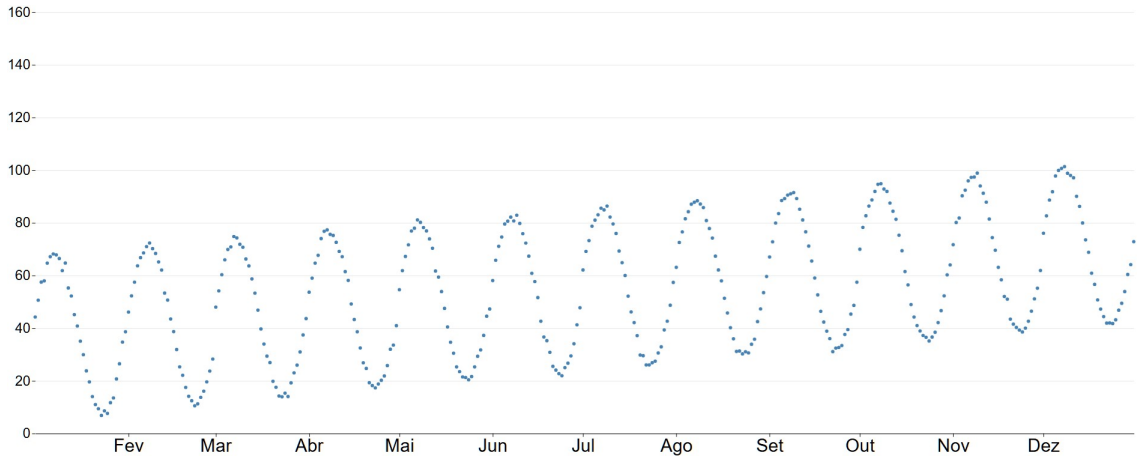


Figura 5.2: Série temporal do atributo S1 com agrupamento pela média

(abrangendo 366 dias, dado que 2012 foi um ano bissexto), seguindo uma distribuição uniforme. Portanto, múltiplas medições – e, por conseguinte, valores para cada um dos atributos sintéticos – podem ser encontradas associadas a um determinado instante do tempo, em uma densidade de dados que flutua ao longo dos dias do ano, demandando o uso de funções de agregação.

Considerando $day(x)$ e $month(x)$ funções que retornam o dia do mês e o mês da observação x e $random(x, y)$ uma função que retorna um número aleatório entre x e y, as variáveis G1 e S1 foram definidas, respectivamente, através das seguintes expressões:

$$G1(x) = 5 + 4 * day(x) + 2 * month(x) + random(-30, 30),$$

$$S1(x) = 35 + 30 * sen(2\pi * \frac{day(x)}{31}) + 3 * month(x) + random(-30, 30).$$

Adicionalmente, foram realizados também experimentos sobre um conjunto de dados real a fim de demonstrar sua eficácia quando aplicado na prática. Foi utilizado o conjunto de dados de preços de ações de empresas listadas no índice SP500 como as empresas mais valiosas dos Estados Unidos da América negociadas na bolsa de valores de Nova York. Este *dataset* é particularmente relevante devido à sua abrangência temporal, abrangendo todos os dias de negociação da bolsa entre 2010 e 2016, com preços de abertura, fechamento, máximas e mínimas, totalizando 851.264 linhas. Para cada empresa, foram incluídos dados relativos a lucro, fluxo de caixa e outros atributos que podem ser utilizados na filtragem de um subconjunto de ações.

Estão listadas 505 diferentes ações de 353 setores distintos, sendo o setor de consumo e o setor industrial os mais representados, com 17% e 14% das empresas vindas desses setores, respectivamente. Considerando todo o universo de ações coberto pelo conjunto de dados, o preço de fechamento médio foi de aproximadamente $65,012$ dólares e desvio padrão de $75,201$ dólares, com menor preço registrado de

1, 59 dólares e maior de 1578, 13 dólares. A empresa listada há mais tempo no índice foi inserida em 30/03/1964 e a mais recente em 04/01/2017, sendo a data de inclusão média 25/07/2001. Já a moda da cidade-sede das empresas listadas é a cidade de Nova Iorque, nos Estados Unidos da América, com 9% das companhias sediadas lá.

Com isso, é possível notar a grande diversidade representada no conjunto de dados, com distribuições não uniformes e com distintas possibilidades de filtro para cada variável. Porém, diferentemente dos dados sintéticos, o número de registros de cada empresa em cada dia é sempre único para cada dia e, por conseguinte, constante.

A escolha desse conjunto de dados se justifica por sua representatividade no mercado financeiro e pela possibilidade de realizar análises robustas e abrangentes das tendências de mercado. Ainda, a natureza multidimensional deste conjunto, que inclui atributos como setor de atuação e outras métricas financeiras individuais das empresas, permite simular cenários de exploração de dados mais realistas. A aplicação de diferentes filtros sobre esses atributos – por exemplo, isolando empresas de um setor específico como tecnologia ou selecionando apenas aquelas com determinado perfil de lucratividade ou fluxo de caixa – invariavelmente resulta na geração de séries temporais com características distintas, refletindo comportamentos específicos de cada subconjunto analisado.

Essa característica é fundamental não somente para testar a capacidade das técnicas de visualização progressiva propostas em lidar com a variabilidade inerente às consultas ad-hoc, mas também para avaliar como a performance dos diferentes algoritmos testados (em termos de tempo de processamento e acurácia da aproximação) pode variar significativamente em função das propriedades estatísticas – como tamanho, variância ou presença de sazonalidades específicas – do subconjunto de dados resultante da filtragem. Subconjuntos menores podem, em termos de velocidade, favorecer abordagens de amostragem, por exemplo, enquanto dados com forte periodicidade após filtragem podem beneficiar particularmente a compressão via transformada de Fourier, influenciando assim a escolha da técnica mais adequada em cenários de uso dinâmicos.

Esse conjunto de dados foi obtido através da plataforma colaborativa de ciência de dados *Kaggle*, amplamente reconhecida por sua vasta coleção de *datasets*, e pode ser encontrada no endereço https://www.kaggle.com/datasets/dgawlik/nyse.

5.2 Resultados

Os testes foram realizados em um ambiente Anaconda (conda 23.5.0), escolhido para o gerenciamento dos pacotes utilizados e muito empregado em projetos de ciência de dados.

Devido a sua maior compatibilidade com as necessidades de comunicação assíncrona de uma aplicação de transmissão progressiva, foi feita a escolha pelo uso do servidor Python baseado em ASGI *Uvicorn* para oferecer a aplicação a ser desenvolvida. Esse é um servidor baseado na biblioteca `asyncio` da linguagem Python e otimizado para desempenho, sendo amplamente utilizado para servir aplicações web assíncronas, como APIs desenvolvidas com FastAPI.

Sua capacidade de lidar com requisições HTTP de forma eficiente se aproveita de recursos modernos de assincronia para suportar múltiplas conexões simultaneamente, garantindo baixa latência e alta largura de banda. Essa escolha garante que as requisições sejam processadas de forma ágil e que os dados sejam transmitidos com baixa latência, contribuindo para uma experiência de usuário mais fluida e responsiva. Além disso, o Uvicorn permite a implementação de WebSockets e outras funcionalidades assíncronas, bem como pode ser combinado a servidores mais robustos e escaláveis como o Gunicorn, em desenvolvimentos futuros.

Inicialmente, para cada uma das três estratégias de progressividade (amostragem, transformadas e redes neurais), serão detalhados os experimentos e as métricas obtidas com o uso do conjunto de dados sintéticos. Esta abordagem permite uma análise paramétrica controlada, investigando sistematicamente como as variações nos algoritmos impactam o desempenho em um ambiente com características conhecidas.

Posteriormente, ao final da discussão de cada uma dessas estratégias, serão apresentados os resultados da aplicação da mesma técnica sobre o conjunto de dados reais. Esta segunda etapa servirá como validação, demonstrando a generalização e a eficácia dos métodos em um cenário prático, com dados de maior complexidade e variabilidade.

5.2.1 Abordagem não progressiva

A fim de estabelecer valores de referência para o tempo de processamento decorrido, cobertura (proporção do intervalo de tempo disponível nos dados efetivamente selecionados na amostragem), erro quadrático médio e volume, foi implementada a geração da série temporal sem a aplicação de nenhuma otimização. Computada utilizando todo o conjunto de dados, a série gerada neste caso-base fornece os valores exatos que buscamos equiparar através dos demais algoritmos de maneira mais rápida, demandando menos recursos computacionais.

A consulta a ser respondida por este método de referência pode ser descrita de forma análoga a um comando SQL, que calcula a média diária de uma variável de interesse. Por exemplo, para a variável S1, a consulta seria:

```
SELECT dia, AVG(S1) FROM dados GROUP BY dia;
```

Na implementação, foram utilizados dois *arrays* acumuladores correspondendo a todos os dias do ano cobertos pelos dados. O primeiro *array* acumula a soma dos valores da variável (e.g., S1) para cada dia, enquanto o segundo contabiliza o número de registros encontrados para aquele mesmo dia. Ao final da varredura, a série temporal final é gerada pela divisão, elemento a elemento, do *array* de somas pelo *array* de contagens, obtendo-se assim a média exata para cada dia.

O uso da média como função de agregação foi uma decisão metodológica que buscou garantir uma base de comparação consistente entre todas as abordagens, permitindo isolar e entender as características essenciais de cada técnica progressiva de forma mais direta. Agregações mais sofisticadas introduzem uma camada adicional de complexidade, o que poderia tornar mais complicada a interpretação dos resultados experimentais. No entanto, com poucas modificações, a ferramenta desenvolvida é extensível a qualquer função de agregação definida pelo usuário.

As métricas se encontram exibidas na Tabela 5.1, medidas em consultas para a média da variável S1 do conjunto de dados sintético, a fim de ilustrar os resultados obtidos para uma dada variável. Todos os valores se encontram arredondados para a terceira casa decimal.

Embora a simulação e subsequente análise tenham abrangido diversas variáveis, incluindo N1, N2 e U1, optou-se por circunscrever a apresentação e a discussão detalhada de resultados obtidos às variáveis S1 e G1. Essa decisão se fundamenta na observação de que as demais variáveis não exibiram resultados que se desviassem significativamente do comportamento já ilustrado por S1 e G1 e não ofereceram *insights* adicionais substanciais para a compreensão das características e desempenho das técnicas.

N1, N2 e U1 foram cruciais para garantir a completude da avaliação em diferentes condições; no entanto, a ausência de padrões mais complexos não permite inferências ricas sobre a eficácia das técnicas em identificar e representar estruturas subjacentes aos dados. Por exemplo, o comportamento de U1 na visualização progressiva se mostrou trivial, apresentando uma dispersão homogênea dos dados, sem que pudessem se destacar as nuances das técnicas propostas de modo diferente das variáveis S1 e G1.

Assim, a inclusão de uma análise exaustiva de todas as variáveis simuladas implicaria em uma repetição desnecessária, sem um ganho proporcional em termos de valor científico. Portanto, nos concentraremos na análise nas variáveis S1 e G1, por serem representativas e suficientes para demonstrar as particularidades e eficácia das técnicas testadas.

É esperado para esta abordagem de referência que os valores para cobertura, erro e volume de dados sejam constantes. A cobertura temporal é, por definição, igual a 1,0, pois a totalidade dos intervalos de tempo disponíveis é processada ao se utilizar

Número de linhas	Tempo (s)
100k	0,047
100M	36,870

Tabela 5.1: Tabela de medições relativas ao caso-base sem progressividade para o cálculo da série temporal da média de S1 com conjunto de dados fictícios

o conjunto completo de dados. De forma análoga, o erro quadrático médio (MSE) é sempre nulo, uma vez que a série gerada representa o resultado exato que serve como linha de base para a avaliação das demais técnicas de aproximação. Finalmente, o volume de 22,0 kB permanece invariável, pois corresponde à representação completa e inalterada dos 366 pontos da série temporal gerada para a consulta em questão e transmitidos do servidor ao cliente.

De modo análogo, a mesma medição de referência foi realizada para o conjunto de dados reais. A consulta para o cálculo da série temporal da média diária do preço de fechamento de todas as ações listadas no índice demandou 0,422 s de tempo de processamento no servidor. O volume de dados da resposta, contendo a série temporal exata, foi de 86,1 kB, devido à extensão por vários anos dos dados. A título de ilustração do impacto dos demais componentes de latência, o tempo total decorrido, desde a submissão da requisição pelo cliente até a completa renderização da visualização na tela, foi de 1,39 segundo, englobando, além do processamento supracitado, a transmissão pela rede e a execução no navegador.

5.2.2 Abordagem por amostragem

Na implementação mais simples de um algoritmo de amostragem, é realizado um sorteio randômico de uma quantidade de linhas da base de dados. Neste trabalho, o cliente especifica, no envio da requisição, um número p entre 0 e 1 que representa a fração do total de linhas a ser selecionada como amostra.

Foram calculados o tempo de resposta e o erro quadrático médio para diferentes valores de p, sendo $p = 100$ a série temporal exata considerando todas as linhas da base de dados. Vale salientar que nesta abordagem, visto que a seleção de linhas para compor a amostra ocorre randomicamente, não existe garantia de que será selecionado pelo menos um dado para cada intervalo de tempo renderizado. Nesses casos, a ferramenta de visualização interpola linearmente os valores da série nos tempos vizinhos para o desenho. O percentual de dias com dados presentes na amostra também foi calculado para os diferentes valores de p testados. Os tempos, coberturas, erros médios quadráticos e volumes são mostrados na Tabela 5.2.

Além do sorteio randômico dessa amostra sobre a totalidade do conjunto de dados, podem ser aplicados algoritmos baseados em amostragem agrupada. Ao utilizar a amostragem agrupada, são realizados sorteios independentes dentro de

%	Tempo (s)	Cobertura	MSE	Volume (kB)
1	$0{,}010 \pm 0{,}003$	$0{,}923 \pm 0{,}012$	$143{,}381 \pm 7{,}123$	19,8
2	$0{,}011 \pm 0{,}003$	$0{,}992 \pm 0{,}005$	$65{,}811 \pm 3{,}255$	20,9
10	$0{,}013 \pm 0{,}004$	$1{,}000 \pm 0{,}000$	$10{,}530 \pm 0{,}705$	22,0
20	$0{,}019 \pm 0{,}006$	$1{,}000 \pm 0{,}000$	$4{,}672 \pm 0{,}235$	22,0
50	$0{,}026 \pm 0{,}008$	$1{,}000 \pm 0{,}000$	$1{,}127 \pm 0{,}056$	22,0

Tabela 5.2: Tabela de tempos, cobertura e erro médio quadrático (MSE) da média diária de S1 por porcentual amostrado para o conjunto de 100.000 linhas

diferentes agrupamentos previamente criados sobre o conjunto de dados.

Os agrupamentos podem ser gerados de diferentes maneiras. Por exemplo, os dados podem ser agrupados em determinados intervalos de tempo, como dias, meses, anos ou outras janelas de tempo que se mostrem mais adequadas para o recorte e o nível de detalhe que o usuário deseja visualizar. Outro caso é o agrupamento por similaridade, utilizando algoritmos mais intrincados de aprendizagem de máquina como KNN e DBSCAN, em que o conjunto de dados é subdividido em agrupamentos de modo que integrantes de um mesmo grupo possuam características em comum. Em resumo, essa modalidade de amostragem tem como objetivo alcançar uma seleção mais representativa, garantindo que dados com características diversas se tornem parte da amostra.

Agrupado por data

Uma primeira melhoria a ser implementada no algoritmo de amostragem consiste em um agrupamento prévio dos dados de acordo com o intervalo de tempo a que se referem. Nos testes realizados, por exemplo, ao gerar uma série temporal com granularidade em nível de dia, cada linha do conjunto de dados foi classificada em grupos de cada data disponível.

No momento do sorteio dos dados a serem considerados como amostra, dado um percentual x a ser amostrado, são selecionados $x\%$ das amostras em cada data. Essa abordagem busca garantir a maior cobertura possível e uniforme do tempo contemplado pelo conjunto de dados, minimizando a chance de eventuais amostragens com representatividade excessiva de um recorte específico.

Os resultados são exibidos na Tabela 5.3, em que pode-se observar a garantia de cobertura de todas as datas presentes no conjunto de dados e uma redução marginal no erro médio quadrático. Observou-se que a cobertura temporal atingiu o valor máximo de 1,0 em todos os casos testados, conforme esperado. Este comportamento é uma consequência direta da metodologia empregada, que, ao realizar sorteios independentes dentro de cada agrupamento de data, assegura que todos os dias do período analisado sejam representados na amostra final. Como resultado da

%	Tempo (s)	MSE
1	$0{,}350 \pm 0{,}041$	$140{,}150 \pm 7{,}051$
2	$0{,}353 \pm 0{,}044$	$62{,}581 \pm 3{,}126$
10	$0{,}372 \pm 0{,}055$	$10{,}038 \pm 0{,}679$
20	$0{,}383 \pm 0{,}059$	$4{,}126 \pm 0{,}205$
50	$0{,}416 \pm 0{,}071$	$0{,}812 \pm 0{,}041$

Tabela 5.3: Tabela de tempos, cobertura e erro médio quadrático (MSE) por porcentual amostrado com agrupamento por data da média diária da variável S1 para o conjunto de 100.000 linhas

cobertura completa e consistente, consequentemente, o volume de dados transmitido ao cliente também se manteve constante em 22,0 kB, uma vez que a série temporal gerada possui sempre o mesmo número de pontos. Para privilegiar a visualização das métricas que efetivamente variaram entre os experimentos, os valores de cobertura e volume foram omitidos da tabela, que foca na apresentação dos tempos de processamento e do erro quadrático médio.

KNN

O algoritmo *K-Nearest Neighbors* (em tradução livre para o português "K- Vizinhos Mais Próximos"), também conhecido como "KNN", é amplamente empregado em projetos de ciência de dados e aprendizado de máquina para a criação de agrupamentos a partir de um conjunto de dados. Em síntese, são calculadas as distâncias entre cada ponto do conjunto de dados e os demais pontos, utilizando uma métrica de similaridade, como a distância euclidiana. Para cada ponto, são selecionados os k vizinhos mais próximos, que são aqueles com as menores distâncias em relação ao ponto em questão.

A partir dessa seleção, o algoritmo pode realizar tarefas como classificação, atribuindo ao ponto a classe mais frequente entre seus vizinhos, ou regressão, calculando uma média ponderada dos valores dos vizinhos. No contexto de séries temporais, o KNN pode ser utilizado para identificar padrões locais e realizar aproximações baseadas em pontos semelhantes, o que pode ser útil para a visualização progressiva de dados.

Na experimentação realizada, utilizamos os grupos, também denominados *clusters*, criados pela classificação através do KNN para dividir o conjunto de dados em conjuntos menores de observações que sejam similares entre si. Assim, no momento da escolha de uma amostra, são selecionados exemplos de cada agrupamento. Com isso, espera-se obter uma amostra mais representativa, contemplando dados com determinadas características notáveis, mas menos frequentes e que, em uma amostragem randômica simples, poderiam ser ignorados.

Essa abordagem é particularmente útil em conjuntos de dados com alta densi-

dade em determinadas regiões, pois permite capturar nuances locais sem depender exclusivamente de sorteios aleatórios. Além disso, o KNN pode ser ajustado para considerar pontos baseados na distância, atribuindo maior importância a pontos mais próximos do centro de interesse, o que pode melhorar a representatividade da amostra.

A métrica utilizada para o algoritmo KNN é a distância euclidiana, calculada no espaço de atributos multivariado do conjunto de dados (considerando todas as variáveis simultaneamente). Nesse modelo, cada observação é representada como um vetor, e o algoritmo agrupa os pontos com base na sua proximidade nesse espaço n-dimensional. Portanto, a proximidade de um ponto ao centroide de um *cluster* reflete sua similaridade vetorial com as demais observações do mesmo grupo. A subsequente amostragem, ao ser estratificada por esses *clusters* de similaridade, torna-se mais representativa da estrutura subjacente dos dados do que uma seleção puramente aleatória.

Vale salientar que, por se tratar de uma espécie de aprendizado de máquina não supervisionada, não se faz necessário qualquer trabalho prévio de agrupamento, permitindo que tal abordagem seja rapidamente utilizada para qualquer conjunto de dados carregado na arquitetura aqui proposta, como mostrado na Tabela 5.4. Por outro lado, número de *clusters* k em que o conjunto de dados será subdividido é um hiperparâmetro, que deve ser dado como entrada para o algoritmo e requer um esforço de otimização para que os agrupamentos gerados como saída permitam a maior redução do erro da série temporal estimada via amostragem.

É pertinente detalhar os hiperparâmetros utilizados na implementação do algoritmo KNN, cujos valores foram mantidos constantes em todos os experimentos para garantir a comparabilidade dos resultados. O número de *clusters*, k, foi definido como 128. A escolha de k é de fundamental importância, pois um valor inadequado pode comprometer a capacidade do algoritmo em modelar a estrutura dos dados. Por isso, a definição de $k = 128$ não foi arbitrária, mas sim emergiu como o resultado de uma análise exploratória preliminar, na qual foram testados diferentes valores.

Adicionalmente, para mitigar o risco de convergência para um mínimo local subótimo, o algoritmo foi executado 10 vezes com diferentes sementes de aleatoriedade para a inicialização dos centroides ($n_init = 10$), sendo selecionado o resultado com a menor inércia. O número máximo de iterações para cada execução foi fixado em 5000 ($max_iter = 5000$), um limiar suficientemente elevado para garantir a convergência na vasta maioria dos cenários.

DBSCAN

Já o algoritmo DBSCAN (do inglês, *Density-Based Spatial Clustering of Applications with Noise*) é uma poderosa técnica de agrupamento baseada em densidade que

%	Tempo (s)	Cobertura	MSE	Volume (kB)
1	$0{,}076 \pm 0{,}021$	$0{,}929 \pm 0{,}012$	$131{,}139 \pm 6{,}550$	20,2
2	$0{,}079 \pm 0{,}024$	$0{,}997 \pm 0{,}007$	$64{,}435 \pm 3{,}220$	21,8
10	$0{,}097 \pm 0{,}034$	$1{,}000 \pm 0{,}000$	$10{,}188 \pm 0{,}667$	22,0
20	$0{,}109 \pm 0{,}039$	$1{,}000 \pm 0{,}000$	$4{,}415 \pm 0{,}220$	22,0
50	$0{,}132 \pm 0{,}048$	$1{,}000 \pm 0{,}000$	$1{,}049 \pm 0{,}052$	22,0

Tabela 5.4: Tabela de tempos, cobertura e erro médio quadrático (MSE) da média diária de S1 por porcentual amostrado via KNN para o conjunto de 100.000 linhas

identifica *clusters* em um conjunto de dados ao considerar regiões de alta densidade separadas por áreas de baixa densidade. O algoritmo começa selecionando um ponto aleatório e verificando se ele possui pelo menos um número mínimo de vizinhos dentro de uma distância predefinida, chamada de épsilon (ϵ), dada como entrada e que requer uma otimização prévia. Caso essa condição seja atendida, o ponto é considerado um núcleo de *cluster*, e seus vizinhos são adicionados ao mesmo *cluster*. Esse processo é repetido para cada novo ponto adicionado, expandindo o *cluster* até que não haja mais pontos vizinhos que atendam aos critérios de densidade. Pontos que não pertencem a nenhum *cluster* são classificados como ruído.

O valor de épsilon utilizado nos experimentos foi previamente ajustado em uma busca no intervalo de 100 a 1.000 em incrementos de 50. Este processo envolveu uma avaliação empírica com diferentes valores do parâmetro, acompanhada da avaliação do desempenho do agrupamento em relação às características específicas da série S1, de modo similar à escolha do valor de k para o KNN. Nos testes com dados sintéticos, um valor de 500 foi selecionado para refletir adequadamente a densidade dos dados, permitindo que o algoritmo conectasse pontos que pertencem à mesma oscilação fundamental, ao mesmo tempo em que se mostrava robusto às pequenas flutuações introduzidas pelo ruído.

No contexto de amostragem aleatória, o DBSCAN pode ser utilizado para identificar regiões densas no conjunto de dados, as quais são caracterizadas pela concentração de observações – sendo cada observação, correspondente a uma linha da estrutura tabular de dados, interpretada como um ponto em um espaço multidimensional para fins de análise de proximidade – e, a partir dessas regiões, selecionar um subconjunto de tais observações para compor a amostra. Isso é especialmente relevante em séries temporais ou dados multidimensionais onde a densidade pode variar significativamente. Por exemplo, em um conjunto de dados com períodos de alta e baixa atividade, o DBSCAN pode ajudar a garantir que a amostra inclua representações adequadas de ambas as situações, evitando que regiões de baixa densidade sejam sub-representadas ou ignoradas. Além disso, ao identificar pontos de ruído, o algoritmo pode auxiliar na exclusão de dados discrepantes (*outliers*) que poderiam distorcer a análise ou a visualização dos dados. Essa abordagem permite

uma seleção mais informada e estruturada, especialmente em cenários onde a distribuição dos dados não é uniforme. Os resultados obtidos com o uso do DBSCAN se encontram na Tabela 5.5.

%	**Tempo (s)**	**MSE**
1	$0{,}199 \pm 0{,}035$	$135{,}126 \pm 6{,}851$
2	$0{,}201 \pm 0{,}039$	$63{,}159 \pm 3{,}152$
10	$0{,}217 \pm 0{,}049$	$10{,}029 \pm 0{,}664$
20	$0{,}229 \pm 0{,}052$	$4{,}159 \pm 0{,}206$
50	$0{,}264 \pm 0{,}066$	$0{,}959 \pm 0{,}046$

Tabela 5.5: Tabela de tempos, cobertura e erro médio quadrático (MSE) da média diária da variável S1 por porcentual amostrado via DBSCAN para o conjunto de 100.000 linhas

O algoritmo DBSCAN demonstrou comportamento consistente em relação aos parâmetros de cobertura e volume de dados transmitidos ao longo de todos os experimentos realizados. Observou-se que, independentemente do percentual de amostragem aplicado, a cobertura temporal atingiu o valor máximo de 1,0 em todas as execuções. Este resultado reflete a capacidade do algoritmo em identificar e representar adequadamente todas as regiões temporais do conjunto de dados, mesmo quando aplicadas taxas de amostragem reduzidas.

De forma análoga, o volume de dados transmitido ao cliente manteve-se constante em 22,0 kB para todos os percentuais testados, o que decorre diretamente da cobertura completa. Por motivo de concisão e para enfatizar as métricas que apresentaram variação significativa, os valores de cobertura e volume não foram explicitados na Tabela 5.5, que concentra-se nos tempos de processamento e no MSE.

OPTICS

Por usa vez, quanto ao algoritmo OPTICS (do inglês, *Ordering Points To Identify the Clustering Structure*), tratamos de uma extensão do DBSCAN que busca superar suas limitações em relação à escolha fixa de parâmetros, como o raio ϵ. Em vez de formar *clusters* diretamente definidos, o OPTICS ordena os pontos do conjunto de dados com base em sua densidade local, criando uma estrutura hierárquica que permite a identificação de *clusters* em diferentes escalas. O algoritmo calcula duas métricas principais para cada ponto: a distância de alcance (*reachability distance*) e a distância central (*core distance*), que determinam a proximidade e a densidade relativa dos pontos, respectivamente. A partir dessas métricas, é gerada uma representação visual chamada de gráfico de alcance (*reachability plot*), que facilita a identificação de *clusters* e *subclusters*, subdivisões ainda menores e mais particulares.

No contexto de séries temporais, o OPTICS pode ser empregado para explorar padrões em diferentes níveis de granularidade. Para amostragem aleatória, o OP-

TICS pode ser utilizado para selecionar linhas do conjunto de dados com base na hierarquia de densidade que ele gera. Por exemplo, ao identificar regiões de alta densidade em diferentes escalas, o OPTICS permite que a amostra inclua pontos de *clusters* mais densos, bem como de regiões menos densas, garantindo uma cobertura mais equilibrada do conjunto de dados. Essa abordagem é particularmente útil em séries temporais com padrões sazonais ou eventos raros. Além disso, a flexibilidade do OPTICS em lidar com diferentes densidades torna-o uma ferramenta poderosa para evitar vieses na seleção de amostras, especialmente em conjuntos de dados complexos e heterogêneos, garantindo os erros mais baixos conforme a Tabela 5.6.

%	**Tempo (s)**	**MSE**
1	$1{,}955 \pm 0{,}071$	$120{,}259 \pm 6{,}012$
2	$1{,}969 \pm 0{,}073$	$58{,}126 \pm 2{,}902$
10	$2{,}009 \pm 0{,}085$	$8{,}069 \pm 0{,}560$
20	$2{,}026 \pm 0{,}090$	$3{,}126 \pm 0{,}152$
50	$2{,}091 \pm 0{,}106$	$0{,}659 \pm 0{,}032$

Tabela 5.6: Tabela de tempos, cobertura e erro médio quadrático (MSE) da média diária de S1 por porcentual amostrado via OPTICS para o conjunto de 100.000 linhas

O algoritmo OPTICS, assim como o DBSCAN, apresentou resultados uniformes quanto à cobertura temporal e ao volume de dados transmitidos em todos os cenários de teste. A cobertura temporal manteve-se consistentemente no valor máximo de 1,0, independentemente do percentual de amostragem utilizado. Este comportamento confirma a eficácia do OPTICS em representar completamente o espectro temporal dos dados, mesmo com taxas de amostragem menores, resultado da sua capacidade superior de ordenação e identificação hierárquica de estruturas de agrupamento. E em relação ao volume de dados transmitidos, observou-se o mesmo padrão de constância verificado nos demais algoritmos com cobertura completa, permanecendo invariavelmente em 22,0 kB para todos os percentuais testados.

Comparativo

Inicialmente, os testes foram executados sobre o conjunto de dados sintético de 100.000 linhas, com o objetivo de avaliar como as métricas de desempenho dos algoritmos de amostragem respondiam à variação do percentual amostrado. Nesta fase, observou-se o claro *trade-off* entre o tempo de resposta e o erro (MSE), onde menores percentuais de amostragem resultavam em respostas mais rápidas, porém com maior erro. Além disso, para percentuais de amostragem baixos, a cobertura temporal nem sempre atingia 100%, o que tornava o volume de dados transmitido variável.

Posteriormente, para simular um ambiente de *big data* e testar a escalabilidade das soluções, os experimentos foram refeitos utilizando o conjunto de dados massivo de 100.000.000 de linhas, fixando um percentual de 10% para os algoritmos de amostragem a fim de criar um ponto de comparação consistente. Como esperado, o tempo de processamento aumentou drasticamente para todas as abordagens, evidenciando o desafio computacional imposto por grandes volumes de dados. Em contraste, o erro (MSE) diminuiu de forma expressiva, uma vez que uma amostra de 10% sobre a base maior é robusta o suficiente para gerar agregados extremamente precisos. Finalmente, a cobertura atingiu 1 de forma consistente, o que levou o volume de dados transmitido a se tornar constante, já que a saída sempre continha o número total de dias da série.

A Tabela 5.7 e a Tabela 5.8 consolidam os resultados, permitindo uma análise direta dos *trade-offs* entre tempo de resposta e precisão (MSE) para cada abordagem sob as condições de teste. Seus valores de cobertura e de volume permaneceram constantes em 1,0 e 22,0 kB, respectivamente. Já a Figura 5.3 ilustra os achados para análise visual, sendo a Figura 5.4 ideal para melhor compreender a variabilidade do erro de cada técnica.

	Tempo (s)	**MSE**
Simples	$0{,}012 \pm 0{,}007$	$10{,}261 \pm 0{,}703$
Agrupado	$0{,}372 \pm 0{,}055$	$10{,}038 \pm 0{,}679$
KNN	$0{,}097 \pm 0{,}034$	$10{,}188 \pm 0{,}667$
DBSCAN	$0{,}217 \pm 0{,}049$	$10{,}029 \pm 0{,}664$
OPTICS	$2{,}009 \pm 0{,}085$	$8{,}069 \pm 0{,}560$

Tabela 5.7: Quadro comparativo entre algoritmos com amostragem de 10% sobre a variável S1 para um conjunto de dados de 100.000 linhas

	Tempo (s)	**MSE**
Simples	$185{,}0 \pm 15{,}0$	$0{,}105 \pm 0{,}007$
Agrupado	$850{,}5 \pm 55{,}8$	$0{,}100 \pm 0{,}007$
KNN	$440{,}8 \pm 33{,}1$	$0{,}102 \pm 0{,}007$
DBSCAN	$660{,}2 \pm 48{,}8$	$0{,}100 \pm 0{,}007$
OPTICS	$3200{,}1 \pm 85{,}8$	$0{,}081 \pm 0{,}006$

Tabela 5.8: Quadro comparativo entre algoritmos com amostragem de 10% sobre a variável S1 para um conjunto de dados de 100.000.000 de linhas

A cobertura temporal manteve-se consistentemente no valor máximo de 1,0 para todos os algoritmos avaliados, tanto no conjunto de dados de 100.000 linhas quanto no mais expressivo de 100.000.000 de linhas. De forma análoga, o volume de dados transmitido permaneceu constante em 22,0 kB em todos os casos. Para privilegiar a clareza visual e facilitar a comparação direta entre os tempos de processamento

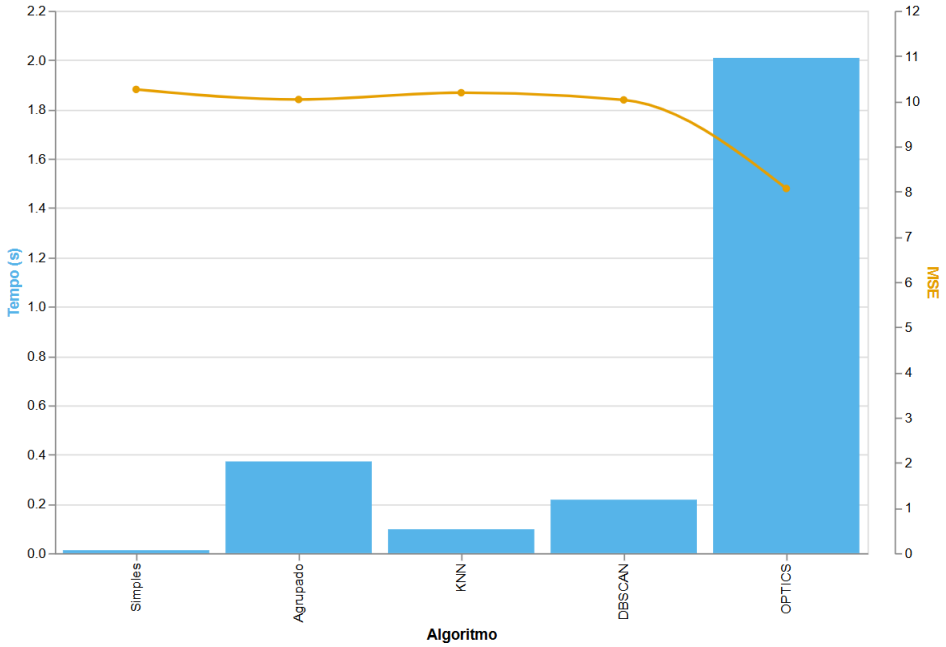


Figura 5.3: Gráfico comparativo entre tempos e erros de algoritmos no cálculo de médias diárias da variável S1 testados com conjunto de 100.000 linhas e 10% de amostragem

e os erros quadráticos médios – métricas que efetivamente apresentaram variação significativa entre os algoritmos – optou-se por omitir os valores constantes.

Para melhor compreensão do comportamento das técnicas de amostragem em cenários de consultas ad-hoc multivariadas, foram realizados experimentos adicionais com o algoritmo KNN submetido a diferentes filtros sobre o conjunto de dados sintético. Vale ressaltar que a aplicação de tais filtros foi implementada por meio de uma varredura linear simples sobre a totalidade dos registros, sem o uso de estruturas de indexação de banco de dados ou outras otimizações. Essa escolha visa simular um cenário computacionalmente mais desafiador, onde otimizações de consulta pré-existentes não estão disponíveis, avaliando o desempenho do algoritmo de amostragem isoladamente. A Tabela 5.9 apresenta os resultados dessas execuções, mantendo fixa a taxa de amostragem em 10%, de modo a isolar o efeito dos filtros sobre as métricas avaliadas.

	Tempo (s)	**Cobertura**	**MSE**
Sem filtro	$0{,}097 \pm 0{,}034$	$1{,}000 \pm 0{,}000$	$10{,}188 \pm 0{,}667$
N1 $>$ 80	$0{,}082 \pm 0{,}030$	$0{,}993 \pm 0{,}008$	$12{,}547 \pm 0{,}782$
N2 $<$ 20	$0{,}085 \pm 0{,}031$	$0{,}988 \pm 0{,}009$	$13{,}216 \pm 0{,}824$
U1 $>$ 75	$0{,}079 \pm 0{,}029$	$0{,}971 \pm 0{,}012$	$15{,}739 \pm 0{,}985$

Tabela 5.9: Quadro comparativo da geração da série temporal referente à variável S1 através do algoritmo KNN com amostragem de 10% sob diferentes filtros

Os resultados obtidos demonstram que a aplicação de filtros provoca variações

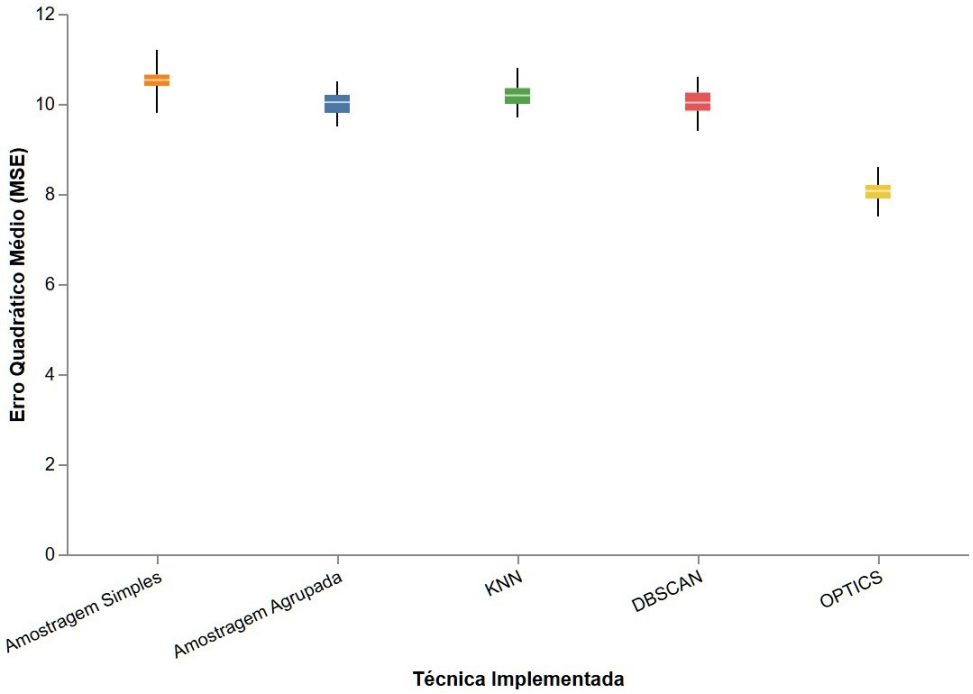


Figura 5.4: Diagrama comparativo do tipo "box plot" de erros de algoritmos testados com conjunto de 100.000 linhas

significativas no desempenho do algoritmo KNN, mesmo mantendo a taxa de amostragem constante. Primeiramente, observa-se que o tempo de processamento apresenta uma redução modesta quando filtros mais seletivos são aplicados, como na condição $U1 > 75$, que resultou em 0,079 s. Esse comportamento é esperado, pois filtros mais restritivos efetivamente diminuem o espaço de busca para o algoritmo, embora o ganho seja parcialmente compensado pelo *overhead* introduzido pelo próprio processamento do filtro.

Um aspecto particularmente relevante é o impacto dos filtros sobre a cobertura temporal das séries resultantes. Enquanto a condição sem filtro manteve uma cobertura perfeita, filtros altamente seletivos, como $U1 > 75$, reduziram esse valor para 0,971. Esta diminuição, apesar de aparentemente modesta, pode comprometer significativamente a identificação de eventos pontuais na série temporal, uma vez que representa a ausência de dados para aproximadamente 10 dias no período anual analisado, e impactou negativamente o MSE medido.

Tais resultados evidenciam que, em cenários de visualização exploratória com diferentes filtros ad-hoc, a escolha de parâmetros ótimos para o KNN pode levar em consideração não apenas os parâmetros de amostragem, mas também características específicas dos filtros aplicados e suas implicações na distribuição resultante dos dados. Por exemplo, para filtros que resultam em dados esparsos, pode ser necessário aumentar a taxa de amostragem para manter níveis aceitáveis de erro.

A seleção do algoritmo KNN para a análise comparativa sob diferentes condições

de filtragem foi motivada por suas características particulares que o posicionam em um ponto estratégico no espectro dos algoritmos de amostragem avaliados. Diferentemente da amostragem simples, que não considera a estrutura multidimensional dos dados, ou de algoritmos como OPTICS, que demandam recursos computacionais significativamente mais elevados, o KNN oferece um equilíbrio favorável entre eficiência computacional e capacidade de preservação das características estruturais dos dados.

Estas características tornam o KNN particularmente adequado para exemplificar como a aplicação de filtros em diferentes dimensões do conjunto de dados afeta o desempenho da visualização progressiva e também para a experimentação com dados reais, demonstrando comportamentos que podem ser extrapolados, com as devidas considerações, para os demais algoritmos de amostragem analisados neste trabalho.

A escolha dos filtros testados em particular foi motivada pela necessidade de representar diferentes padrões de seletividade sobre as distribuições subjacentes dos dados. O filtro $N1 > 80$ foi aplicado sobre uma variável com distribuição normal (média 75, variância 20), selecionando valores no extremo superior da distribuição. Já $N2 < 20$ atua sobre outra distribuição normal (média 25, variância 20), capturando valores abaixo da média. Por sua vez, $U1 > 75$ opera sobre uma distribuição uniforme (entre 0 e 100), isolando o quartil superior. Esta diversidade de condições permitiu observar como a seletividade dos filtros impacta particularmente a cobertura temporal e o erro quadrático médio das séries resultantes.

É importante ressaltar que estes filtros representam apenas um recorte inicial do vasto espaço de possibilidades de filtragem ad-hoc que caracterizam cenários de visualização exploratória. Um estudo mais abrangente poderia contemplar combinações de filtros (por exemplo, combinações utilizando operadores lógicos como $N1 > 80 \wedge U1 < 25$), condições mais complexas sobre as variáveis sintéticas G1 e S1 com padrões periódicos, ou ainda filtros com diferentes graus de seletividade sobre o mesmo atributo para melhor caracterizar a relação entre seletividade e degradação da cobertura. Tal investigação, a ser explorada com mais profundidade, contribuiria significativamente para a construção de sistemas de visualização progressiva adaptativos, capazes de ajustar dinamicamente seus parâmetros de amostragem em resposta aos padrões de consulta dos usuários.

Conjunto de dados reais

Os resultados obtidos na aplicação dos algoritmos de amostragem aos dados financeiros reais demonstram um comportamento consistente em termos comparativos entre as técnicas, mesmo apresentando valores de erro quadrático médio substancialmente mais elevados do que os observados nos experimentos com dados sintéticos. Enquanto nos dados sintéticos os erros se mantiveram na faixa de 8 a 10 unidades,

	Tempo (s)	Cobertura	MSE
Simples	$0{,}063 \pm 0{,}005$	$0{,}995 \pm 0{,}003$	$132{,}072 \pm 7{,}285$
Agrupado	$1{,}703 \pm 0{,}099$	$1{,}0 \pm 0{,}0$	$121{,}807 \pm 6{,}942$
KNN	$0{,}171 \pm 0{,}052$	$0{,}999 \pm 0{,}001$	$126{,}486 \pm 7{,}106$
DBSCAN	$1{,}053 \pm 0{,}088$	$1{,}0 \pm 0{,}0$	$118{,}946 \pm 6{,}825$
OPTICS	$8{,}427 \pm 0{,}315$	$1{,}0 \pm 0{,}0$	$94{,}522 \pm 5{,}318$

Tabela 5.10: Quadro comparativo entre algoritmos com amostragem de 10% sobre o conjunto de dados reais para as médias diárias do preço de fechamento das ações

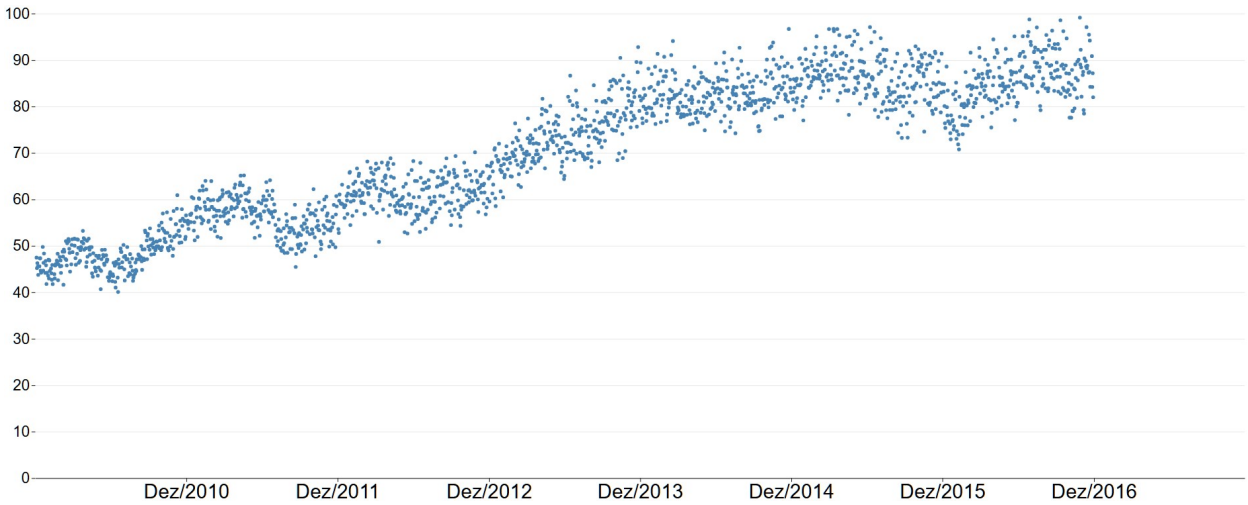


Figura 5.5: Gráfico da série temporal da média diária geral do preço de fechamento das ações gerado por 10% de amostragem utilizando KNN sobre dados reais

nos dados financeiros reais esses valores aumentaram para a faixa de 94 a 132, um incremento de aproximadamente uma ordem de grandeza.

É imperativo analisar essa diferença com cautela, considerando a escala das variáveis. A série temporal do conjunto de dados real foi calculada sobre o preço de fechamento das ações, cuja média geral no período analisado foi de 65,012 dólares. Para uma comparação mais equitativa, podemos analisar o erro relativo. A raiz do erro quadrático médio (RMSE) para os dados sintéticos, de aproximadamente 3, representa cerca de 5,5% da média da variável S1, que é cerca de 54,465. Para os dados financeiros, a RMSE de aproximadamente 11 corresponde a quase 17% do preço médio das ações, representando um erro notavelmente superior porém de cerca de 3 vezes o valor encontrado com dados fictícios.

Este aumento expressivo no MSE não representa, contudo, um problema para a aplicabilidade prática das técnicas de visualização progressiva. Tal comportamento é esperado e justifica-se pela natureza inerentemente mais complexa e volátil dos dados financeiros reais, caracterizados por padrões não lineares, eventos extremos, volatilidade agrupada e correlações temporais variantes. Devido à extensão maior da série, todos os volumes registrados foram de 86,2 kB.

O aspecto mais relevante a ser destacado é a preservação da hierarquia de eficácia

entre os algoritmos, como visto na Tabela 5.10. A amostragem simples continua apresentando o processamento mais rápido (0,063 s) com o maior erro (132,072), enquanto o OPTICS mantém sua posição como o algoritmo mais preciso (94,522) ao custo de maior tempo de processamento (8,427 s). Esta consistência na relação entre os métodos valida a robustez das abordagens em diferentes contextos de dados.

Mais importante ainda, mesmo com erros mais elevados, todas as técnicas testadas conseguiram preservar as características principais das séries temporais financeiras, como tendências de mercado, sazonalidades e pontos de reversão significativos, como pode ser observado na Figura 5.5. No contexto de visualização progressiva, onde o objetivo primário é fornecer rapidamente ao usuário uma aproximação que permita a tomada de decisão inicial enquanto os resultados exatos são processados, esses níveis de erro não comprometem a usabilidade do sistema nem a interpretação preliminar dos dados.

5.2.3 Abordagem por transformadas

No que tange à abordagem utilizando transformadas matemáticas, foram realizados testes para compreender o impacto da redução no número de coeficientes transmitidos na precisão da aproximação da série temporal. Optou-se pela experimentação com transformadas rápidas de Fourier e com transformadas por *wavelets*, contemplando duas técnicas amplamente conhecidas no campo de processamento de sinais.

Transformada de Fourier

Quanto à transformada rápida de Fourier, a transformada e a transformada inversa foram realizadas através da própria biblioteca NumPy 2.2.1, de acordo com as equações 3.1 e 3.2 abordadas anteriormente.

Nas tabelas 5.11 e 5.12, podem ser observados erros e volumes de dados obtidos nos testes executados. A primeira tabela apresenta os valores referentes a testes com o conjunto de dados sintéticos com 100.000 linhas, enquanto a segunda exibe resultados para um conjunto de 100.000.000 linhas. A realização de testes com diferentes tamanhos de conjuntos de dados visou ilustrar o impacto percebido no tempo de processamento do algoritmo de FFT decorrente do tamanho da entrada. Os tempos de processamento apresentaram um salto de 15,625 ms para 2,210 s, representando um aumento de cerca de 141,45 vezes para um crescimento de 1.000 vezes do número de linhas.

Ademais, 366 foi o número máximo de coeficientes considerado nos testes com o conjunto de dados sintéticos pois esse é o número de valores temporais possíveis compreendidos no intervalo abarcado pela série temporal. Como o sinal a ser transformado através de FFT é composto por um número finito de pontos, essa

# de coeficientes	MSE G1	MSE S1	Volume (kB)
5	1239,190	464,002	5,5
10	485,635	5,583	5,7
20	208,949	3,039	6,1
50	66,441	1,177	7,3
100	0,000	0,000	9,3
183	0,000	0,000	12,8
366	0,000	0,000	20,0

Tabela 5.11: Tabela de medições de decomposições FFT para médias diárias das variáveis G1 e S1 para 100.000 linhas

# de coeficientes	MSE G1	MSE S1	Volume (kB)
5	1240,384	464,065	5,5
10	483,436	4,652	5,7
20	207,034	2,159	6,1
50	66,558	0,655	7,3
100	0,000	0,000	9,3
183	0,000	0,000	12,6
366	0,000	0,000	19,9

Tabela 5.12: Tabela de medições de decomposições FFT para médias diárias das variáveis G1 e S1 para 100.000.000 linhas

quantidade de coeficientes já garante a reconstrução sem perdas do original, com total fidelidade.

Vale ressaltar que os resultados apresentados não incluem desvios-padrão. Isso ocorre pois, diferentemente dos experimentos anteriores com técnicas de amostragem inerentemente probabilísticas, o uso de FFTs para geração de curvas simplificadas e redução do volume de dados transmitidos é determinístico. É dispensado assim o cálculo do desvio-padrão, que seria 0 para todos os casos. Um racional análogo se aplica à métrica de cobertura definida para a abordagem por amostragem e omitida aqui por sempre assumir o valor de 1,0 tendo em vista que a totalidade do conjunto de dados é utilizada no cálculo da transformada.

Um achado importante foi a presença de valores mais elevados de erro quadrático médio para a variável G1 devido às suas descontinuidades. A variável S1, calculada a partir de senoides, foi naturalmente melhor ajustada à combinação linear de senos e cossenos representada pela transformada de Fourier e exibiu erros menores em comparação. Espera-se também que funções periódicas apresentem erros menores do que funções não periódicas.

Alguns padrões podem ser percebidos a partir das medições feitas. Em especial, destaca-se que o tempo gasto no cálculo da transformada permanece constante independentemente da variável a ser transformada, pois o número de pontos fornecidos como entrada para o algoritmo continua o mesmo. Outra importante observação –

e mais relevante para nossa análise – é o comportamento de decrescimento do erro quadrático médio com o aumento do número de coeficientes enviados. Tendência inversa ocorre para o volume de dados transmitidos pelo servidor ao cliente, crescendo de modo proporcional à quantidade de coeficientes.

Com isso, deve-se buscar um equilíbrio entre o erro considerado aceitável e o volume de dados que podem ser transmitidos sem impacto na qualidade da interação do usuário devido a atrasos de rede. Em nosso caso, nota-se empiricamente que o número de 50 coeficientes já representa um bom *trade-off* entre essas duas métricas, com escolhas maiores que esse patamar apresentando erro considerado desprezível. Nas Figuras 5.6 e 5.7, podemos ver o atingimento de patamares de erro quadrático médio consistentemente mais baixos com uma parcela pequena dos coeficientes calculados na FFT. No tocante ao volume de dados transmitidos, a Figura 5.8 mostra um comportamento notavelmente linear à medida que a quantidade de coeficientes empregados na reconstrução da série aumenta.

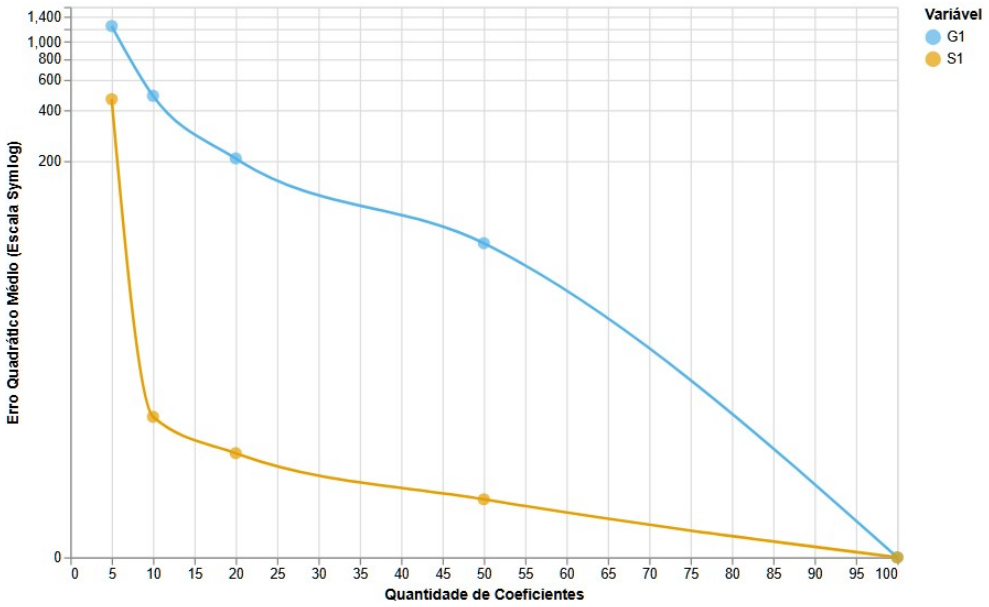


Figura 5.6: Gráfico do erro quadrático médio da série reconstruída via FFT a partir do conjunto de dados sintéticos com 100.000 linhas em relação à quantidade de coeficientes utilizados em escala *symlog*

Mais ainda, pode ser constatado outro achado relevante: qualquer quantidade de coeficientes maior ou igual a 92 ($= \left\lceil \frac{366}{4} \right\rceil$) resulta em um erro quadrático médio igual a 0 para qualquer variável e filtro. A combinação de duas propriedades importantes nos campos de análise de Fourier e amostragem é responsável por esse comportamento.

Em primeiro lugar, a saída da Transformada Discreta de Fourier (DFT) para uma sequência de entrada real de tamanho N é uma sequência de N coeficientes complexos. Devido à série temporal original ser composta unicamente por partes

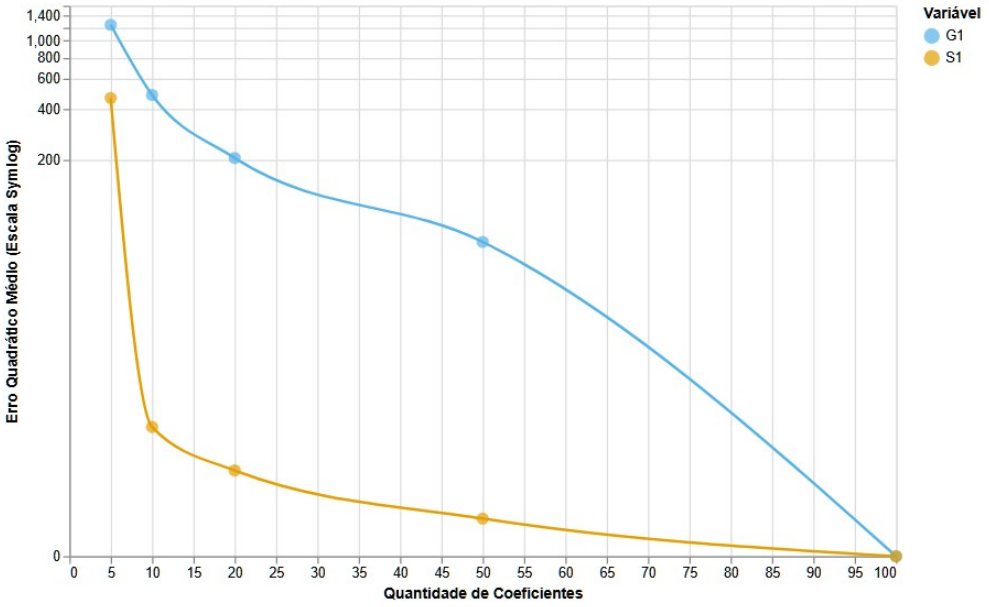


Figura 5.7: Gráfico do erro quadrático médio da série reconstruída via FFT a partir do conjunto de dados sintéticos com 100.000.000 linhas em relação à quantidade de coeficientes utilizados em escala *symlog*

reais (não contendo nenhuma parte imaginária), a saída da DFT é simétrica hermitiana. Ou seja, a segunda metade dos termos gerados são complexos conjugados da primeira metade. Portanto, basta o envio de metade não redundante dos 366 termos complexos do espaço completo, ou seja, 183 termos, para reconstruir os coeficientes completos e posteriormente aplicar a transformada inversa.

É crucial notar que essa redução é possível apenas porque a entrada é real. A sequência de coeficientes transmitida é complexa e não possui a mesma simetria, impedindo qualquer compressão recursiva. No âmbito da Teoria da Informação, a capacidade de compactar a saída da DFT decorre da redundância inerente na representação em frequência de sinais reais. A transformação simplesmente revela essa redundância, permitindo uma representação mais eficiente sem qualquer perda de informação. A entropia do sinal original é integralmente preservada na porção não redundante dos coeficientes de Fourier, sendo cada número composto por uma parte real e uma imaginária.

A segunda propriedade a ser destacada é a chamada taxa de Nyquist [50]. Decorre do teorema de Nyquist-Shannon que $\left\lceil \frac{N}{2} \right\rceil$ termos são suficientes para a reconstrução sem perdas da série original no processo da transformada inversa, sendo N o número de amostras. Portanto, dos 183 termos mencionados anteriormente, bastam 92 coeficientes para recuperar a informação original perfeitamente, sem quaisquer erros. E, aplicado em nosso caso, para qualquer consulta que resulte em uma série temporal com N pontos, são suficientes $\left\lceil \frac{N}{4} \right\rceil$ coeficientes para uma transmissão sem perdas.

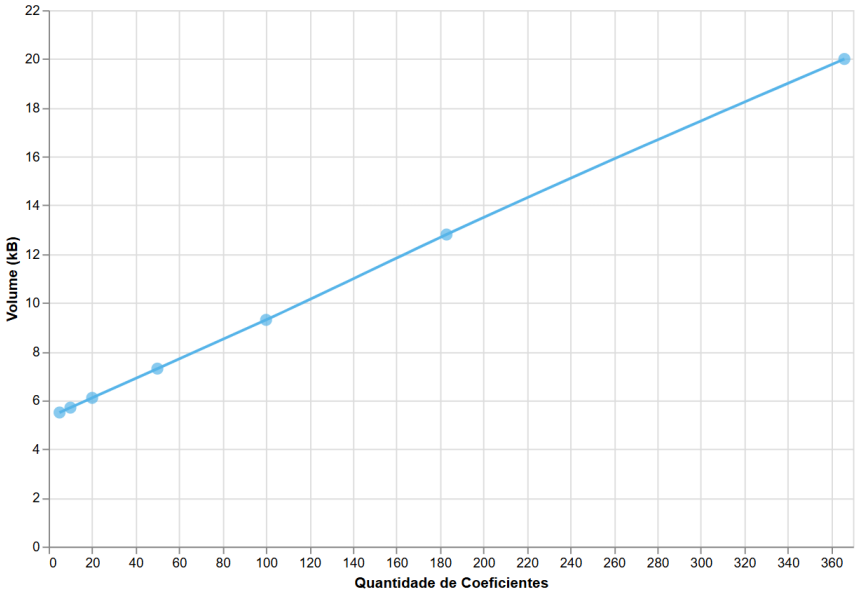


Figura 5.8: Gráfico do volume de dados transmitido em relação à quantidade de coeficientes da FFT utilizados para dados sintéticos

Analisando os resultados da abordagem por transformada de Fourier nos dados reais, observa-se que esta técnica demonstrou ser bastante eficaz na geração de séries temporais aproximadas com diferentes níveis de precisão. Na Tabela 5.13, são apresentados os resultados dos experimentos com o conjunto de dados reais, mostrando a relação entre a quantidade de coeficientes utilizados, o erro quadrático médio (MSE) e o volume de dados transmitidos. O tempo de processamento foi constante de 31,25 ms.

Os resultados indicam que o tempo de processamento permaneceu constante em 31,25 ms independentemente da quantidade de coeficientes utilizados, o que sugere que o cálculo da transformada tem um custo computacional fixo para um mesmo tamanho de entrada. No entanto, o erro quadrático médio diminui significativamente à medida que mais coeficientes são utilizados: com apenas 5 coeficientes, o MSE é de 12,995, reduzindo para 7,535 com 10 coeficientes, 3,250 com 20 coeficientes, e alcançando 1,326 com 50 coeficientes. Esta tendência de redução do erro é visualmente representada na Figura 5.9, que mostra o comportamento do erro quadrático médio em relação à quantidade de coeficientes utilizados para o conjunto de dados real. Observa-se uma curva de decaimento semelhante à encontrada nos experimentos com dados sintéticos.

O comportamento inverso foi observado para o volume de dados transmitido: começando com 25 kB para 5 coeficientes e chegando a 95,8 kB ao transmitirmos a lista completa de 1762 coeficientes. Assim como no caso sintético, o ponto de equilíbrio ficou, empiricamente, em torno de 50 coeficientes – *trade-off* que entrega erro desprezível com apenas 26,8 kB trafegados, valor perfeitamente compatível com

aplicações web em redes de largura de banda modesta.

Em suma, os resultados com dados reais confirmam que a abordagem baseada em FFT é capaz de gerar boas estimativas da série temporal, permitindo a reconstrução de uma série aproximada que representa adequadamente tendências de crescimento e flutuações locais dos preços médios das ações. Isto demonstra a aplicabilidade prática desta otimização na redução do tamanho da resposta do servidor, com consequente diminuição da latência de rede experimentada pelo usuário durante a visualização dos dados.

# de coeficientes	MSE	Volume (kB)
5	12,995	25,0
10	7,535	25,2
20	3,250	25,6
50	1,326	26,8
100	0,644	28,8
1000	0,000	64,8
1762	0,000	95,8

Tabela 5.13: Tabela de medições de decomposições FFT com dados reais relativas à média diária do preço de fechamento das ações

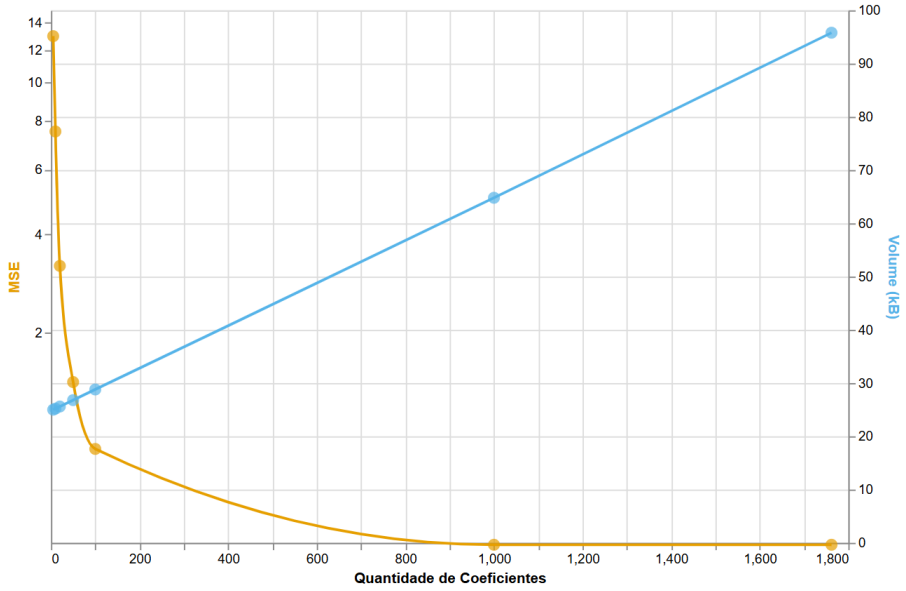


Figura 5.9: Gráfico do erro quadrático médio da série reconstruída via FFT e do volume dos coeficientes transmitidos a partir do conjunto de dados real em relação à quantidade de coeficientes utilizados

Transformada por *wavelets*

Adicionalmente aos testes com transformadas de Fourier, foram conduzidos experimentos exploratórios com transformadas de *wavelets*, notadamente a família Daube-

chies, utilizando a biblioteca PyWavelets. A metodologia consistiu na decomposição multi-nível da série temporal original, gerando coeficientes de aproximação e de detalhe em diferentes escalas. A série era então reconstruída utilizando apenas os coeficientes de aproximação e um subconjunto dos coeficientes de detalhe, em uma abordagem análoga à seleção de coeficientes da FFT.

Observa-se, pela Tabela 5.14 e pela Tabela 5.15, que as *wavelets* se mostraram particularmente eficazes para a variável G1, cujas descontinuidades foram mais bem representadas pelos coeficientes de detalhe em comparação com a abordagem de Fourier. Este achado sugere que, para séries com características não estacionárias ou eventos localizados e abruptos, as *wavelets* podem oferecer uma alternativa vantajosa para a geração de aproximações progressivas, também com aplicação a dados reais validada conforme a Tabela 5.16.

Wavelet - Nível	**Tempo (s)**	**MSE G1**	**MSE S1**	**Volume (kB)**
db2 - 1	0,172	1217,458	478,627	6,4
db2 - 5	0,172	198,810	4,203	9,6
db10 - 1	0,188	832,270	285,889	6,4
db10 - 5	0,188	0,000	0,000	9,6
db20 - 1	0,250	373,352	3,611	6,4
db20 - 5	0,250	0,000	0,000	9,6

Tabela 5.14: Tabela de medições de decomposições com *wavelets* de Daubechies para variáveis G1 e S1 para 100.000 linhas

Wavelet - Nível	**Tempo (s)**	**MSE G1**	**MSE S1**	**Volume (kB)**
db2 - 1	5,672	1215,258	479,962	6,4
db2 - 5	5,672	198,779	3,524	9,6
db10 - 1	5,734	830,216	284,965	6,4
db10 - 5	5,734	0,000	0,000	9,6
db20 - 1	5,797	371,972	2,871	6,4
db20 - 5	5,797	0,000	0,000	9,6

Tabela 5.15: Tabela de medições de decomposições com *wavelets* de Daubechies para variáveis G1 e S1 para 100.000.000 linhas

No contexto dos experimentos realizados, a notação adotada, como “db10 - 5”, segue uma convenção para descrever a configuração específica da transformada. O componente “dbN” refere-se à *wavelet*-mãe da família Daubechies de ordem N, onde N indica o número de *vanishing moments* (pontos nulos), uma propriedade que se relaciona com a capacidade da *wavelet* de representar eficientemente polinômios e, por conseguinte, aproximar sinais suaves. *Wavelets* de ordem superior tendem a ser mais suaves, porém computacionalmente mais intensivas.

O segundo componente, o número que sucede o hífen, representa o nível (ou profundidade) da decomposição multirresolução aplicada à série temporal. Um nível de

Wavelet - Nível	Tempo (s)	MSE	Volume (kB)
db2 - 1	0,875	13,865	6,4
db2 - 5	0,875	1,308	9,6
db10 - 1	0,906	2,366	6,4
db10 - 5	0,906	0,177	9,6
db20 - 1	0,938	1,336	6,4
db20 - 5	0,938	0,073	9,6

Tabela 5.16: Tabela de medições de decomposições com *wavelets* de Daubechies para dados reais referentes à média diária do preço de fechamento de todas as ações

decomposição 1 gera um conjunto de coeficientes de aproximação (baixa frequência) e um de detalhe (alta frequência). Um nível 5, por sua vez, realiza esse processo recursivamente cinco vezes sobre os coeficientes de aproximação. A escolha desses dois parâmetros, a ordem da *wavelet* e o nível de decomposição, representa, portanto, um *trade-off* fundamental entre a compactação da informação e a fidelidade da série reconstruída.

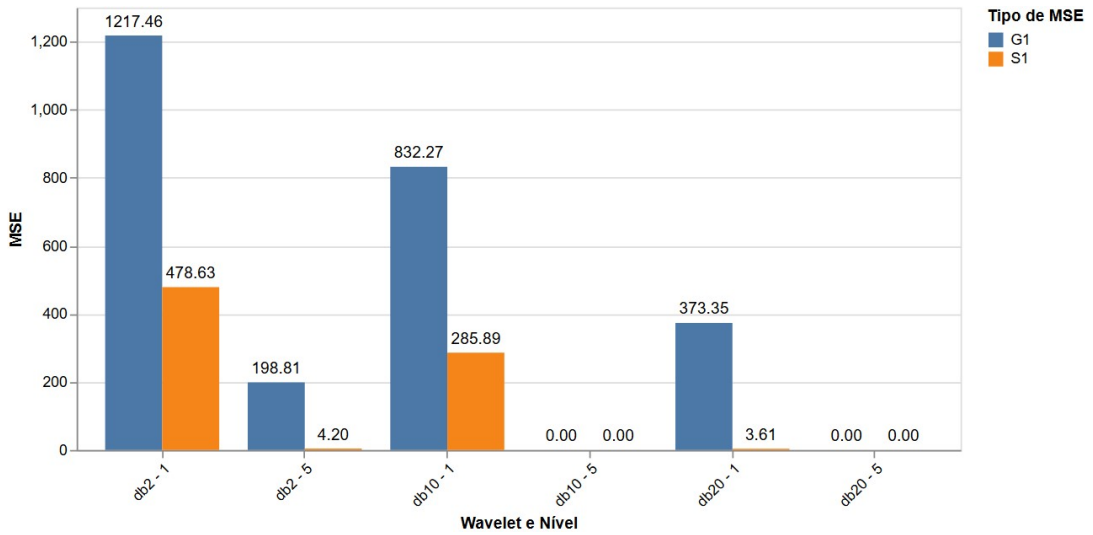


Figura 5.10: Gráfico do erro quadrático médio da série reconstruída a partir do conjunto de dados sintético de 100.000 linhas em relação à *wavelet* utilizada

Um achado de particular relevância reside no impacto do nível de decomposição sobre a precisão da aproximação. Como esperado, para as *wavelets* de ordem superior db10 e db20, o MSE converge para valores nulos (ou praticamente nulos) quando o nível de decomposição atinge 5. Esta convergência para erro zero, em um nível de decomposição mais profundo, é um indicador robusto da capacidade dessas *wavelets* em capturar a informação essencial da série temporal com uma fidelidade quase perfeita. Em contrapartida, a *wavelet* db2, mesmo no nível 5, apresenta um MSE residual não desprezível, especialmente para a variável G1. Este comportamento, que pode ser visualizado na Figura 5.10 sugere uma menor eficácia da db2 na minimização do erro em decomposições mais profundas, quando comparada às

suas contrapartes de ordem superior.

Em relação ao tempo de processamento, observa-se uma constância notável para todas as *wavelets* e níveis de decomposição analisados. Esta estabilidade temporal, independentemente do nível de decomposição (1 ou 5), sugere que o custo computacional primário está associado à etapa inicial de cálculo da transformada para um determinado tamanho de entrada, com um impacto marginal do aprofundamento da decomposição.

Quanto ao volume de dados transmitido, verifica-se que este permanece constante em 6,4 kB para o nível 1 e 9,6 kB para o nível 5, independentemente da *wavelet* utilizada. Essa consistência indica que o volume de dados está intrinsecamente ligado ao nível de decomposição selecionado, e não à ordem específica da *wavelet* de Daubechies empregada.

5.2.4 Abordagem por Redes Neurais

Optou-se pela implementação de um modelo de rede neural relativamente simples, composto de duas camadas de 256 nós, como mostrado na Figura 5.11. A escolha por 256 nós em cada uma das duas camadas ocultas buscou estabelecer um equilíbrio entre a capacidade de representação do modelo e a complexidade computacional.

Esta arquitetura foi definida após uma série de testes preliminares, nos quais foram avaliados modelos com diferentes números de neurônios, variando de 64 a 512 nós por camada, como mostrado na Tabela 5.17. Verificou-se experimentalmente que arquiteturas com menos de 256 nós apresentavam dificuldade em modelar adequadamente as dependências temporais e não lineares dos dados filtrados, resultando em erros de aproximação significativamente maiores. Em contrapartida, redes com 512 nós ou mais, embora pudessem apresentar uma ligeira redução marginal no erro em alguns cenários específicos, demandavam um tempo de treinamento desproporcionalmente maior e mostravam maior propensão a *overfitting*. Ademais, considerando a necessidade de eventual reconstrução e inferência no lado do cliente, a complexidade adicional de redes maiores representaria um custo computacional potencialmente proibitivo. Portanto, a arquitetura com 256 nós por camada emergiu, tanto dos testes quanto da análise da literatura, como a solução que oferecia o melhor compromisso entre capacidade de aprendizado, eficiência computacional (treinamento e inferência) e risco de *overfitting* para os objetivos desta investigação exploratória.

Em seu treinamento, foi realizado um ciclo de 10.000 épocas com valores randomizados de entrada de forma a buscar cobrir uniformemente as diferentes combinações de variáveis e filtros passíveis de serem selecionados por um usuário com igual probabilidade. Os tensores utilizados foram enviados para processamento na GPU (Unidade de Processamento de Gráficos, do inglês *Graphics Processing Unit*) base-

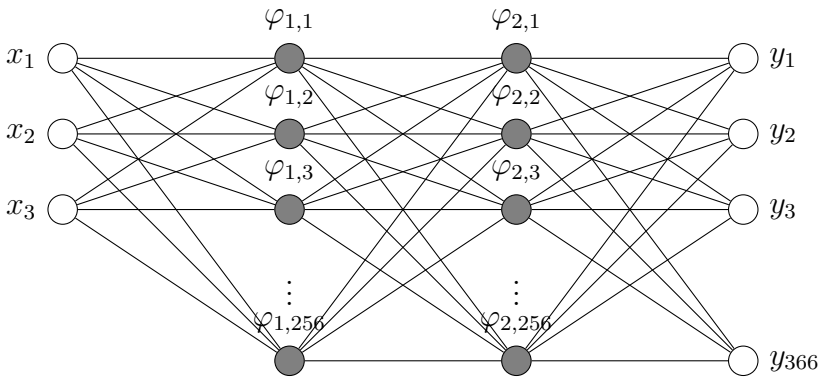


Figura 5.11: Diagrama da arquitetura da rede neural implementada

ada em CUDA (*Compute Unified Device Architecture*) para aceleração do tempo de treinamento e conferindo a essa alternativa uma melhor escalabilidade. A máquina de teste empregou uma placa gráfica NVIDIA GeForce GTX 1660 Ti, com 1536 núcleos e 6 GB de memória VRAM GDDR6.

Um exemplo do resultado alcançado por uma rede neural para a visualização da variável S1 com filtro $N2 < 20$ aplicado pode ser visto na Figura 5.12.

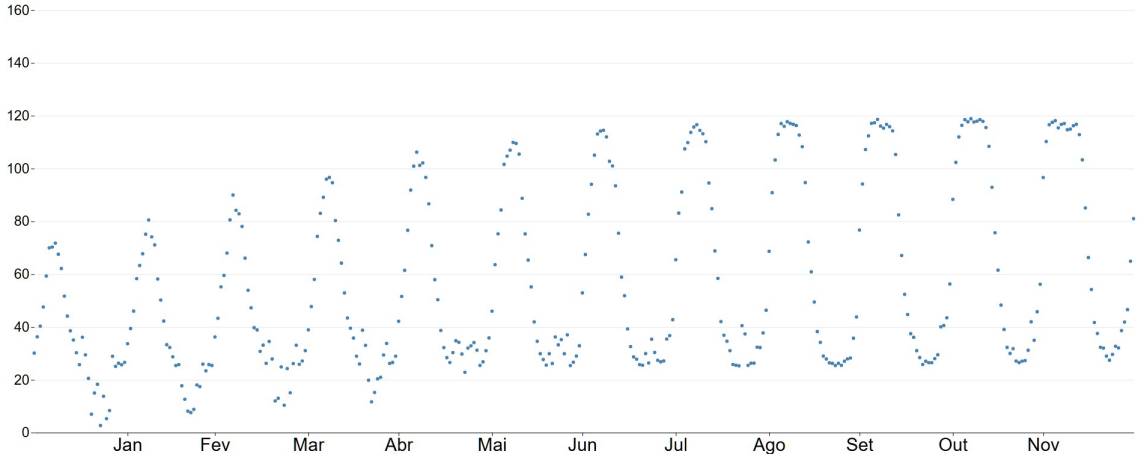


Figura 5.12: Série temporal da média diária da variável S1 com filtro $N2 < 20$ gerada pela rede neural com erro quadrático médio de 211,730

Posteriormente, foi implementada uma otimização de modo a priorizar o treinamento sobre uma variável específica, alcançando um menor erro quadrático médio. Dessa forma, constata-se que o conhecimento das seleções e filtragens mais frequentes pode fundamentar uma estratégia de treinamento mais informada, direcionando o esforço computacional para casos mais prováveis. Em um cenário real, a observação do comportamento dos usuários de modo a identificar as interações com o sistema mais comuns permite a geração de séries temporais estimadas mais acuradas, em média.

Conforme apresentado na Tabela 5.17, os testes conduzidos com dados sintéticos considerando a variável S1 e o filtro $N2 < 50$ demonstraram uma variação signi-

ficativa no erro quadrático médio (MSE) em função do número de nós utilizados nas camadas ocultas. Os resultados indicam que arquiteturas com número reduzido de nós (64 e 128) apresentaram valores de MSE consideravelmente elevados, respectivamente 191,118 e 223,155, apesar do tempo de treinamento relativamente baixo (aproximadamente 61 segundos). Este comportamento evidencia a incapacidade dessas configurações mais simples em capturar adequadamente as dependências temporais e não lineares presentes nos dados sintéticos.

Número de nós	**Tempo de treinamento (s)**	**MSE**	**Volume (kB)**
64	61,422	191,118	112,5
128	61,969	223,155	255,0
256	64,594	7,715	632,5
512	61,969	389,867	1774,0
512 (10x épocas)	703,688	8,948	1774,0

Tabela 5.17: Tabela de medições da rede neural com dados sintéticos considerando a média diária da variável S1 e filtro $N2 < 50$

Por outro lado, observa-se uma redução drástica no erro quadrático médio ao utilizar 256 nós por camada, alcançando um MSE de apenas 7,715, representando uma melhoria de aproximadamente 96% em comparação com as arquiteturas mais simples. Esta significativa redução no erro foi obtida com apenas um ligeiro aumento no tempo de treinamento para 64,594 segundos, demonstrando um excelente compromisso entre capacidade de modelagem e custo computacional.

Curiosamente, como se pode observar na Figura 5.12, o incremento adicional para 512 nós não resultou em melhorias no MSE, pelo contrário, evidenciou-se um expressivo aumento no erro para 389,867. Este comportamento sugere fortemente a ocorrência de overfitting, onde o modelo com maior capacidade parametrizada mostra-se excessivamente ajustado às particularidades da amostra de treinamento, comprometendo sua capacidade de generalização.

Ao aumentar o número de épocas de treinamento por um fator de 10 para a arquitetura de 512 nós, constatou-se uma substancial melhoria no MSE, que foi reduzido para 8,948. No entanto, este ganho na qualidade da aproximação veio acompanhado de um aumento significativo no tempo de treinamento (703,688 segundos), aproximadamente 11 vezes superior ao necessário para a configuração de 256 nós, que oferece um erro ligeiramente menor.

A Figura 5.13 complementa estes achados ao ilustrar o crescimento expressivo no volume de dados transmitidos em função do número de nós, evidenciando o custo adicional em termos de largura de banda e latência potencial para arquiteturas mais complexas. Estes resultados corroboram a escolha da arquitetura de 256 nós por camada como solução ótima para o conjunto de dados sintéticos testado, oferecendo o melhor equilíbrio entre precisão da aproximação, tempo de treinamento e volume

de dados transmitidos.

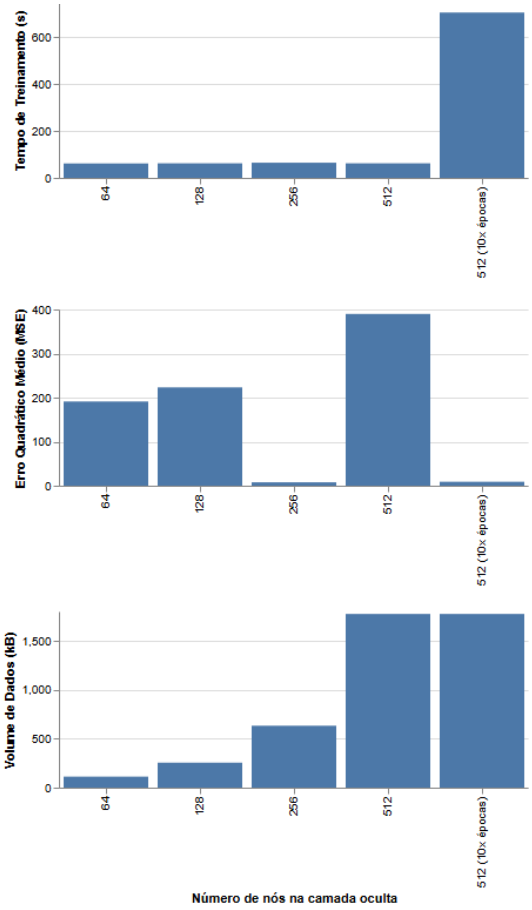


Figura 5.13: Comparação das medições relativas à rede neural com dados sintéticos em relação ao número de nós das camadas ocultas

Como era esperado, modelos mais sofisticados, que envolvem um maior número de neurônios em suas camadas ocultas, demandam mais ciclos de treinamento e, consequentemente, um tempo total maior de treinamento para ajustar o número maior de parâmetros e graus de liberdade que possuem. No entanto, observou-se nos experimentos que mesmo com um tempo de treinamento 10 vezes maior em relação ao modelo de 256 nós, a qualidade da série gerada pelo modelo de 512 nós não foi capaz de superar os modelos mais simples.

Não obstante, outra limitação do uso de modelos cada vez mais complexos nessa arquitetura é a quantidade de informação que necessitaria ser transmitida ao cliente para recriação do modelo local. O volume de dados transmitido cresce em uma velocidade superlinear em relação ao número de nós, tendo a representação do modelo de 512 nós mais de 1 MB de tamanho, como visto na Figura 5.14.

Vale salientar a proporção entre o volume para a construção da rede neural, transmitido uma única vez na inicialização da aplicação de visualização de dados, e o volume para o envio de uma única série temporal simplificada através de FFT. A rede neural mais simples testada demanda 112,5 kB para ser serializada e enviada, número

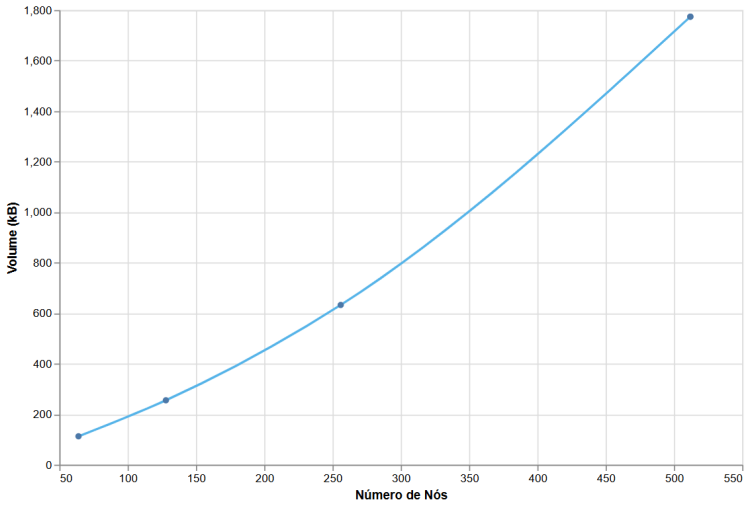


Figura 5.14: Volume dos dados transmitidos vs número de nós das camadas ocultas da rede neural

mais de 15 vezes maior que os 7,3 kB utilizados para representar 50 coeficientes da série transformada pela FFT. Portanto, a aplicação de redes neurais é mais vantajosa em cenários em que o usuário efetua numerosas interações com a ferramenta de visualização, cruzando variáveis e aplicando filtros dezenas de vezes no decorrer do uso.

Assim, um aumento de complexidade do modelo oferece incrementos de qualidade cada vez menos vantajosos, dados os tempos de treinamento mais longos e o consumo e latência de rede maiores. Na prática, é necessário ajustar esse hiperparâmetro de forma a atingir um equilíbrio ideal, ou pelo menos mais aderente às necessidades impostas pela infraestrutura computacional disponível, pela rede, pelos dados e pela interação realizada pelos usuários.

Conjunto de dados reais

Tendo em vista os resultados obtidos com dados sintéticos, para fins de comparação, os experimentos foram realizados variando a quantidade de nós da camada oculta – e, portanto, sua complexidade – e a extensão do treinamento realizado. Assim, podemos verificar como essas duas variáveis se comportam para os dois *datasets*.

A aplicação da abordagem por redes neurais ao conjunto de dados reais proporcionou valiosos *insights* sobre a capacidade desta técnica em modelar séries temporais financeiras. Contemplados pela Tabela 5.18, foram realizados testes sistemáticos utilizando diferentes limiares de filtragem para a variável "open", permitindo avaliar o desempenho do modelo em diferentes condições.

Alguns valores foram constantes para todos os testes efetuados com o conjunto de dados real. O tempo total de treinamento do modelo utilizando os dados reais foi de 274,011 s para a rede neural de 256 nós e 349,469 s para a de 512 nós por

Filtro	MSE	MSE (10x)	MSE (512 nós)
$open < 20$	7924,308	3830,163	4448,082
$open < 50$	5115,766	2070,515	2472,536
$open < 71$ (média)	3932,716	1424,681	1704,078
$open < 100$	2936,962	954,614	1091,675
$open < 200$	1915,654	602,036	505,409
$open < 500$	1448,460	507,302	288,323
$open < 1000$	1106,973	459,482	173,738
$open < 1579$ (máximo)	1008,173	320,565	129,122

Tabela 5.18: Tabela de medições de erros da rede neural com dados reais considerando o preço médio de fechamento de todas as ações

camada oculta. Já o tempo de inferência foi o mesmo do caso sintético, de 15,625 ms. Por fim, o tamanho dos dados enviados ao cliente pelo servidor, ou seja, todos os parâmetros para construção da rede, foi de 632,5 kB e 1174 kB para a rede de 256 nós e para a rede de 512 nós, respectivamente. Isso ocorre pois a topologia da rede é a mesma para os dois conjuntos de dados trabalhados.

Os resultados demonstram uma clara tendência de redução do erro quadrático médio à medida que o valor do filtro aumenta. Para a rede neural padrão com 256 nós por camada, observou-se um MSE de 7924,308 para o filtro mais restritivo ($open < 20$), que diminuiu progressivamente até 1008,173 para o filtro mais abrangente ($open < 1579$, correspondente ao valor máximo). Esta tendência sugere que a rede neural apresenta maior facilidade em modelar comportamentos quando há maior inclusão de dados.

Um aspecto particularmente interessante foi o impacto do aumento no número de épocas de treinamento. A implementação da otimização com treinamento intensificado, ao multiplicar por 10 o número de ciclos de treinamento, observou-se uma redução significativa no MSE em todos os níveis de filtragem. Por exemplo, para o filtro $open < 20$, o erro diminuiu de 7924,308 para 3830,163, representando uma melhoria de aproximadamente 52%. Para o filtro máximo ($open < 1579$), a redução foi ainda mais expressiva, com o MSE caindo para 129,122, uma melhoria de cerca de 87% em relação ao treinamento padrão.

Similarmente, a utilização de uma arquitetura mais complexa com 512 nós por camada também resultou em melhorias consistentes no desempenho. Para o filtro $open < 500$, por exemplo, o MSE foi reduzido para 288,323, em comparação com 1448,460 na configuração padrão, representando uma diminuição de aproximadamente 80% no erro.

A Figura 5.15 ilustra visualmente o resultado da aplicação da rede neural para a série temporal da média dos preços de fechamento com o filtro máximo ($open < 1579$), apresentando um erro quadrático médio de 129,122. Nota-se que, mesmo para

este conjunto de dados complexo e volátil, a rede foi capaz de capturar as tendências principais e oscilações relevantes da série temporal.

Em termos de eficiência computacional, o tempo total de treinamento do modelo utilizando dados reais foi significativamente maior que para dados sintéticos: 274,011 s para a rede de 256 nós e 349,469 s para a de 512 nós por camada. Este aumento reflete a maior complexidade inerente aos padrões presentes em dados financeiros reais em comparação com dados sintéticos. No entanto, o tempo de inferência permaneceu constante em 15,625 ms, demonstrando que, uma vez treinada, a rede neural pode gerar séries temporais estimadas rapidamente, independentemente da complexidade dos dados subjacentes.

Estes resultados corroboram a viabilidade da utilização de redes neurais como uma estratégia eficiente para a visualização progressiva de séries temporais financeiras, especialmente quando se considera a possibilidade de transferir o modelo treinado para o cliente, permitindo a geração local das séries temporais com baixa latência e alta fidelidade aos dados originais.

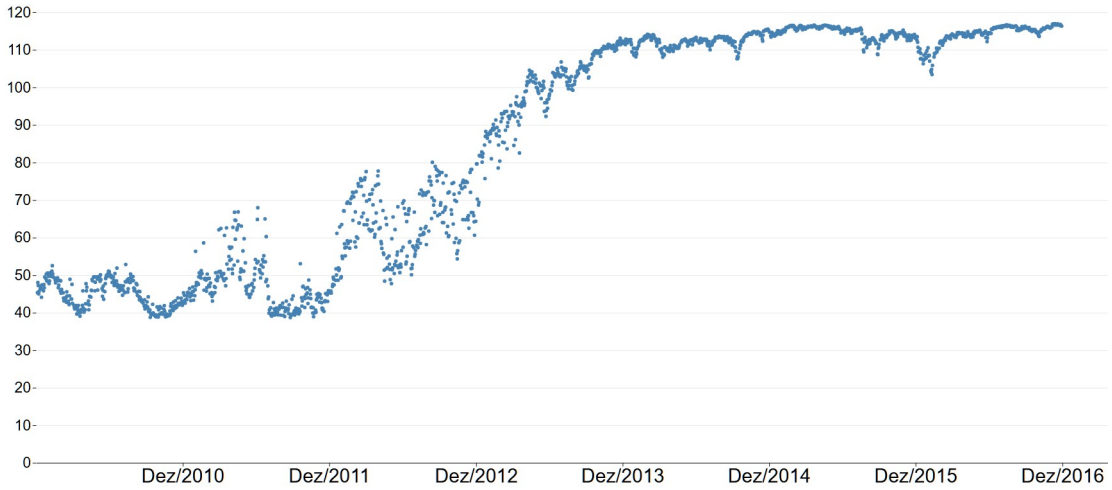


Figura 5.15: Série temporal da média dos preços de fechamento com aplicação do filtro *open* < 1579 gerada pela rede neural com erro quadrático médio de 129,122

5.3 Discussão dos resultados

Os resultados obtidos a partir dos experimentos realizados com dados sintéticos permitem uma análise comparativa das diferentes abordagens implementadas, destacando suas vantagens e limitações em termos de tempo de processamento, cobertura, erro quadrático médio e volume. Cada técnica apresentou características específicas que a tornam mais ou menos adequada dependendo do contexto de aplicação e das prioridades estabelecidas, como velocidade de resposta ou acurácia da série temporal gerada.

A abordagem clássica, que utiliza o conjunto completo de dados para o cálculo das séries temporais, serviu como referência para avaliar as demais técnicas. Embora forneça resultados exatos, seu tempo de processamento elevado a torna inviável em cenários com grandes volumes de dados, especialmente em aplicações que demandam respostas rápidas e interativas.

Já a experimentação com a estratégia de amostragem para implementação da progressividade revelou que a escolha do algoritmo é capaz de impactar significativamente na eficiência e na qualidade das séries temporais aproximadas enviadas de forma antecipada pelo servidor ao cliente. Em uma análise comparativa, observa-se que a amostragem randômica mais simples, apesar de se mostrar rápida, de fácil implementação e nenhuma necessidade de pré-processamento no servidor, apresentou, em média, maior erro quadrático médio entre os algoritmos testados. Além disso, por selecionar dados de maneira completamente aleatória e sem direcionamento algum, pode deixar de capturar sazonalidades importantes e ou até mesmo tendências mais discretas.

Em contraste, a amostragem agrupada, seja por janelas de tempo ou por similaridade, demonstrou uma melhoria significativa na cobertura temporal, garantindo que todas as partes do intervalo de tempo desejado fossem representadas na amostra e, consequentemente, na visualização da série temporal. Os resultados indicaram uma redução no erro quadrático médio, evidenciando uma progressividade com prévias da série temporal mais próxima da série real. As implementações baseadas em algoritmos mais sofisticados, como KNN e DBSCAN, apresentaram os melhores resultados dentre essas técnicas. Ao priorizar pontos próximos em termos de similaridade, o uso do KNN foi capaz de capturar nuances localizadas que a amostragem puramente randômica não conseguiu. Por sua vez, a capacidade própria do DBSCAN de identificar regiões densas e desconsiderar *outliers* permitiu a geração de amostras mais representativas e, com isso, de menor erro quadrático médio. Em contrapartida, o tempo de preparação do modelo do OPTICS, executada uma vez na inicialização do servidor, foi o mais demorado, bem como os tempos de processamento para cada requisição.

Apesar do baixo MSE, o algoritmo OPTICS não seria a escolha ideal para uma exploração inicial extremamente rápida. Seu alto tempo refere-se principalmente ao custo de gerar a ordenação hierárquica dos pontos pela primeira vez. Porém, em uma aplicação real com capacidade de *caching*, esse passo mais demorado pode ser executado como um pré-processamento, logo após o carregamento de um conjunto de dados e antes mesmo da primeira interação do usuário. Uma vez que essa estrutura hierárquica está pré-calculada e armazenada, as consultas subsequentes do usuário para gerar a amostra se tornam muito mais rápidas, pois o passo mais custoso já foi realizado. Dessa forma, é possível combinar a alta qualidade da resposta parcial via

OPTICS com uma experiência de uso fluida para o usuário final.

A abordagem baseada em transformadas de Fourier representou uma técnica muito eficaz e eficiente em gerar séries temporais aproximadas com diferentes níveis de precisão. Em cenários reais, ela pode ser utilizada para o carregamento progressivo conforme uma configuração do usuário ou de maneira programática, recebendo um número inicial de coeficientes e refinando a visualização à medida que novos coeficientes são recebidos pelo cliente de maneira assíncrona. Ao transmitir apenas os coeficientes mais representativos, foi possível reduzir significativamente o volume de dados enviados pela rede, mantendo uma boa aproximação da série temporal original. Essa técnica mostrou-se especialmente vantajosa em cenários onde a largura de banda é um fator limitante.

Por fim, a abordagem baseada em redes neurais destacou-se pela capacidade de gerar séries temporais aproximadas de forma extremamente rápida após o treinamento inicial. No entanto, o custo computacional do treinamento e a necessidade de dados representativos para garantir a eficácia do modelo são fatores que limitam sua aplicabilidade em alguns contextos. Ainda assim, a possibilidade de transferir o modelo treinado para o cliente, permitindo a geração local das séries temporais, representa uma vantagem significativa em termos de latência e escalabilidade.

De maneira geral, a escolha da técnica mais adequada depende do contexto de aplicação e das prioridades estabelecidas. Para cenários que demandam respostas rápidas e com menor custo computacional, a amostragem agrupada e as transformadas de Fourier se destacam como opções robustas. Já em situações que exigem maior acurácia ou que envolvem padrões complexos nos dados, técnicas baseadas em agrupamento ou aprendizado de máquina podem oferecer melhores resultados, desde que os recursos computacionais necessários estejam disponíveis.

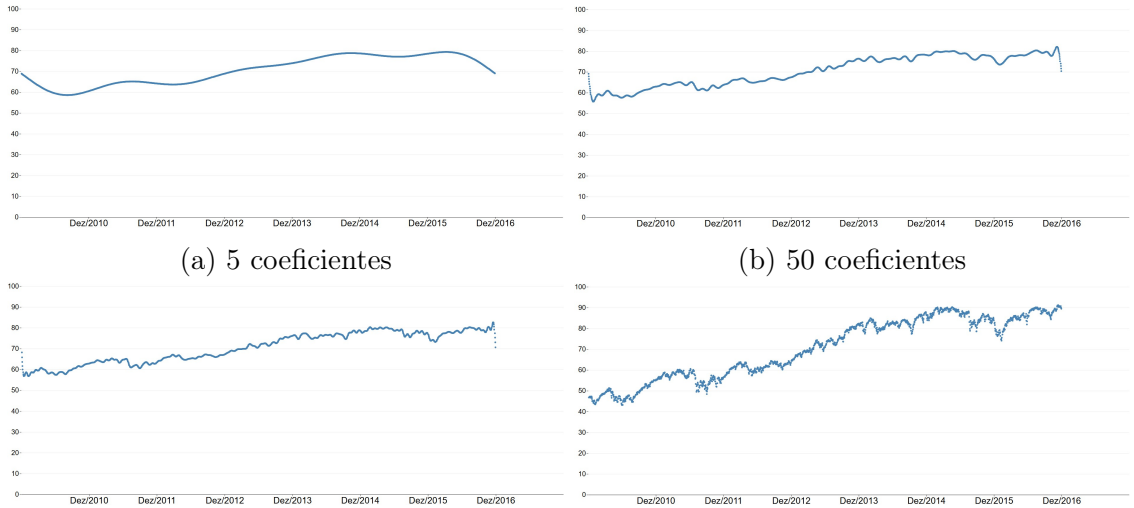
A validação das abordagens propostas foi estendida ao conjunto de dados real de preços de ações do SP500, com o intuito de verificar a generalização dos achados obtidos com os dados sintéticos. Para simular cenários de consulta ad-hoc, característicos da exploração interativa, os testes não se limitaram à aplicação das técnicas sobre o agregado de todas as ações, mas também envolveram subconjuntos de dados gerados a partir da aplicação de filtros exemplificativos.

A criação dos filtros a serem aplicados também se deu randomicamente, selecionando uma das colunas e um valor numérico a fim de servir como limiar de corte para variáveis numéricas ou uma categoria-alvo para variáveis categóricas. Foram selecionados, por exemplo, filtros baseados em setor de atuação das empresas (como Tecnologia da Informação, Saúde etc.) e data de listagem no índice, utilizando os atributos disponíveis no referido conjunto de dados.

Os resultados obtidos a partir da aplicação das técnicas de amostragem e aprendizado de máquina no conjunto de dados real de preços de ações do SP500 foram

compatíveis com as observações feitas anteriormente com dados sintéticos. Essa consistência entre os resultados reforça a validade das abordagens propostas e sua aplicabilidade em cenários práticos.

Durante a análise, foi possível observar que as técnicas de amostragem, como a amostragem agrupada e as abordagens baseadas em agrupamento, mantiveram sua eficácia na redução do erro médio quadrático (MSE) e na melhoria da representatividade das amostras geradas. Os resultados indicaram que, mesmo em um conjunto de dados complexo e dinâmico como o de preços de ações, as técnicas de amostragem foram capazes de capturar tendências e fornecer estimativas precisas das séries, o que pode ser percebido na Figura 5.5.



(a) 5 coeficientes

(b) 50 coeficientes

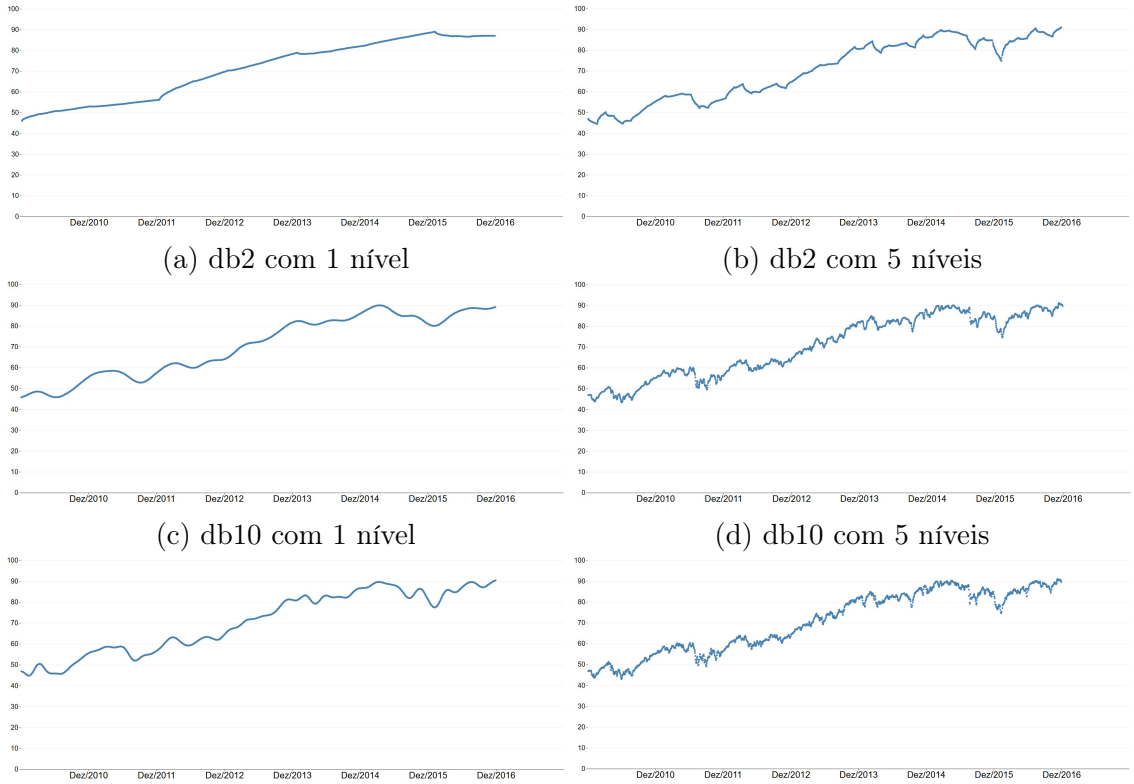
(c) 100 coeficientes

(d) Série exata

Figura 5.16: Comparação entre séries reconstruídas a partir da inversa da FFT com 2, 10 e 50 coeficientes e série exata da média geral do preço de fechamento com dados reais

No que tange ao uso de FFT e *wavelets*, essa técnica também foi capaz de gerar boas estimativas da série temporal real para os dados de ações da bolsa de valores, como visto na Figura 5.16 e na Figura 5.17. Com a transmissão apenas dos coeficientes mais significativos da transformada no domínio da frequência, foi possível reconstruir uma série temporal aproximada capaz de representar a tendência de crescimento do preço médio das ações e flutuações locais, por exemplo. Demonstra-se assim a aplicabilidade real dessa otimização na redução do tamanho da resposta do servidor a ser transmitida pela rede e consequente redução da latência de rede experimentada pelo usuário no uso da ferramenta de visualização de dados.

Podemos observar que *wavelets* descritas com maiores números de *vanishing points* oferecem uma maior precisão mesmo com apenas um nível de detalhamento. No entanto, diferentemente das senoides utilizadas pela FFT, as *wavelets* requerem o uso de coeficientes para o cálculo da própria *wavelet-mãe*, além dos coeficientes de



(a) db2 com 1 nível

(b) db2 com 5 níveis

(c) db10 com 1 nível

(d) db10 com 5 níveis

(e) db20 com 1 nível

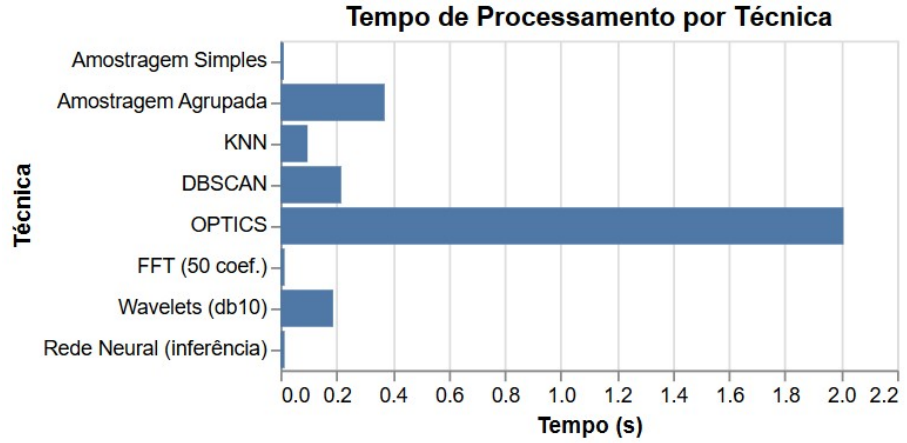
(f) db20 com 5 níveis

Figura 5.17: Comparação entre séries reconstruídas a partir da inversa da transformada de *wavelets* de Daubechies com diferentes números de coeficientes e níveis de detalhe para a média geral dos preços de fechamento do conjunto de dados reais

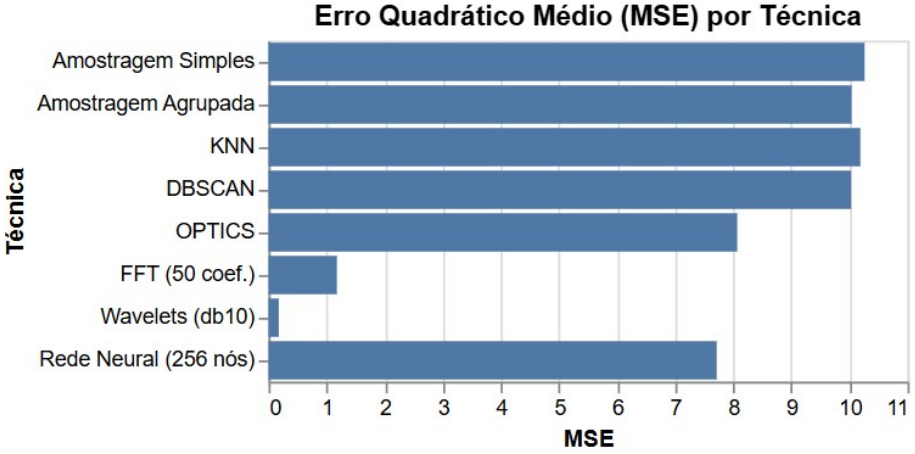
tempo e escala calculados e transmitidos relativos à série temporal decomposta. Tais valores podem ser transmitidos uma única vez ao cliente e reutilizados localmente a cada consulta feita pelo usuário, transmitindo pela rede apenas o necessário: os coeficientes de aproximação ou de detalhes da série temporal em si.

A comparação entre os resultados obtidos com dados reais e sintéticos revelou que as técnicas desenvolvidas são eficazes em diferentes contextos, o que é um indicativo de sua versatilidade e robustez. Isso sugere que os modelos podem ser utilizados em uma variedade de cenários do mundo real, desde a análise de mercado financeiro até outras áreas que envolvem séries temporais. Em suma, os resultados obtidos com dados reais validam as abordagens propostas e destacam a importância de utilizar conjuntos de dados diversificados para testar a eficácia das estratégias propostas.

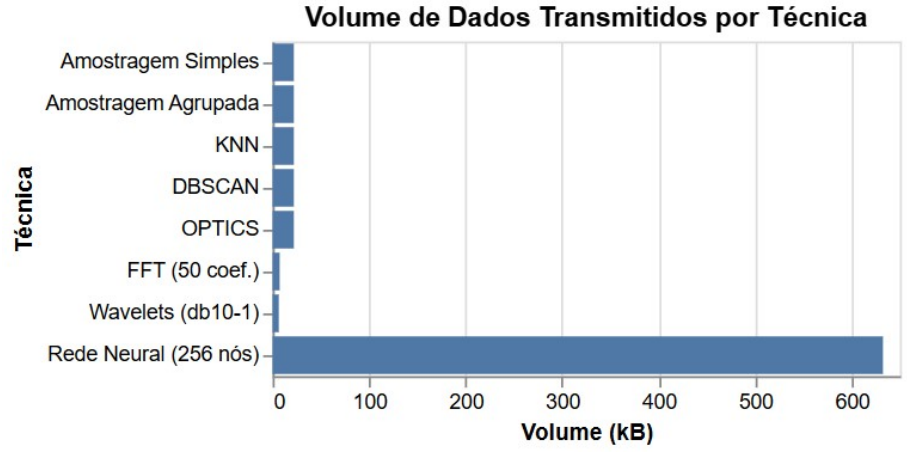
Para fundamentar a seleção da técnica mais adequada em diferentes contextos de aplicação, faz-se necessária uma análise mais rigorosa dos *trade-offs* entre as principais métricas avaliadas neste trabalho. As relações de compromisso entre tempo de processamento, erro quadrático médio, volume de dados transmitidos e cobertura temporal para cada técnica implementada revelam padrões, sintetizados na Tabela 5.19 e na Figura 5.18 que podem orientar decisões de projeto em sistemas de visualização progressiva.



(a) Tempo de processamento



(b) Erro quadrático médio



(c) Volume transmitido

Figura 5.18: Comparação geral entre técnicas testadas de acordo com diferentes métricas

Técnica	**Tempo de Processamento**	**Erro Quadrático Médio**	**Volume de Dados**	**Cobertura Temporal**	**Cenário Ótimo de Aplicação**
Amostragem Simples	Muito Baixo	Alto	Baixo	Variável	Exploração inicial rápida
Amostragem Agrupada	Médio	Médio	Baixo	Alta	Séries com pontos locais relevantes
KNN	Baixo	Médio	Baixo	Alta	Dados multivariados em geral
DBSCAN	Médio	Médio	Baixo	Alta	Dados com distribuição não uniforme
OPTICS	Alto	Baixo	Baixo	Alta	Alta precisão com suporte a pré-processamento
Transformada de Fourier	Muito Baixo*	Variável**	Baixo	Alta	Séries periódicas ou suaves
Transformada de *Wavelet*	Muito Baixo*	Variável**	Muito Baixo	Alta	Rede limitada, séries longas
Rede Neural	Muito Baixo (somente inferência)***	Variável****	Alto, porém único	Alta	Múltiplas consultas repetidas

* Depende principalmente do tamanho do conjunto de dados

** Depende fortemente da natureza da série temporal

*** Tempo de inferência: 15,625 ms; Treinamento: 64-274 s

**** Depende da complexidade do modelo e do filtro aplicado

Tabela 5.19: Matriz de *trade-offs* entre técnicas de visualização progressiva

A análise comparativa entre os métodos de amostragem demonstra que a amostragem simples oferece o menor tempo de processamento, com ganhos de velocidade de até 95% em comparação com técnicas mais sofisticadas como o OPTICS. Nos experimentos com o conjunto de dados de 100.000 linhas, a amostragem simples com 10% dos dados processou a série temporal em apenas 0,013 s, enquanto o OPTICS demandou 2,009 s para a mesma taxa de amostragem. No entanto, essa vantagem em velocidade é contrabalançada por um erro quadrático médio 28% maior (10,530 versus 8,069). A amostragem agrupada por data e o DBSCAN representam pontos intermediários neste espectro, oferecendo tempos de processamento de 0,372 s e 0,217 s, respectivamente, com erros quadráticos médios aproximadamente 4% e 2% menores que a amostragem simples.

A abordagem por transformada de Fourier apresenta uma relação custo-benefício notável em termos de volume de dados transmitidos. Utilizando apenas 50 coeficientes, observou-se uma redução de aproximadamente 63% no tamanho da resposta (7,3 kB versus 20 kB da série completa) com um erro quadrático médio de apenas 1,177 para a variável S1, representando uma precisão 89% superior à obtida com 10% de amostragem simples. Esta vantagem é ainda mais pronunciada para séries com forte componente periódica, como demonstrado pela diferença de desempenho entre as variáveis S1 e G1. Para esta última, o mesmo número de coeficientes resultou em um erro quadrático médio 56,8 vezes maior (66,441), evidenciando a dependência desta técnica em relação às características intrínsecas da série temporal.

A inclusão das transformadas de *wavelets* nos experimentos revela uma importante ressalva em relação à transformada de Fourier. Enquanto a FFT demonstrou um custo-benefício notável para séries com forte periodicidade e continuidade, como a variável S1, sua eficiência decai para sinais com transientes abruptos ou características não estacionárias. Este fenômeno ocorre porque as funções de base da FFT (senos e cossenos) são globais, exigindo um grande número de coeficientes para modelar eventos localizados, como as descontinuidades presentes na variável G1.

Em contrapartida, as *wavelets*, por serem localizadas no tempo e na frequência, mostraram-se mais aptas a representar tais eventos de forma compacta e precisa. A decomposição multi-nível permite que um subconjunto reduzido de coeficientes de *wavelets* capture tanto as tendências de baixa frequência (aproximação) enviadas rapidamente quanto os picos e descontinuidades de alta frequência (detalhe) enviados posteriormente, resultando em um erro de aproximação potencialmente menor para séries complexas. Portanto, em um sistema de visualização progressiva, a seleção da transformada não deve ser estática; uma abordagem adaptativa, que avalie a estacionariedade ou a presença de transientes na série de dados solicitada, poderia selecionar dinamicamente entre Fourier e *wavelets* para otimizar o compromisso entre o volume de dados transmitido e a fidelidade da representação visual preliminar.

Quanto às redes neurais, o custo computacional inicial de treinamento é substancial – 64,594 s para a rede com 256 nós por camada – porém o tempo de inferência é extremamente reduzido, apenas 15,625 ms, representando uma aceleração de 86% em relação à amostragem simples. O modelo treinado intensivamente (10x épocas) para dados reais mostrou uma redução de 68% no erro quadrático médio em comparação com o treinamento padrão (320,565 versus 1008,173), justificando o investimento em treinamento mais extenso para aplicações que demandam maior precisão. O volume de dados transmitido para a reconstrução do modelo no cliente (632,5 kB) é significativo, porém trata-se de uma transmissão única que elimina a necessidade de novas transferências para consultas subsequentes.

A análise cruzada dessas métricas permite identificar cenários de aplicação ótimos para cada técnica. A amostragem simples é mais adequada para explorações iniciais rápidas, oferecendo uma visão geral da série temporal com latência mínima. A transformada de Fourier apresenta vantagem decisiva em ambientes com restrições de largura de banda, especialmente para séries temporais com componentes periódicos bem definidos. Os algoritmos baseados em agrupamento como DBSCAN e OPTICS, apesar de seu maior custo computacional, justificam-se em cenários onde a precisão é primordial e a distribuição dos dados é não uniforme, com regiões de diferentes densidades que precisam ser adequadamente representadas na amostra.

O modelo baseado em redes neurais, por sua vez, demonstra maior eficiência em sessões de exploração prolongadas, nas quais o usuário realiza múltiplas consultas dentro do mesmo espaço de parâmetros. O tempo inicial de transmissão do modelo e seu custo de treinamento são amortizados pela velocidade de resposta às consultas subsequentes, resultando em uma experiência de usuário mais fluida ao longo do tempo. Este comportamento é especialmente valioso em ambientes analíticos corporativos, nos quais os analistas tipicamente exploram variações de uma mesma consulta durante sessões estendidas.

5.4 Escalabilidade

Um aspecto relevante a ser considerado na implementação de sistemas de visualização progressiva em ambientes de produção é a abordagem de escalabilidade adotada frente ao crescimento contínuo do volume de dados. Neste contexto, torna-se necessário avaliar como as diferentes técnicas experimentadas neste trabalho comportam-se sob estratégias de escalabilidade vertical (aumento de recursos computacionais em um único nó) e horizontal (distribuição do processamento entre múltiplos nós).

As técnicas baseadas em amostragem demonstraram alta adequação à escalabilidade horizontal, uma vez que o processo de amostragem pode ser naturalmente distribuído entre diferentes nós computacionais através de estratégias de particio-

namento dos dados. Particularmente, algoritmos como DBSCAN e KNN podem ser implementados em variantes distribuídas, permitindo que o processamento dos agrupamentos ocorra paralelamente em diferentes partições dos dados. Já o OPTICS, apesar de sua maior precisão, apresenta desafios adicionais para paralelização devido à natureza sequencial de sua construção do ordenamento de alcançabilidade.

Por outro lado, a abordagem baseada em transformadas beneficia-se significativamente da escalabilidade vertical, especialmente de recursos como unidades de processamento vetorial e GPUs, que aceleram consideravelmente operações matemáticas complexas. A FFT, por exemplo, pode ser adaptada para contextos distribuídos, mas incorre em sobrecarga de comunicação entre nós que pode comprometer seu desempenho em cenários de alta latência de rede. Nos testes realizados, observou-se que o tempo de processamento da FFT aumentou em proporção significativamente menor que o aumento do volume de dados (141,45 vezes para um aumento de 1.000 vezes no tamanho dos dados), evidenciando sua escalabilidade sublinear favorável. De forma similar, a transformada de *wavelet* discreta também tira proveito da aceleração por hardware e apresenta uma vantagem teórica em eficiência, com uma complexidade de tempo menor em comparação com a FFT, a posicionando como uma alternativa altamente escalável.

Por fim, a abordagem por redes neurais apresenta comportamento híbrido quanto à escalabilidade. O processo de treinamento beneficia-se tanto da escalabilidade vertical, através da aceleração por GPUs como demonstrado nos experimentos com CUDA, quanto da escalabilidade horizontal, mediante técnicas de treinamento distribuído. Em contraste, a inferência, etapa crítica para a geração de aproximações em tempo real, é altamente paralelizável e escalável horizontalmente, permitindo que diferentes nós atendam simultaneamente a distintas requisições de usuários.

5.5 Limitações

Algumas limitações foram identificadas no decorrer dos testes. Primeiramente, a utilização de dados sintéticos, embora permita maior controle sobre as características dos dados e facilite a experimentação, pode não refletir completamente os desafios encontrados em cenários reais. O mundo real frequentemente apresenta peculiaridades, como padrões de sazonalidade complexos, *outliers* inesperados e distribuições não uniformes, que podem impactar o desempenho das técnicas implementadas.

Além disso, as abordagens propostas foram testadas em um ambiente controlado, com recursos computacionais específicos e condições de rede estáveis. Em aplicações reais, fatores como limitações de hardware, variações na latência da rede e concorrência por recursos podem influenciar significativamente os resultados obtidos.

Outro ponto a ser considerado é a dependência de hiperparâmetros em algu-

mas das técnicas implementadas, como o número de *clusters* no KNN ou o valor de épsilon no DBSCAN. A escolha inadequada desses parâmetros pode comprometer a representatividade das amostras e, consequentemente, a acurácia das séries temporais geradas. Embora esforços tenham sido feitos para otimizar esses valores, a generalização para diferentes conjuntos de dados e contextos de aplicação ainda requer investigação adicional.

No entanto, o ajuste de hiperparâmetros também pode, por sua vez, ser automatizado e otimizado. Através de técnicas de otimização, como algoritmos genéticos, otimização bayesiana e até a busca exaustiva em *grid search*, os melhores valores de, por exemplo, k para o KNN podem ser calculados com precisão em uma série de treinamentos e melhorias incrementais. Apesar de ser computacionalmente mais custoso e demandar maior tempo de preparação desses modelos no servidor, esse processo só precisa ser realizado uma vez para cada conjunto de dados, o que é facilmente compensado pelos melhores resultados para todas as consultas subsequentes pelo cliente. Em uma ferramenta em produção, os hiperparâmetros encontrados neste estudo podem também ser utilizados como valores iniciais para explorações prévias enquanto valores ótimos são buscados em segundo plano.

Ademais, o deslocamento parcial do processamento para o cliente, característica central de algumas estratégias progressivas, implica na transmissão de artefatos que nem sempre são triviais do ponto de vista de segurança da informação. Sob a ótica da confidencialidade, um vetor de exposição que merece atenção surge na transmissão do modelo de rede neural.

Na abordagem por redes neurais, todos os parâmetros (pesos e vieses) do modelo são serializados e enviados uma única vez ao *front-end*. Esses parâmetros, embora não contenham registros individuais, podem possibilitar ataques de *"model inversion"* ("inversão de modelo") [51] ou "membership inference" ("inferência de pertencimento") [52], nos quais um agente mal-intencionado busca extrair, total ou parcialmente, características estatísticas do conjunto de treinamento. Para mitigar tais riscos, técnicas de treinamento com ruído podem ser consideradas em implementações futuras.

Por fim, a implementação de técnicas de aprendizado de máquina, como redes neurais, apresenta desafios relacionados ao custo computacional do treinamento e à necessidade de dados representativos para garantir a eficácia do modelo. Em cenários com dados escassos ou altamente heterogêneos, a performance dessas técnicas pode ser limitada, exigindo estratégias complementares ou alternativas.

Adicionalmente, é imperativo reconhecer que a própria construção dos conjuntos de dados sintéticos poderia ser aprimorada. Em particular, as variáveis S1 e G1 foram concebidas com fortes componentes periódicos e sazonais. Essa característica favorece, por natureza, as abordagens baseadas em transformadas, como a FFT e as

wavelets. Consequentemente, a eficácia notável dessas técnicas na reconstrução de tais séries pode não ser representativa de seu desempenho em dados com estruturas temporais mais complexas ou não periódicas, resultando em uma avaliação potencialmente otimista de seu custo-benefício em relação aos métodos de amostragem ou de aprendizado de máquina.

A criação de conjuntos de dados sintéticos com características temporais mais heterogêneas e não periódicas também seria de grande valia. A inclusão de variáveis com distribuições que se assemelhem a decaimentos logarítmicos, processos estocásticos como passeios aleatórios (*random walks*), tendências com lei de potência ou mesmo séries temporais com comportamento caótico criaria um ambiente de teste mais desafiador e generalista.

Essas limitações ressaltam a necessidade de estudos futuros que explorem a aplicação das técnicas discutidas em contextos mais diversos e realistas, bem como a investigação de métodos híbridos que combinem diferentes abordagens para superar os desafios identificados.

Capítulo 6

Conclusões

Neste trabalho, foram apresentadas algumas técnicas para visualização progressiva de séries temporais, partindo de sua conceituação em nível mais abstrato até sua implementação e subsequente teste em um cenário simulado com dados sintéticos.

Um dos principais gargalos em termos de tempo de resposta para visualizações de grandes volumes de dados multidimensionais é o tempo de processamento para o cálculo das agregações a serem exibidas na série temporal, muitas vezes agrupadas em diferentes intervalos de tempo e com diferentes filtros aplicados a depender da exploração realizada pelo usuário. Tendo em vista que um dos principais casos de uso para esse tipo de ferramenta é o da exploração de diferentes recortes e agregações dos dados (seja em uma exploração livre ou orientada à geração de algum tipo de *insight*), tempos de carregamento demasiadamente elevados podem comprometer drasticamente a experiência de uso da visualização e, em casos mais extremos, até impedir o atingimento de seus objetivos em tempo hábil.

Por isso, a habilidade de computar rapidamente resultados que possam ser interpretados o quanto antes pelos usuários é de grande importância e tende a se tornar cada vez mais importante com o surgimento de bases de dados cada vez mais numerosas e mais complexas, como no contexto de *big data* e da Internet das Coisas.

A partir de uma base de dados gerada proceduralmente, foram realizados experimentos com diversos algoritmos para geração de séries temporais aproximadas de diferentes variáveis e seus tempos de execução e erros médios de modo a comparar velocidade e acurácia de tais abordagens. Através da discussão do funcionamento de cada algoritmo, pode-se compreender vantagens e desvantagens de seus usos e optar por diferentes técnicas em diferentes cenários a depender da natureza dos dados visualizados. Os resultados não apresentam uma solução definitiva para obtenção de soluções de visualização de dados com *big data* velozes e acuradas, mas, sim, um leque de ferramentas disponíveis para alcançar uma experiência de usuário fluida e responsiva nesse tipo de aplicação.

Os resultados obtidos com o conjunto de dados financeiros reais reforçam a apli-

cabilidade prática das técnicas discutidas, demonstrando sua capacidade de lidar com grandes volumes de informações e de gerar *insights* relevantes para a tomada de decisões financeiras. Essa validação prática destaca a importância de utilizar dados reais para avaliar a eficácia de métodos analíticos, garantindo que as soluções propostas não sejam apenas teoricamente sólidas, mas, sim, úteis em cenários do mundo real.

Adicionalmente, as técnicas discutidas poderiam ser aplicadas a contextos mais complexos, como o do projeto Amplia Saúde, a ferramenta exploratória de visualização de dados sobre natalidade e saúde materna e neonatal que serviu de motivação para o desenvolvimento deste estudo [5]. A aplicação dessas abordagens no Amplia Saúde permitiria otimizar o carregamento progressivo e a interação com os dados, viabilizando uma experiência mais fluida para os usuários e facilitando a análise de séries temporais relacionadas à saúde materna, neonatal e aos impactos da poluição do ar. Isso ampliaria o potencial da ferramenta em gerar *insights* valiosos para pesquisadores e demais interessados na área da saúde.

Assim, este trabalho demonstrou, ao dissecar diferentes gargalos e validar experimentalmente um repertório de estratégias, que a latência inerente à exploração de grandes volumes de dados temporais não é uma barreira intransponível, mas um desafio que pode ser superado com a aplicação criteriosa de técnicas de visualização progressiva. O próprio sistema de visualização de dados pode, ainda, guiar a escolha das técnicas de progressividade mais apropriadas, inferindo características da análise a partir da requisição do usuário. Por exemplo, a criticidade de estimativas mais precisas pode ser inferida a partir dos filtros aplicados ou janelas de tempo selecionadas mais curtas, preferindo o uso de técnicas de amostragem mais precisas; enquanto restrições de rede podem ser detectadas automaticamente, optando por ativar técnicas baseadas em transformadas matemáticas.

6.1 Trabalhos Futuros

Muitas são as possibilidades de aprofundamento neste tema, bem como de expansão para assuntos correlatos. Primeiramente, existem diversos outros possíveis algoritmos a serem implementados e testados em todas as abordagens aqui descritas, bem como o estudo mais aprofundado da combinação de diferentes técnicas, como o uso de amostragem e simplificações através de FFT concomitantemente. Ou ainda, podem ser aplicadas outras transformadas matemáticas, como as transformadas baseadas em outras *wavelets* definidas de forma customizada.

Outro potencial desdobramento deste trabalho é a investigação de mais técnicas envolvendo inteligência artificial e aprendizado de máquina para a geração de séries temporais aproximadas, empregando, por exemplo, a arquitetura de *autoencoders*, e

até mesmo para segmentação dos dados para transmissão do servidor para o cliente. Com o tema de inteligência artificial em alta e com destaque que tende a crescer ainda mais no futuro próximo, este pode ser um momento alvissareiro para estudos nessa temática. Técnicas baseadas em amostragem não aleatória e em outras transformadas, como a transformada de cossenos discreta, também podem ser experimentadas em busca de outros algoritmos com potencial de oferecer uma redução do erro quadrático médio ainda mais substancial.

Neste contexto, o uso de estruturas de dados baseadas no particionamento recursivo do domínio, de forma análoga ao que algoritmos como as *quadtrees* alcançam no campo de processamento de imagens, apresenta-se como uma linha de investigação particularmente relevante. Sua lógica pode ser adaptada para o tema de séries temporais, onde os eixos representariam o tempo e o valor da variável de interesse, permitindo particionar a série temporal em quadrantes e armazenar informações agregadas (como médias, mínimos e máximos) em nós internos de uma árvore. Consultas para um determinado nível de detalhe poderiam, então, ser respondidas rapidamente ao inspecionar apenas os nós da árvore até a profundidade correspondente, evitando a varredura da totalidade dos pontos de dados. Tal abordagem poderia hibridizar os benefícios das otimizações de processamento e transmissão, oferecendo uma representação multirresolução nativa dos dados.

Para além da otimização dos algoritmos de processamento, uma vertente promissora para trabalhos futuros reside na exploração de técnicas que se fundamentem na estrutura inerente dos próprios dados temporais. As abordagens investigadas nesta dissertação tratam o conjunto de dados como uma coleção de registros a serem processados ou amostrados. Uma nova perspectiva consistiria em impor uma estrutura hierárquica sobre o domínio dos dados, permitindo a agregação e a consulta em múltiplos níveis de detalhe de forma mais eficiente.

Podemos destacar como outro estudo a se realizar a experimentação com otimizações do lado do cliente, em especial no que tange ao sistema de renderização da visualização de dados no navegador. É possível, por exemplo, explorar alternativas que contem com o poder de processamento paralelo da GPU para processar e renderizar certas requisições utilizando *shaders* e a tecnologia WebGL. Ou ainda, parte dos dados pode ser armazenada junto ao cliente para geração de respostas rápidas estimadas e ser combinada a dados vindos do servidor posteriormente, construindo uma representação local dos dados conforme a demanda do usuário por diferentes filtragens, variáveis e agrupamentos, ou ponderando as aproximações calculadas pelo cliente e pelo servidor com pesos específicos.

Além disso, a realização de testes com usuários em futuros estudos ou projetos pode trazer benefícios significativos. Esses testes, organizados em sessões para a observação de usuários interagindo com ferramentas que usem as melhorias aqui

propostas, permitirão avaliar a usabilidade das visualizações, identificar áreas de melhoria e garantir que as soluções propostas atendam às necessidades reais dos usuários. Um dos principais objetivos desses testes será validar a interação fluida que a progressividade permite, assegurando que os usuários consigam navegar e interagir com os dados de forma intuitiva e eficiente; ou ainda, identificar melhorias de design que evitem erros de interpretação e vieses cognitivos de ancoragem devido às estimativas carregadas, utilizando, por exemplo, faixas que representem o possível erro associado à curva mostrada.

Por fim, testes A/B podem ser realizados para comparar diferentes abordagens de progressividade e identificar quais delas oferecem a melhor experiência ao usuário. Questionários e entrevistas pós-teste também podem ser utilizados para aprofundar a compreensão das experiências dos usuários e suas percepções sobre a eficácia da progressividade na visualização de dados. Métricas como tempo de resposta, taxa de sucesso nas tarefas realizadas e nível de satisfação dos participantes podem ser analisadas para embasar decisões sobre ajustes e melhorias. A análise dos dados coletados permitirá ajustes e melhorias contínuas na ferramenta, garantindo que ela se torne cada vez mais alinhada às expectativas e necessidades dos usuários. Esses testes não apenas validarão as abordagens implementadas, como também fornecerão *insights* valiosos para o desenvolvimento de novas funcionalidades e para a adaptação das técnicas a diferentes contextos de aplicação, em um futuro em que se farão cada vez mais necessárias.

Referências Bibliográficas

[1] ANGELINI, M., SANTUCCI, G., SCHUMANN, H., et al. "A review and characterization of progressive visual analytics". In: *Informatics*, v. 5, p. 31. MDPI, 2018.

[2] INSTITUTE, B. "Covid-19 surveillance dashboard", *University of Virginia*, 2020. Disponível em: <https://nssac.bii.virginia.edu/covid-19/dashboard/>.

[3] RAMASWAMY, H. . "sentiment viz: Tweet Sentiment Visualization", *Department of Computer Science at NC State University*, 2022. Disponível em: <https://www.csc2.ncsu.edu/faculty/healey/tweet_viz/>.

[4] GILBERT, J., BRODERSEN, R. "Globally progressive interactive web delivery". In: *IEEE INFOCOM '99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No.99CH36320)*, v. 3, pp. 1291–1299 vol.3, 1999. doi: 10.1109/INFCOM.1999.752147.

[5] KOSMINSKY, D., ESPERANÇA, C., ILLARRAMENDI, X., et al. "Amplia Saúde: Observatório dos períodos pré- e perinatal", *Projeto Amplia Saúde*, 2022. Disponível em: <https://ampliasaude.org/>.

[6] FINK, E., GANDHI, H. S. "Compression of time series by extracting major extrema", *Journal of Experimental & Theoretical Artificial Intelligence*, v. 23, n. 2, pp. 255–270, 2011. doi: 10.1080/0952813X.2010.505800. Disponível em: <https://doi.org/10.1080/0952813X.2010.505800>.

[7] ESLING, P., AGON, C. "Time-series data mining", *ACM Comput. Surv.*, v. 45, n. 1, dez. 2012. ISSN: 0360-0300. doi: 10.1145/2379776.2379788. Disponível em: <https://doi.org/10.1145/2379776.2379788>.

[8] KOHONEN, T. "The self-organizing map", *Proceedings of the IEEE*, v. 78, n. 9, pp. 1464–1480, 1990. doi: 10.1109/5.58325.

[9] BARRETO, G. A. "Time series prediction with the self-organizing map: A review", *Perspectives of neural-symbolic integration*, pp. 135–158, 2007.

[10] FANG, Y., XU, H., JIANG, J. "A Survey of Time Series Data Visualization Research", *IOP Conference Series: Materials Science and Engineering*, v. 782, n. 2, pp. 022013, mar 2020. doi: 10.1088/1757-899X/782/2/022013. Disponível em: <https://dx.doi.org/10.1088/1757-899X/782/2/022013>.

[11] JUGEL, U., JERZAK, Z., HACKENBROICH, G., et al. "M4: a visualization-oriented time series data aggregation", *Proc. VLDB Endow.*, v. 7, n. 10, pp. 797–808, jun. 2014. ISSN: 2150-8097. doi: 10.14778/2732951.2732953. Disponível em: <https://doi.org/10.14778/2732951.2732953>.

[12] MARTIN, S., QUACH, T.-T. "Interactive visualization of multivariate time series data". In: *International Conference on Augmented Cognition*, pp. 322–332. Springer, 2016.

[13] SARAIYA, P., LEE, P., NORTH, C. "Visualization of graphs with associated timeseries data". In: *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.*, pp. 225–232, 2005. doi: 10.1109/INFVIS.2005.1532151.

[14] JAYASANKAR, U., THIRUMAL, V., PONNURANGAM, D. "A survey on data compression techniques: From the perspective of data quality, coding schemes, data type and applications", *Journal of King Saud University-Computer and Information Sciences*, v. 33, n. 2, pp. 119–140, 2021.

[15] LELEWER, D. A., HIRSCHBERG, D. S. "Data compression", *ACM Computing Surveys (CSUR)*, v. 19, n. 3, pp. 261–296, 1987.

[16] STORER, J. A. *Data compression: methods and theory*. Estados Unidos da América, Computer Science Press, Inc., 1987. ISBN: 0716781565.

[17] ANDREWS, C., DAVIES, J., SCHWARZ, G. "Adaptive data compression", *Proceedings of the IEEE*, v. 55, n. 3, pp. 267–277, 1967.

[18] SALOMON, D. *Data compression: the complete reference*. Berlin, Heidelberg, Springer-Verlag, 2004. ISBN: 1846286026.

[19] GIORGI, G. "A combined approach for real-time data compression in wireless body sensor networks", *IEEE Sensors Journal*, v. 17, n. 18, pp. 6129–6135, 2017.

[20] LEHMANN, H., JUNG, B. "In-situ multi-resolution and temporal data compression for visual exploration of large-scale scientific simulations". In: *2014 ieee 4th symposium on large data analysis and visualization (ldav)*, pp. 51–58. IEEE, 2014.

[21] SENGUPTA, N., KASABOV, N. "Spike-time encoding as a data compression technique for pattern recognition of temporal data", *Information Sciences*, v. 406, pp. 133–145, 2017.

[22] KRISHNAN, K., MARCELLIN, M. W., BILGIN, A., et al. "Efficient transmission of compressed data for remote volume visualization", *IEEE transactions on medical imaging*, v. 25, n. 9, pp. 1189–1199, 2006.

[23] CHIUEH, T.-C., YANG, C.-K., HE, T., et al. "Integrated volume compression and visualization". In: *Proceedings. Visualization'97 (Cat. No. 97CB36155)*, pp. 329–336. IEEE, 1997.

[24] GERSTNER, T. "Multiresolution visualization and compression of global topographic data", *GeoInformatica*, v. 7, n. 1, pp. 7–32, 2003.

[25] AILAMAKI, A., PANDIS, I. "Query Processor". In: LIU, L., ÖZSU, M. T. (Eds.), *Encyclopedia of Database Systems*, pp. 2307–2308, Boston, MA, Springer US, 2009. ISBN: 978-0-387-39940-9. doi: 10.1007/978-0-387-39940-9_676. Disponível em: <https://doi.org/10.1007/978-0-387-39940-9_676>.

[26] DING, B., HUANG, S., CHAUDHURI, S., et al. "Sample+ seek: Approximating aggregates with distribution precision guarantee". In: *Proceedings of the 2016 International Conference on Management of Data*, pp. 679–694, 2016.

[27] LI, J. K., MA, K.-L. "P5: Portable progressive parallel processing pipelines for interactive data analysis and visualization", *IEEE transactions on visualization and computer graphics*, v. 26, n. 1, pp. 1151–1160, 2019.

[28] ULMER, A., SESSLER, D., KOHLHAMMER, J. "NetCapVis: Web-based Progressive Visual Analytics for Network Packet Captures". In: *2019 IEEE Symposium on Visualization for Cyber Security (VizSec)*, pp. 1–10, 2019. doi: 10.1109/VizSec48167.2019.9161633.

[29] KNUTH, D. E. "Dynamic huffman coding", *Journal of Algorithms*, v. 6, n. 2, pp. 163–180, 1985. ISSN: 0196-6774. doi: https://doi.org/10.1016/0196-6774(85)90036-7. Disponível em: <https://www.sciencedirect.com/science/article/pii/0196677485900367>.

[30] VITTER, J. S. "Design and Analysis of Dynamic Huffman Codes", *J. ACM*, v. 34, n. 4, pp. 825–845, oct 1987. ISSN: 0004-5411. doi: 10.1145/31846.42227. Disponível em: <https://doi.org/10.1145/31846.42227>.

[31] LI, Z., DREW, M., LIU, J. *Fundamentals of Multimedia*. Texts in Computer Science. Suíça, Springer International Publishing, 2014. ISBN: 9783319052908. Disponível em: <https://books.google.com.br/books?id=R6vBBAAAQBAJ>.

[32] MARPE, D., SCHWARZ, H., WIEGAND, T. "Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard", *IEEE Transactions on Circuits and Systems for Video Technology*, v. 13, n. 7, pp. 620–636, 2003. doi: 10.1109/TCSVT.2003.815173.

[33] PINCUS, S. M. "Approximate entropy as a measure of system complexity." *Proceedings of the National Academy of Sciences*, v. 88, n. 6, pp. 2297–2301, 1991.

[34] CHIAROT, G., SILVESTRI, C. "Time Series Compression Survey", *ACM Comput. Surv.*, v. 55, n. 10, feb 2023. ISSN: 0360-0300. doi: 10.1145/3560814. Disponível em: <https://doi.org/10.1145/3560814>.

[35] GOYAL, V. K., FLETCHER, A. K., RANGAN, S. "Compressive Sampling and Lossy Compression", *IEEE Signal Processing Magazine*, v. 25, n. 2, pp. 48–56, 2008. doi: 10.1109/MSP.2007.915001.

[36] CHANDAK, S., TATWAWADI, K., WEN, C., et al. "LFZip: Lossy Compression of Multivariate Floating-Point Time Series Data via Improved Prediction". In: *2020 Data Compression Conference (DCC)*, pp. 342–351, 2020. doi: 10.1109/DCC47342.2020.00042.

[37] MOON, A., PARK, J., SONG, Y. J. "Prediction of Compression Ratio for Transform-based Lossy Compression in Time-series Datasets". In: *2022 24th International Conference on Advanced Communication Technology (ICACT)*, pp. 142–146, 2022. doi: 10.23919/ICACT53585.2022.9728954.

[38] GOTTLIEB, D., SHU, C.-W. "On the Gibbs Phenomenon and Its Resolution", *SIAM Review*, v. 39, n. 4, pp. 644–668, 1997. doi: 10.1137/S0036144596301390.

[39] LOAN, C. V. "3. High-Performance Frameworks". In: *Computational Frameworks for the Fast Fourier Transform*, pp. 121–187, Philadelphia, PA, USA, Society for Industrial and Applied Mathematics, 1987. doi:

10.1137/1.9781611970999.ch3. Disponível em: <https://epubs.siam.org/doi/abs/10.1137/1.9781611970999.ch3>.

[40] COOLEY, J. W., TUKEY, J. W. "An algorithm for the machine calculation of complex Fourier series", *Mathematics of computation*, v. 19, n. 90, pp. 297–301, 1965.

[41] RHIF, M., BEN ABBES, A., FARAH, I. R., et al. "Wavelet Transform Application for/in Non-Stationary Time-Series Analysis: A Review", *Applied Sciences*, v. 9, n. 7, 2019. ISSN: 2076-3417. doi: 10.3390/app9071345. Disponível em: <https://www.mdpi.com/2076-3417/9/7/1345>.

[42] EBY, P. J. *PEP 333 – Python Web Server Gateway Interface v1.0*. PEP 333, 2003. Disponível em: <https://peps.python.org/pep-0333/>.

[43] EBY, P. J. *PEP 3333 – Python Web Server Gateway Interface v1.0.1*. PEP 3333, 2010. Disponível em: <https://peps.python.org/pep-3333/>.

[44] FETTE, I., MELNIKOV, A. *The websocket protocol*. Relatório técnico, 2011.

[45] TEAM, T. A. *ASGI Documentation*. Relatório técnico, 2018. Disponível em: <https://asgi.readthedocs.io/en/latest/>.

[46] CURRIER, C. "Protocol Buffers". In: Hummert, C., Pawlaszczyk, D. (Eds.), *Mobile Forensics – The File Format Handbook: Common File Formats and File Systems Used in Mobile Devices*, pp. 223–260, Cham, Springer International Publishing, 2022. ISBN: 978-3-030-98467-0. doi: 10.1007/978-3-030-98467-0_9. Disponível em: <https://doi.org/10.1007/978-3-030-98467-0_9>.

[47] LIU, Z., SAIFULLAH, Y., GREIS, M., et al. "HTTP compression techniques". In: *IEEE Wireless Communications and Networking Conference, 2005*, v. 4, pp. 2495–2500 Vol. 4, 2005. doi: 10.1109/WCNC.2005.1424906.

[48] LU, J., LIU, A., DONG, F., et al. "Learning under Concept Drift: A Review", *IEEE Transactions on Knowledge and Data Engineering*, v. 31, n. 12, pp. 2346–2363, 2019. doi: 10.1109/TKDE.2018.2876857.

[49] JORDON, J., SZPRUCH, L., HOUSSIAU, F., et al. "Synthetic Data – what, why and how?" 2022. Disponível em: <https://arxiv.org/abs/2205.03257>.

[50] LANDAU, H. "Sampling, data transmission, and the Nyquist rate", *Proceedings of the IEEE*, v. 55, n. 10, pp. 1701–1706, 1967. doi: 10.1109/PROC.1967.5962.

[51] SHOKRI, R., STRONATI, M., SONG, C., et al. “Membership Inference Attacks against Machine Learning Models”. 2017. Disponível em: <https://arxiv.org/abs/1610.05820>.

[52] ZHOU, Z., ZHU, J., YU, F., et al. “Model Inversion Attacks: A Survey of Approaches and Countermeasures”. 2024. Disponível em: <https://arxiv.org/abs/2411.10023>.