Regresso: uma Ferramenta para Ajuste de Curvas na Análise de Algoritmos

Vinicius Sathler[1], **Jorge Goulart**[1], **Bruno Porto Masquio**[1]
Paulo Eustáquio Duarte Pinto[1]

[1]Instituto de Matemática e Estatística – Universidade do Estado do Rio de Janeiro (UERJ)
Rio de Janeiro, RJ – Brasil

vinicius.sathler2@gmail.com, jorge.goulart@uerj.br,

brunomasquio@ime.uerj.br, pauloedp@ime.uerj.br

Abstract. *This article introduces Regresso, an open-source web tool developed to help the analysis of algorithm complexity. Featuring a Python-based back-end and an Angular front-end, the tool allows for the analysis of algorithm execution data using non-linear regression methods, with the goal of supporting the teaching of Algorithms and Data Structures.*

Resumo. *Este artigo apresenta o Regresso, uma ferramenta web de código aberto desenvolvida para facilitar a análise de complexidade de algoritmos. Utilizando um back-end em Python e um front-end em Angular, a ferramenta permite a análise de dados amostrais de execução de algoritmos por meio de métodos de regressão não linear com o fim de oferecer suporte para o ensino de Algoritmos e Estruturas de Dados.*

1. Introdução

O estudo de complexidade de algoritmos é um aspecto fundamental na Ciência da Computação e tem sido explorado há décadas [Cormen et al. 2022, Szwarcfiter 2018, Dasgupta et al. 2006]. Compreender a eficiência de um algoritmo, tanto em termos de tempo de execução quanto de memória necessária, é crucial para determinar a viabilidade e a eficácia de soluções computacionais, influenciando diretamente a abordagem escolhida para resolver um problema.

Por exemplo, para ordenar um conjunto grande de dados, se utilizarmos um algoritmo ineficiente como o BubbleSort, com complexidade $\mathcal{O}(n^2)$, o tempo para concluir a ordenação poderia estender-se por vários dias. Em contraste, ao optar por um algoritmo mais eficiente, como o HeapSort ou MergeSort, com complexidade de $\mathcal{O}(n \log n)$, a mesma tarefa poderia ser realizada em apenas 1 hora.

Diante desse fato, a análise de complexidade de algoritmos é um assunto essencial no ensino dos cursos Ciência da Computação, Sistemas de Informação e correlatos. Tal

Cadernos do IME - Série Informática
e-ISSN: 2317-2193 (online)
DOI: 10.12957/cadinf.2024.87951

aprendizado pode ser facilitado pelo uso de ferramentas empíricas e visuais, pois facilitam a assimilação de conceitos teóricos por meio de experimentação prática e de visualização gráfica.

Dentre as ferramentas encontradas para esse fim, destacamos PlanetCalc [PlanetCalc 2024], Wolfram Alpha [Wofram Research 2024], Desmos [Desmos 2024] e a aplicação EMA [de Souza Oliveira 2024]. Apesar de serem aplicações robustas, observamos que são pouco intuitivas e precisam de um tempo considerável para aprender a usá-las, tornando seu uso infrutífero em turmas de algoritmos.

Nesse contexto, apresentamos o *Regresso*, aplicação web, livre, de código aberto, gratuita e desenvolvida com Python e Angular pelo aluno Vinicius Sathler de Ciência da Computação da Universidade do Estado do Rio de Janeiro (UERJ). O objetivo é disponibilizar uma ferramenta simples e gratuita para auxílio na análise de complexidade de algoritmos.

A ferramenta permite que os usuários insiram pares ordenados e obtenham a função que melhor se ajusta aos dados — seja ela linear, quadrática, exponencial, logarítmica, entre outras — além de fornecer os gráficos correspondentes. Regresso torna simples o processo de identificação do melhor ajuste para os dados obtidos, possibilitando uma análise intuitiva e pouco trabalhosa comparada com outras ferramentas disponíveis.

Ao longo deste artigo, tomamos como referências os livros *Introduction to Algorithms* [Cormen et al. 2022] e *Teoria computacional de grafos: os Algoritmos* [Szwarcfiter 2018], para os conceitos relativos à complexidade de algoritmos. Ademais, como base para tópicos em estatística, fundamentamo-nos mo livro *Probability and Statistics* [DeGroot and Schervish 2012].

O restante do trabalho é organizado da seguinte forma. Na Seção 2, introduzimos o estudo de ajuste de curvas e apresentamos outras ferramentas utilizadas para ajuste de curvas. Na Seção 3, descrevemos a ferramenta Regresso, mostrando características visuais, da sua arquitetura, e de sua implementação. Na Seção 3, demonstramos o uso da ferramenta a partir de testes com alguns algoritmos clássicos. A Seção 4 é destinada a estudos comparativos que fizemos entre o Regresso e as ferramentas descritas na Seção 2. Finalmente, apresentamos nossas conclusões e sugestões de trabalhos futuros na Seção 5.

2. Ajuste de Curvas e Ferramentas

2.1. Ajuste de Curvas

O ajuste de curvas é uma técnica essencial em matemática e estatística, amplamente empregada para modelar e compreender o comportamento de conjuntos de dados. Ao contrário da interpolação, que visa encontrar com precisão uma função que passe exatamente pelos pontos dados, o ajuste de curvas busca uma função que minimize a discrepância entre os valores observados e os valores previstos pela função ajustada [Kiusalaas 2013].

Existem várias técnicas para realizar o ajuste de curvas, sendo a mais comum o *método dos mínimos quadrados* (**MMQ**). Nesse método, busca-se encontrar a equação da curva que minimiza a soma dos quadrados das diferenças entre os valores observados e os valores previstos pela curva ajustada. Isso geralmente envolve escolher uma função

paramétrica, sendo a linear a mais comum, e ajustar os parâmetros dessa função para otimizar o ajuste aos dados [Kolb 1984].

Escrevendo em notação matemática, o **MMQ** busca resolver a seguinte expressão:

$$\min_{\theta} \sum_{i=1}^{n} (y_i - f(x_i; \theta))^2$$

Para resolver tal expressão podemos encontrar as derivadas parciais para cada parâmetro da função e igualá-las a zero.

$$\frac{\partial}{\partial \theta_j} \sum_{i=1}^{n} (y_i - f(x_i; \theta))^2 = 0, \quad j = 1, 2, \ldots, m$$

Onde m é o número de parâmetros na função f [Dekking et al. 2005].

No presente trabalho estamos interessados em algumas funções que regularmente aparecem no contexto de análise de algoritmos. Essas funções devem explicar a variação dos recursos usados pela execução do algoritmo, sendo o tempo o principal deles, em função do tamanho da entrada desse algoritmo. Ao longo do artigo, denotamos por n o tamanho da entrada do algoritmo analisado. As principais funções consideradas estão descritas na Tabela 1. Além dessas funções, também é de interesse avaliar expressões menos comuns, mas ainda dentro do escopo de possibilidades de funções que descrevem a complexidade de um algoritmo.

Ajuste	Equação da Curva
Linear	$f(x) = ax + b$
Quadrático	$f(x) = ax^2 + bx + c$
Cúbico	$f(x) = ax^3 + bx^2 + cx + d$
Logarítmico	$f(x) = a * log_2(x) + b$
Linearítmico	$f(x) = a * x * log_2(x) + b$
Linearítmico modificado 1	$f(x) = a * x * log_2(x)^2 + b$
Linearítmico modificado 2	$f(x) = a * x^2 * log_2(x) + b$
Potência	$f(x) = a * x^b$

Tabela 1. Regressões utilizadas no Regresso.

2.1.1. Interpretação dos Resultados

Após ajustar uma curva aos dados, é importante interpretar os resultados. Isso inclui, principalmente, a avaliação da qualidade do ajuste. No Regresso foram adotados os principais avaliadores do ajuste de curvas citados na literatura [Draper and Smith 1998]. São os seguintes:

- Coeficiente de determinação (R^2): é uma medida que indica a proporção da variância total da variável dependente que é explicada pelo modelo. O R^2 normalmente varia entre 0 e 1, onde 0 indica que o modelo não explica nenhuma variação e 1 indica que o modelo explica toda a variação.

- Coeficiente de correlação de Pearson (ρ): esse valor varia no intervalo $\{-1, 1\}$ e mede a intensidade e a direção da relação linear. O valor -1 indica que a variável dependente varia de forma inversa à dependente e o valor 1 que a relação é direta. Valor 0 significa um ajuste muito ruim.
- Raiz Quadrada do Erro Médio: Também conhecida como *Root Mean Square Error* (RMSE). É a raiz quadrada da média do quadrado das diferenças entre os valores observados e os valores obitidos. Em termos simples, o RMSE fornece uma estimativa do desvio padrão dos erros residuais, ou seja, da diferença entre os valores previstos e os valores reais. Quanto menor for o valor do RMSE, mais precisa é a modelagem do fenômeno em questão.
- Erro médio relativo ($\overline{A}$): É a média dos valores relativos dos erros entre os valores observados e os valores previstos pelo modelo expressos em porcentagens. O $\overline{A}$ fornece uma medida da magnitude média dos erros de previsão. Quanto menor o valor do $\overline{A}$, melhor o ajuste do modelo.

2.2. Ajuste de Curvas em Python

A linguagem de programação Python é amplamente reconhecida e utilizada nas áreas de ciência de dados e aprendizado de máquina. Ela oferece uma vasta gama de bibliotecas especializadas nesses tópicos, como NumPy, Pandas e SciPy. Esta última foi de grande utilidade para o desenvolvimento do Regresso, pois disponibiliza uma ampla variedade de funcionalidades para otimização e ajuste de curvas.

O SciPy oferece um módulo especializado para problemas de otimização e ajuste, o scipy.optimize. Esse módulo inclui métodos para ajuste de curvas que utilizam algoritmos avançados, como o algoritmo de Levenberg-Marquardt e o de Região de Confiança Reflexiva.

2.3. Ferramentas para Ajuste de Curvas

Diversas ferramentas computacionais são amplamente utilizadas para a realização de ajustes de curvas em problemas de regressão. Entre as mais reconhecidas e acessíveis estão o *PlanetCalc*, o *Wolfram Alpha* e o *Desmos*. A seguir, procederemos à descrição dessas ferramentas.

2.3.1. *PlanetCalc*

PlanetCalc é uma coleção de calculadoras online que disponibiliza uma ampla gama de ferramentas para cálculos matemáticos e científicos, incluindo recursos para ajuste de curvas [PlanetCalc 2024]. A plataforma oferece uma interface que possibilita aos usuários comparar diferentes resultados de ajustes, além de visualizar as curvas calculadas de forma sobreposta, o que facilita tanto a análise quanto a interpretação dos dados.

Em comparação com outras ferramentas mais sofisticadas, como *Wolfram Alpha* e *Desmos*, o *PlanetCalc* ainda apresenta algumas limitações. Embora ofereça uma boa representação gráfica dos resultados, carece de suporte para ajustes de curvas mais específicos, como aqueles comumente utilizados na análise de algoritmos. Como exemplo, citamos curvas logarítmicas na base dois, essenciais em áreas como análise de complexidade computacional, ou a combinação de tais curvas com outras funções, como a definida por $f(x) = x \cdot \log_2(x)$, não estão plenamente suportadas na plataforma.

Em síntese, o *PlanetCalc* é uma ferramenta ótima para ajustes de curvas básicos e intermediários, com uma interface acessível, que facilita a comparação de resultados. No entanto, para usuários que necessitam de funcionalidades mais avançadas, como ajustes de curvas logarítmicas específicas ou combinações complexas de funções, plataformas como *Wolfram Alpha* e *Desmos* podem oferecer maior flexibilidade. Em seguida, apresentamos a ferramenta *Wolfram Alpha*.

2.3.2. *Wolfram Alpha*

Desenvolvido pela Wolfram Research, o *Wolfram Alpha* é uma ferramenta computacional avançada que integra uma extensa base de conhecimento com sofisticados algoritmos de computação para responder a uma vasta gama de consultas formuladas em linguagem natural [Wofram Research 2024]. Sua capacidade abrange a geração de respostas computacionais para questões complexas em diversas áreas do conhecimento científico, oferecendo soluções detalhadas e fundamentadas em dados rigorosos.

Especificamente, no que diz respeito ao ajuste de curvas, o *Wolfram Alpha* oferece a capacidade de configurar uma variedade de funções padrão, além de permitir a criação de funções personalizadas. Ele apresenta os parâmetros de avaliação e os coeficientes resultantes do ajuste, facilitando a análise e compreensão dos dados.

Consideramos que a principal desvantagem das ferramentas *Wolfram Alpha* e *PlanetCalc*, a quantidade limitada de curvas pré-definidas comuns à análise de algoritmos, e, apesar de oferecerem uma solução para funções personalizadas, apresentam pouca informação sobre o ajuste. Outro questão crucial é que, apesar de apresentarem os cálculos completos dos seus ajustes, elas exigem assinatura no serviço, limitando o alcance da ferramenta para o público geral e para ser usado em cursos universitários. Em seguida, apresentamos a ferramenta *Desmos*.

2.3.3. *Desmos*

O *Desmos* é uma ferramenta gratuita desenvolvida pela *Desmos Studio*, oferecendo uma ampla gama de recursos para cálculos e visualização de gráficos. Essa calculadora gráfica possui a capacidade de realizar ajustes com qualquer curva especificada pelo usuário utilizando um método especial para lidar com ajustes de curva não-lineares. Tal método consiste na seleção de múltiplas estimativas iniciais para os coeficientes, visando, de forma heurística, obter o melhor resultado. Essa abordagem é especialmente eficaz em situações em que o método dos mínimos quadrados enfrenta dificuldades de convergência, proporcionando resultados satisfatórios.

Além de sua capacidade de ajuste de curvas, o *Desmos* é uma ferramenta poderosa para auxiliar na análise da complexidade de algoritmos. No entanto, é importante notar que sua representação dos parâmetros de avaliação é limitada, fornecendo apenas o coeficiente de determinação. Embora seja possível calcular os outros parâmetros, suas fórmulas devem ser explicitamente inseridas pelo usuário, tornando essa tarefa mais trabalhosa.

2.3.4. EMA

A ferramenta EMA (**EM**pirical **A**nalysis of algorithms) foi desenvolvida pelo Professor Fabiano de Souza Oliveira, da Universidade do Estado do Rio de Janeiro e está descrita no site [de Souza Oliveira 2024] e na tese [Juliana Macario de Souza 2018]. O EMA tem como objetivo analisar empiricamente a complexidade de algoritmos, tomando como base execuções iterativas sobre a implementação dos mesmos. Nesse processo, a ferramenta armazena os respectivos tempos de execução, memórias alocadas e as quantidades de outros recursos personalizados que requeiram monitoramento, em função de uma variável de entrada do algoritmo.

Para a análise de um algoritmo pela ferramenta EMA, devem ser implementados dois programas: (i) o algoritmo a ser analisado e (ii) um *script* de geração de entradas para tal algoritmo. Uma execução da ferramenta pode ser dividida em três etapas: calibração, simulação e análise, que são descritas a seguir.

Na etapa de calibração, a ferramenta recebe um *script* que tem o propósito de gerar entradas para o algoritmo a ser testado. Na etapa de simulação, o conjunto de pontos escolhidos na etapa anterior são processados em sequência para a coleta de dados, medindo o consumo dos recursos. Essa fase objetiva estimar, usando o método dos mínimos quadrados, uma função $f(x)$ que tem, como formato geral, a seguinte.

$$f_{\text{geral}}(x) = a_0 x^{a_1} a_2^{x^{a_3} \log x^{a_4}} \log^{a_5} x + a_6, \text{ com } a_0, a_2 > 0$$

As funções ajustadas pelo método de mínimos quadrados são classificadas em três grupos:

- **Função de erro mínimo:** função, dentre todas as candidatas, com o menor erro;
- **Funções equivalentes:** funções cujos erros não ultrapassam um limite percentual do erro da função de erro mínimo (tipicamente, 0.5%). Funções desse grupo são consideradas, para efeitos práticos, equivalentes por não diferirem significativamente para o conjunto de pontos testados;
- **Melhor-palpite:** função que faz parte do grupo de funções equivalentes e classificada, dentro dos critérios do EMA, como a mais provável de corresponder à real. A ideia que motiva os critérios do EMA nesta escolha é conhecida como navalha de Occam (*"Among competing hypotheses, the one with the fewest assumptions should be selected."*).

Após a etapa de análise, a ferramenta apresenta, em forma de texto, as funções retornadas dentro de cada grupo, cabendo ao usuário analisar os resultados.

3. Descrição da Ferramenta Regresso

Neste capítulo, descreveremos a ferramenta Regresso. Detalharemos a arquitetura da aplicação e discutiremos seus casos de uso, oferecendo uma melhor compreensão de como esta ferramenta pode auxiliar estudantes da Ciência da Computação. Vamos mostrar o funcionamento interno do Regresso e explorar as suas funcionalidades.

3.1. Arquitetura da aplicação

A aplicação Regresso foi estruturada em dois módulos independentes: o back-end e o front-end. Essa abordagem modularizada permite uma gestão mais eficiente do código, facilitando atualizações e manutenções futuras. Além disso, essa separação clara de responsabilidades entre o back-end, responsável pelo processamento de requisições, e o front-end, responsável pela interface com o usuário, melhora a escalabilidade e a flexibilidade da aplicação.

Para garantir uma infraestrutura escalável e robusta, o back-end do Regresso foi hospedado nos serviços da *Amazon Web Services* (AWS). A escolha da AWS proporciona uma série de benefícios, incluindo escalabilidade automática, alta disponibilidade e facilidade de gerenciamento, permitindo que a aplicação atenda às variadas demandas de uso e simplifique questões de infraestrutura.

O módulo *front-end* foi desenvolvido utilizando Angular, um *framework* de código aberto conhecido por sua agilidade, segurança e facilidade de manutenção.

Para hospedar a página foi utilizado o *GitHub Pages*, um serviço gratuito oferecido pelo *GitHub* para hospedagem de sites estáticos. Essa escolha foi feita visando reduzir os custos de manutenção da aplicação, além de garantir um certificado SSL, aprimorando a segurança das requisições entre os módulos da aplicação. A ferramenta encontra-se disponibilizada em https://sathler.github.io/Regresso-Page/.

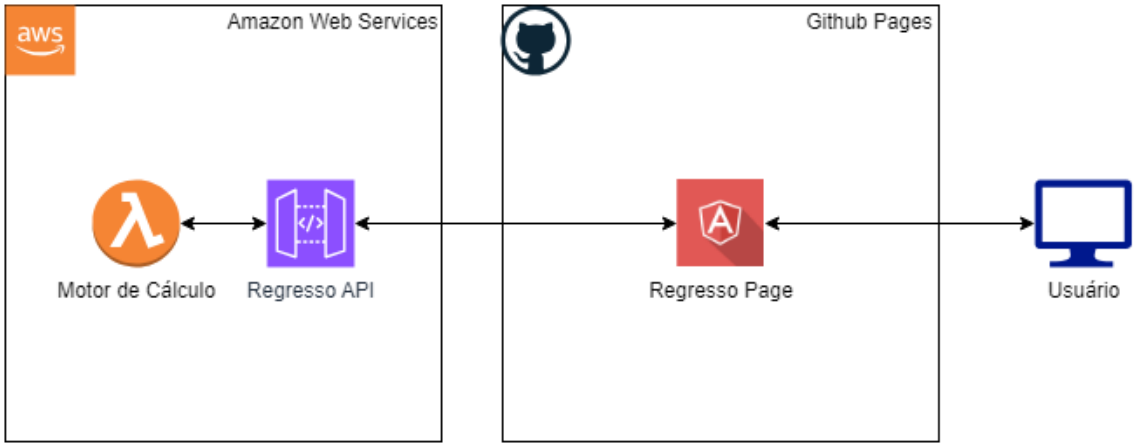

Figura 1. Diagrama da arquitetura do Regresso.

3.2. Regresso API

3.2.1. Características

O núcleo de nossa aplicação, denominado “motor de cálculo”, consiste em um programa em Python que utiliza as bibliotecas SciPy e NumPy para realizar os cálculos numéricos necessários na resolução dos problemas de ajuste de curvas.

A implementação é simples: o motor recebe no corpo da requisição a curva pré-definida para a qual deve tentar o ajuste e retorna os coeficientes calculados, juntamente com os parâmetros de avaliação.

Dado que a quantidade de curvas possíveis de serem analisadas pode ser imprevisível no contexto de complexidade algorítmica, o motor também aceita curvas personalizadas. Para isso, exige-se uma expressão que represente essa curva em linguagem Python. Essa expressão é avaliada para evitar injeção de código no motor. Após o cálculo, o motor também retorna alguns pontos da curva ajustada, auxiliando o cliente da requisição na sua visualização e possível renderização.

3.2.2. API hospedada

A infraestrutura escalável e gerenciável da API é mantida por meio dos recursos fornecidos pela Amazon Web Services (AWS).

Para executar o nosso motor de cálculo, optamos pelo serviço AWS Lambda, uma solução de computação sem servidor que elimina a necessidade de configurar servidores para a execução do código. Além disso, o AWS Lambda oferece integração com diversos serviços da AWS [Amazon Web Services 2024b].

Para hospedagem e gerenciamento da API, utilizamos o AWS API Gateway. Esse serviço permite que desenvolvedores criem e gerenciem APIs integradas com os serviços da AWS, como é o caso em que nosso código é mantido no AWS Lambda [Amazon Web Services 2024a].

Essa escolha arquitetônica proporciona a manutenção de uma API robusta e confiável, além de permitir que o nosso motor de cálculo possa ser utilizado por outras aplicações não relacionadas ao Regresso, que podem se beneficiar dos cálculos de ajuste de curva disponibilizados.

No entanto, é importante considerar que, apesar do plano gratuito do AWS API Gateway ser capaz de atender a uma demanda razoável de requisições, é impossível prever todos os cenários de carga. Por isso, se a carga for muito alta, pode ser necessário alocar recursos financeiros para sua manutenção ou, até mesmo, considerar a suspensão do serviço. Propusemos e implementamos uma solução para esse problema, que será apresentada na próxima seção.

3.2.3. API Local

Para garantir a disponibilidade da ferramenta mesmo em caso de indisponibilidade da API hospedada, desenvolvemos uma forma de manter essa API na máquina do usuário.

A solução envolve encapsular nosso "motor de cálculo", originalmente executado no AWS Lambda, e disponibilizá-lo através da biblioteca FastAPI em Python. Optamos por essa ferramenta por sua fácil instalação e implementação. Levando em conta essa possibilidade de local de processamento, na página da aplicação, é possível alternar entre as duas APIs através de uma seleção via *checkbox* no topo da página.

O código, as instruções de instalação e a documentação da API podem estão publicados em https://github.com/Sathler/Regresso-API.

3.3. Página Regresso

Nas próximas seções, descreveremos as características e a implementação do módulo *front-end* da ferramenta. Ela tem sua base em Angular, um *framework* para o desenvolvimento de aplicações web que utiliza *Typescript*, uma linguagem derivada do *Javascript*. A escolha pelo Angular foi motivada por sua alta modularidade, o que facilita a manutenção e o desenvolvimento progressivo da aplicação. O código deste módulo é aberto e está disponível em https://github.com/Sathler/Regresso-Page.

3.3.1. Página Principal

A página principal foi dividida em quatro seções distintas: o cabeçalho, que inclui uma lista de links; um formulário para inserção de dados; uma tabela inicialmente oculta que apresenta os resultados dos parâmetros de avaliação de todos os ajustes; e os cards que exibem os resultados e o desenho do ajuste.

O cabeçalho não apenas exibe um link para as outras páginas da aplicação, mas também inclui um *checkbox* para alternar entre a versão hospedada e local da API. Essa funcionalidade permite a troca sem a necessidade de recarregar a página, evitando a perda dos dados inseridos no formulário.

O formulário foi projetado para lidar com um grande volume de registros, mantendo a entrada de dados visível para o usuário. Além disso, inclui uma opção numérica para selecionar a precisão decimal desejada.

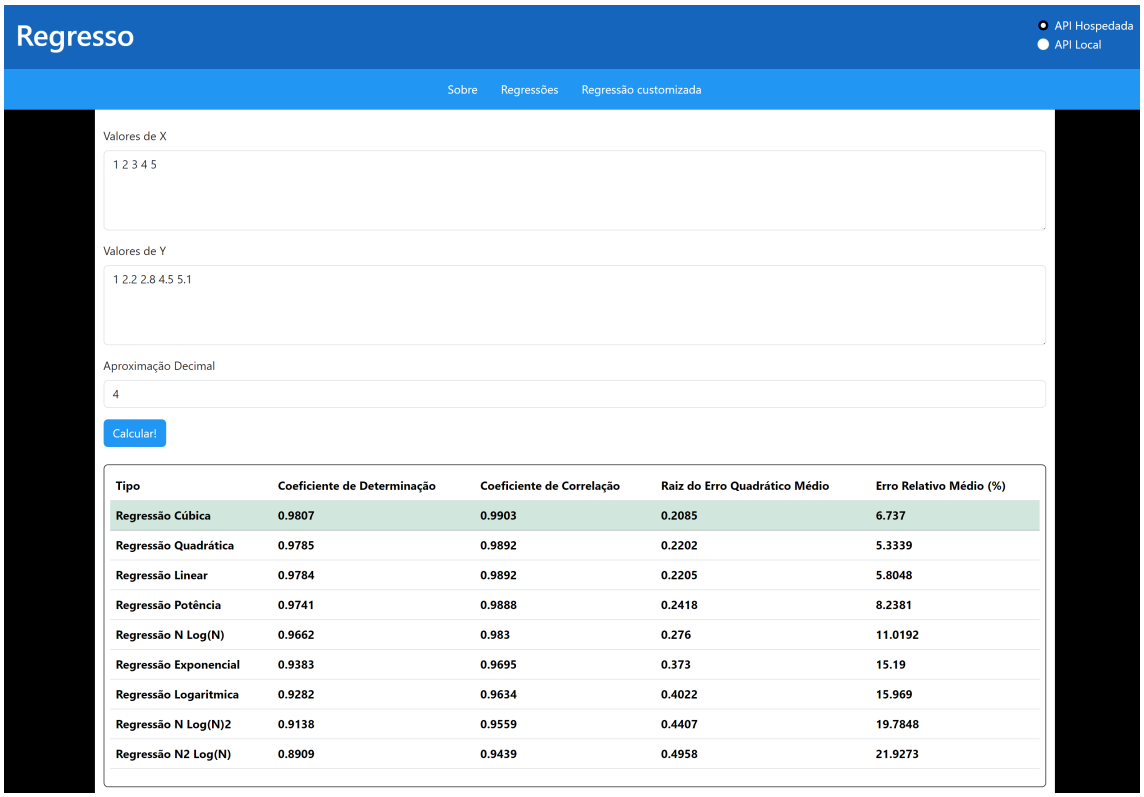

Figura 2. Resultado de uma ajuste de curvas calculado pela Aplicação Regresso.

A tabela de comparação permite uma visualização consolidada e resumida dos resultados dos ajustes de curvas através dos parâmetros de avaliação, destacando aqueles que obtiveram os melhores resultados.

Por fim, os *cards* apresentam os resultados do ajuste de forma individual, exibindo além dos parâmetros presentes na tabela de comparação, a equação e o gráfico da curva ajustada.

Em termos de arquitetura, cada *card* é um componente Angular que processa os dados de maneira independente. Após a conclusão dos cálculos, são compartilhados os resultados com o componente responsável pela tabela de comparação. Por fim, esse componente compila e exibe os dados consolidados.

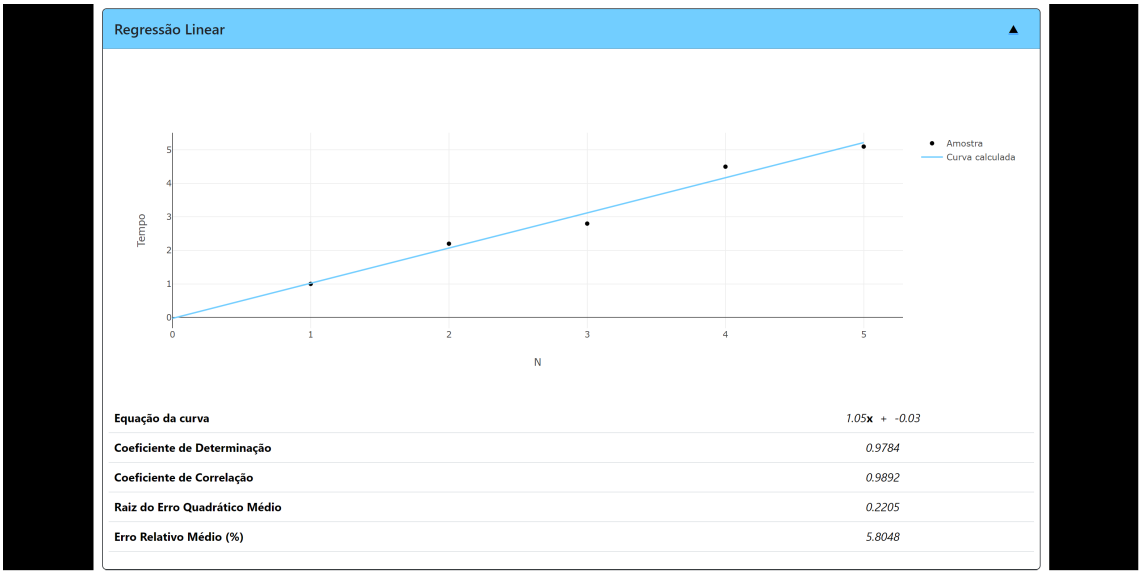

Figura 3. Curva calculada pela aplicação Regresso.

4. Testes da Ferramenta

Foram feitos variados testes da ferramenta para algoritmos básicos estudados nas disciplinas iniciais de Ciência da Computação. Alguns desses testes são detalhados a seguir e um resumo dos demais é mostrado na Tabela 8.

4.1. Número médio de comparações na Pesquisa Binária

Neste teste, considera-se o número médio de comparações feitas na Pesquisa Binária, para vetores de 9 tamanhos. A curva esperada do número médio de comparações, em função do tamanho do vetor é logarítmica, pois esse problema tem complexidade de caso médio $O(\log n)$. A Tabela 2 exibe os tamanhos dos vetores e os respectivos números médios de comparações medidos. Já a Tabela 3 e a Figura 4 são os resultados obtidos pela ferramenta.

n	10^2	5.10^2	10^3	10^4	5.10^4	10^5	10^6	5.10^6	10^7
NC_{med}	10,60	14,99	16,97	23,73	28,38	30,38	36,90	41,64	43,64

Tabela 2. Dados do Teste 1.

Tipo Regressão	Coef. Determ.	Coef. Corr.	RMSE	Erro rel.	Função
Logarítmica	1,00	1,00	0,08	0,37%	$2\log_2(n) - 2{,}81$
Potência	0,97	0,99	1,78	8,08 %	$8{,}65n^{0{,}1}$

Tabela 3. Melhores ajustes para o Teste 1.

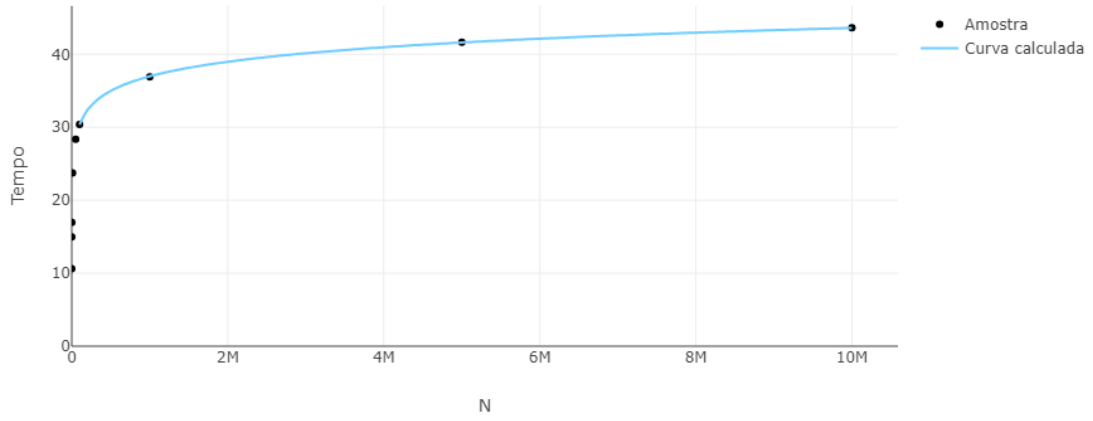

Figura 4. Melhor curva ajustada para o Teste 1.

Pode-se verificar que esse teste foi muito bem sucedido, porque a primeira opção de curva ajustada, dada pela Tabela 3, é justamente a função $f(n) = 2\log n - 2.8 = O(\log n)$, conforme esperado. A curva da Figura 4 ilustra o perfeito ajuste.

4.2. Fórmula para o termo da série de Fibonacci

Neste teste, são calculados os 20 primeiros termos da série de Fibonacci, a partir da recorrência usual para a mesma:

$$F(0) = 0; F(1) = 1; F(n) = F(n-1) + F(n-2), n \geq 2$$

Queremos comparar esses valores com aqueles de uma fórmula fechada para os termos da série [Cormen et al. 2022], dada por

$$Fib(n) = \frac{1}{\sqrt{5}}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right)$$

Observe que essa fórmula é bem próxima de uma função exponencial, assintoticamente, pois a segunda parcela é a exponencial de um número menor que 1, que tende a zero.

A Tabela 4 exibe os valores calculados para os 20 primeiros termos da série, e a Tabela 5 e a Figura 5 são os resultados obtidos pela ferramenta.

n	1	2	3	4	5	6	7	8	9	10
$Fib(n)$	1	1	2	3	5	8	13	21	34	55
n	11	12	13	14	15	16	17	18	19	20
$Fib(n)$	89	144	233	377	610	987	1597	2584	4181	6765

Tabela 4. Dados do Teste 3.

T. Regressão	Coef. Det.	Coef. Corr.	RMSE	Erro rel.	Função
Exponencial	1.00	1.00	0.08	2.67%	$0.45 * 1.62^n$
Potência	1.00	1.00	49.52	56.34 %	$2.41 * 10^{-8} * n^{8.8}$

Tabela 5. Melhores ajustes para o Teste 3

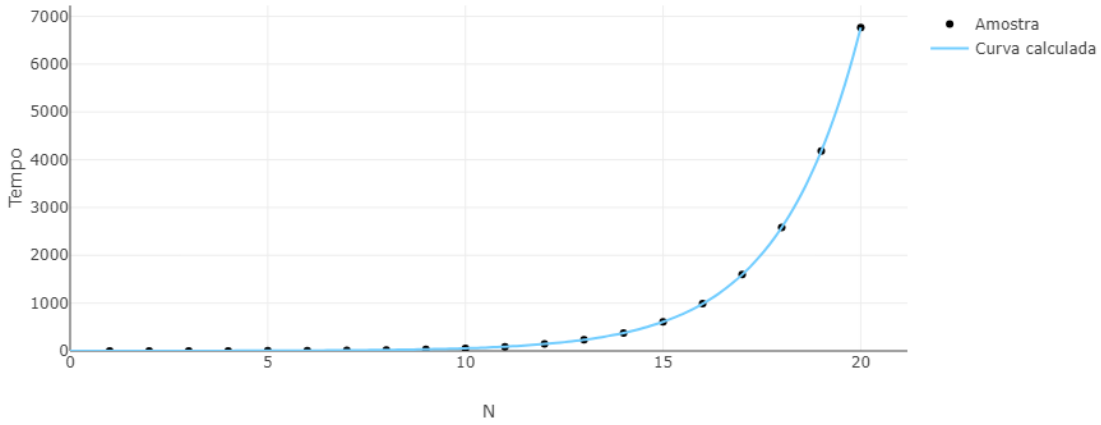

Figura 5. Melhor curva ajustada para o Teste 3.

Pode-se verificar que esse teste foi novamente muito bem sucedido, porque a primeira opção de curva ajustada, dada pela Tabela 5 corresponde exatamente à função $f(n) = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n$, que é a primeira parcela da fórmula exata. Note que a segunda parcela corresponde a $|0.47 \times 0.62^n|$ que é um valor que tende a zero com o crescimento de n. Por exemplo, para $n = 10$, essa parcela vale 0.004. Ou seja, o resultado é uma excelente aproximação para o que se esperava. A curva da Figura 5 ilustra o perfeito ajuste.

4.3. Ordenação pelo Shellsort

Neste teste, queremos obter a complexidade da ordenação de um vetor, pelo Shellsort, versão Pratt, a partir da contagem de comparações de chaves feitas no processo, com vetores de 6 tamanhos e utilizando como entrada um vetor inversamente ordenado.

O Algoritmo Shellsort executa um certo número de passos de Inserção, onde, em cada passo, há uma constante k, tal que consideram-se k subvetores com elementos espaçados de k índices entre si. No último passo executa-se a Inserção normal ($k = 1$). Na versão Pratt, k sempre é um inteiro cujos fatores são apenas 2 e 3. Desconhecemos como determinar qual configuração corresponde ao pior caso. Entretanto, o caso usado certamente é dos piores. A complexidade teórica do Shellsort na versão Pratt é $O(n(\log n)^2)$.

Porém, a complexidade depende da sequência dos k's utilizados e, para a maioria dessas sequências, essa complexidade é desconhecida.

A Tabela 6 exibe os tamanhos do vetores e os respectivos números médios de comparações medidos. Já a Tabela 7 e a Figura 6 são os resultados obtidos pela ferramenta.

n	10^2	5.10^2	10^3	10^4	5.10^4	10^5
$comp$	2187	17654	42381	697434	4648007	10400386

Tabela 6. Dados do Teste 3.

T. Regressão	Coef. Determ.	Coef. Corr.	RMSE	Erro rel.	Função
Potência	1.00	1.00	7472	15.4%	$15.61 * x^{1.16}$
$nlog_2(n)^2$	1.00	1.00	20709	127.1%	$0.38n * log_2(n)^2 + 15046.5$

Tabela 7. Melhores ajustes para o Teste 3.

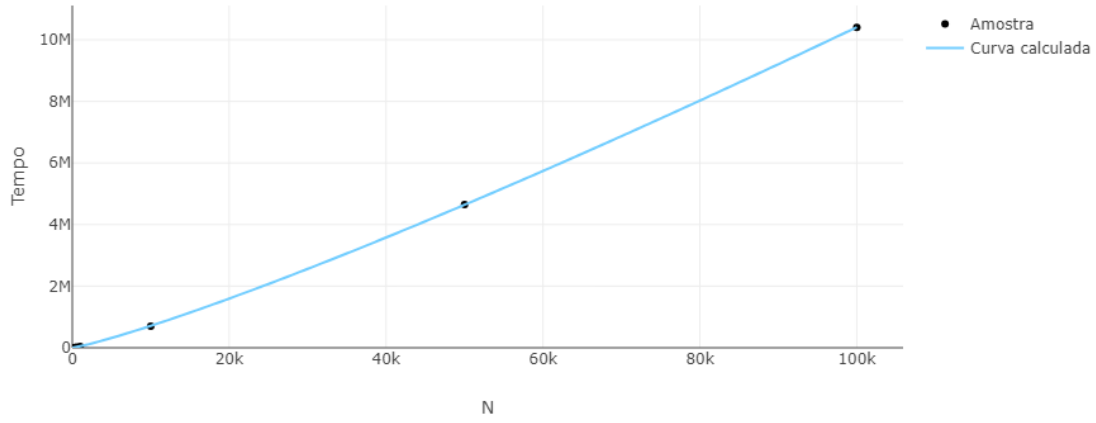

Figura 6. Melhor curva ajustada para o Teste 3.

Os resultados deste teste foram inconclusivos pois, embora os dois primeiros coeficientes da avaliação do ajuste sejam iguais a 1, os erros dados pelos dois últimos parâmetros de avaliação são significativos. Isso reflete a dificuldade de encontrar o pior caso do algoritmo.

Utilizamos um vetor inicialmente inversamente ordenado. Entretanto, desconhecemos como construir o pior caso do algoritmo. O que há de interessante nos resultados é que as duas primeiras funções (respectivamente $O(n^a)$ e $O(n(\log n)^2)$ são usualmente cogitadas como candidatas a expressarem a complexidade do Shellsort para diversas versões, correspondentes a diferentes sequências de k's. A curva da Figura 6 ilustra o ajuste para a primeira função.

4.4. Demais testes

Os demais testes realizados são resumidos na Tabela 8 onde, na primeira coluna, é descrito o teste, na segunda, a função esperada para a complexidade do algoritmo usado e, na terceira, a melhor função de ajuste obtida.

Teste	Complexidade esperada	Ajuste obtido
Produto de Matrizes	$O(n^3)$	n^3
Criação de um Heap com DesceHeap	$O(n)$	$2n - 18.59$
Criação de um Heap com SobeHeap	$O(n \log n)$	$0.96n * log_2(n) - 40762$
Ordenação por Seleção	$O(n^2)$	$0.5n^2 - 0.5n$
Ordenação pelo Mergesort	$O(n \log n)$	$n * log_2(n) + 34.5$

Tabela 8. Resumo dos demais testes realizados

Concluímos que, para todos os testes cujo algoritmo correspondente tenha complexidade conhecida, os resultados obtidos com a ferramenta Regresso foram plenamente satisfatórios pois, em todos eles, o ajuste obtido pela ferramenta correspondeu ao esperado. No caso do algoritmo de Ordenação pelo Shellsort, cuja complexidade é desconhecida, a ferramenta sugeriu funções usualmente candidatas para expressar a complexidade real.

5. Comparações do Regresso com outras ferramentas

Nesta seção, iremos comparar a ferramenta desenvolvida com outras apresentadas anteriormente, destacando as vantagens do seu uso em relação às demais e sugerindo possíveis melhorias a serem implementadas. São feitas comparações com três ferramentas cujo processo é bem semelhante ao do Regresso: *PlanetCalc*, *WolframAlpha* e *Desmos*. A ferramenta *EMA* não foi incluída na comparação porque o seu processo de uso é fundamentalmente diferente dessas outras quatro.

Vale destacar que todas as ferramentas avaliadas demonstraram eficiência em termos de tempo de resposta satisfatório. Apesar disso, o Regresso se destaca por oferecer a opção de execução da API local, o que permite aos usuários alocarem mais recursos para o cálculo dos ajustes, se necessário, contribuindo para uma maior flexibilidade e desempenho otimizado durante o processamento dos cálculos[1].

Em relação ao propósito de apoiar o ensino de análise de algoritmos, um aspecto importante é a capacidade de comparar vários ajustes de curvas para análises diversas, utilizando funções comuns na análise algorítmica. Na Tabela 9, examinamos algumas características de cada ferramenta apresentada. Usamos as classificações atente totalmente (✔), atende parcialmente (—), ou não atende (✖) e, em seguida, descrevemos cada um dos tópicos e as avaliações dadas.

	Interface Intuitiva	Usabilidade	Comparativo entre Ajustes	Funções de Interesse	Funções Customizadas
Regresso	✔	✔	✔	✔	✔
PlanetCalc	✔	✔	✔	✖	✖
WolframAlpha	—	—	✖	✔	✔
Desmos	—	—	✔	✔	✔

Tabela 9. Funcionalidades disponíveis em cada ferramenta.

[1]Na API hospedada, os algoritmos de regressão têm um limite máximo de iterações, o que pode impactar o desempenho em determinadas situações.

5.1. Interface Intuitiva

A ferramenta Regresso buscou oferecer uma interface simples e intuitiva, muito semelhante à do *PlanetCalc*. Por outro lado, o *WolframAlpha* e o *Desmos*, apesar de terem interfaces simples, podem não ser facilmente compreendidos, pois exigem que o usuário tenha conhecimento prévio sobre o formato e as expressões válidas para utilizá-los.

5.2. Usabilidade

Neste tópico, comparamos a complexidade e o volume de entrada para utilizar as ferramentas, tanto para curvas padrão quanto para uma função customizada. Como exemplo, vamos utilizar os pares ordenados $\{(1,2),(2,3),(3,9),(4,8),(5,10)\}$ para os quais desejamos sugestões de funções.

Avaliamos que as ferramentas Regresso e *PlanetCalc* atendem totalmente a esse quesito, pois a entrada é intuitva e pouco complexa. Colocamos separadamente as entradas separadas por espaços em branco para variáveis independentes("1 2 3 4 5") e variáveis dependentes("2 3 9 8 10"). O *PlanetCalc* não fornece opção de usar uma função customizada e, para Regresso, é necessário especificá-la como uma cadeia como, por exemplo "$a * x + b$" para uma função linear.

As ferramentas *Desmos* e *WolframAlpha* atendem parcialmente à usabilidade desejada. O Desmos permite a entrada de dados por meio de uma tabela. Entretanto, por não possuir detecção automática de curvas, eventualmente, torna-se necessário inserir manualmente várias funções para realizar uma análise. Por outro lado, *WolframAlpha* requer uma entrada mais complexa em relação às outras ferramentas, como no exemplo: "findfit[$\{\{1,2\},\{2,3\},\{3,9\},\{4,8\},\{5,10\}\}$, $\{a*x+b\}$, $\{a,b\}, x$]".

5.3. Comparativo entre Ajustes

As ferramentas Regresso e *PlanetCalc* apresentam resultados consolidados de diversos tipos de ajustes, junto com seus parâmetros de avaliação, o que permite uma rápida comparação visual dos resultados e uma análise mais aprofundada de maneira relativamente simples.

A ferramenta *Desmos* possibilita a inserção de uma quantidade indefinida de funções de ajuste, mas peca na obtenção dos parâmetros de avaliação, delegando ao usuário a responsabilidade de definir como cada parâmetro deve ser calculado.

Por fim, o *WolframAlpha* não permite a comparação direta entre diferentes tipos de ajustes, o que o torna inadequado para situações que exigem essa funcionalidade.

5.4. Funções de Interesse

Nesta seção, verificamos se cada ferramenta permite aos usuários aproximar as funções calculadas para aquelas comumente usadas na análise algorítmica.

A ferramenta emphRegresso foi desenvolvida com a premissa de auxiliar em disciplinas de análise algorítmica, atendendo nativamente às funções mais comuns nesse campo de estudo, além de permitir que o usuário insira funções mais incomuns, conforme necessário.

Já a ferramenta *PlanetCalc*, por outro lado, oferece funções mais genéricas, que não são tão aplicáveis a esse tipo de estudo, e, portanto, não atende adequadamente a essa demanda.

Por fim, tanto as ferramentas *Desmos* quanto *WolframAlpha* permitem a aproximação de qualquer curva definida pelo usuário, cumprindo assim este requisito.

5.5. Funções Customizadas

Avaliamos, por fim, se cada ferramenta possibilita que o usuário aproxime as curvas a qualquer função que o usuário desejar.

Conforme mencionado anteriormente, as ferramentas *Desmos*, *WolframAlpha* e Regresso permitem que o usuário defina a função de ajuste. Essa funcionalidade é de grande interesse para nossa área de estudo, devido à vasta variedade de curvas de complexidade existentes na análise algorítmica.

Em contrapartida, a ferramenta *PlanetCalc* não oferece esse tipo de ajuste personalizado, o que representa uma grande desvantagem em relação às outras ferramentas avaliadas.

6. Conclusões

Neste trabalho, desenvolvemos uma ferramenta educacional destinada a auxiliar o ensino de disciplinas como Algoritmos e Estruturas de Dados. Para isso, oferecemos funcionalidades intuitivas e simples para a inferência de funções de complexidade de algoritmos, complementadas por seus gráficos correspondentes.

Realizamos e apresentamos testes que abrangeram vários dos problemas abordados em disciplinas de Algoritmos e Estruturas de Dados. Na grande maioria dos casos, a ferramenta apresentou resultados satisfatórios para a análise dos algoritmos envolvidos.

Um trabalho futuro que sugerimos é realizar uma validação junto aos alunos, que constituem o principal público-alvo da ferramenta. Isso permitiria coletar feedback direto dos usuários e identificar áreas de melhoria.

Ressaltamos que o Regresso foi desenvolvido no contexto de código aberto, tornando sua evolução disponível para quem desejar realizar melhorias e correções. Sugerimos dois pontos que podem ser reavaliados. O primeiro é uma avaliação das possíveis consequências das restrições de projeto impostas aos coeficientes de ajuste. Não sabemos se isso introduz alguma limitação nos resultados, sendo necessários mais testes para verificar.

O segundo ponto refere-se à interpretação das regressões utilizadas. Os quatro principais parâmetros utilizados nos critérios de escolha das melhores funções ainda não são muito intuitivos para quem não é familiarizado com os termos técnicos. Uma ideia é que a ferramenta seja capaz de fornecer uma análise mais qualitativa desses dados. Em muitos casos, obtiveram-se coeficientes de determinação e correlação muito altos, assim como parâmetros relacionados a erros, havendo discrepâncias que carecem de melhores explicações. Além disso, é necessário revisar a relação entre os coeficientes de determinação, correlação e de erro.

Apesar de muito a ser aprimorado, reforçamos que os testes que realizamos foram bastante positivos e encorajadores. Esperamos deixar uma contribuição modesta, mas relevante para o curso de Ciência da Computação da UERJ.

Referências

Amazon Web Services (2024a). AWS API Gateway. https://aws.amazon.com/pt/api-gateway/. Acesso em 03 de março de 2024.

Amazon Web Services (2024b). AWS Lambda. https://aws.amazon.com/pt/lambda/. Acesso em 03 de março de 2024.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2022). *Introduction to algorithms*. MIT press, Cambridge, EUA, 4 edition.

Dasgupta, S., Papadimitriou, C. H., and Vazirani, U. (2006). *Algorithms*. McGraw-Hill, Inc., USA, 1 edition.

de Souza Oliveira, F. (2024). Ema. https://fabianooliveira.ime.uerj.br/EMA. Acesso em 22 de outubro de 2024.

DeGroot, M. H. and Schervish, M. J. (2012). *Probability and Statistics*. Pearson, Boston, 4 edition.

Dekking, F., Kraaikamp, C., Lopuhaä, H., and Meester, L. (2005). *A Modern Introduction to Probability and Statistics: Understanding Why and How*. Springer.

Desmos (2024). Desmos Calculator. https://www.desmos.com/calculator. Acesso em 03 de março de 2024.

Draper, N. R. and Smith, H. (1998). *Applied Regression Analysis*. John Wiley & Sons, New York, 3rd edition.

Juliana Macario de Souza (2018). Sobre ferramentas para análise automatizada de algoritmos. https://www.bdtd.uerj.br:8443/handle/1/7661. Acesso em 08 de Outubro de 2024.

Kiusalaas, J. (2013). *Numerical Methods in Engineering with Python 3*. Cambridge University Press, 3rd edition.

Kolb, R. E. (1984). *Curve Fitting for Programmable Calculators*. Sybex.

PlanetCalc (2024). PlanetCalc. https://planetcalc.com/. Acesso em 02 de março de 2024.

Szwarcfiter, J. L. (2018). *Teoria Computacional dos Grafos: Os Algoritmos*. Elsevier, Amsterdam, 1 edition.

Wofram Research (2024). Wolfram Alpha. https://www.wolframalpha.com/. Acesso em 02 de março de 2024.