BACHAREL EM CIÊNCIA DA COMPUTAÇÃO

SUPERANDO O DESAFIO PEDAGÓGICO NO ENSINO DE PO: DESENVOLVIMENTO DE UM LABORATÓRIO DIGITAL PARA O MÉTODO SIMPLEX E *BRANCH-AND-BOUND*

MARCOS WINICYUS BORGES LIMA

Rio Verde, GO

2026

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA GOIANO - CAMPUS RIO VERDE
BACHAREL EM CIÊNCIA DA COMPUTAÇÃO

SUPERANDO O DESAFIO PEDAGÓGICO NO ENSINO DE PO: DESENVOLVIMENTO DE UM LABORATÓRIO DIGITAL PARA O MÉTODO SIMPLEX E *BRANCH-AND-BOUND*

MARCOS WINICYUS BORGES LIMA

Trabalho de Conclusão de Curso apresentado ao Instituto Federal de Educação, Ciência e Tecnologia Goiano - Campus Rio Verde, como requisito parcial para a obtenção do Grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Márcio Antônio Ferreira Belo Filho

Rio Verde, GO

Junho, 2026

Ficha de identificação da obra elaborada pelo autor, através do Programa de Geração Automática do Sistema Integrado de Bibliotecas do IF Goiano - SIBi

Borges Lima, Marcos Winicyus
L732s Superando o Desafio Pedagógico no Ensino de PO: Desenvolvimento de um Laboratório Digital para o Método Simplex e Branch-and-Bound / Marcos Winicyus Borges Lima. Rio Verde 2026.

37f. il.

Orientador: Prof. Dr. Márcio Antônio Ferreira Belo Filho.
Tcc (Bacharel) - Instituto Federal Goiano, curso de 0219201 - Bacharelado em Ciência da Computação - Integral - Rio Verde (Campus Rio Verde).
1. Pesquisa Operacional. I. Título.

SERVIÇO PÚBLICO FEDERAL
MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA GOIANO

TERMO DE AUTORIZAÇÃO PARA DISPONIBILIZAR PRODUÇÃO TÉCNICA NO REPOSITÓRIO INSTITUCIONAL DO IF GOIANO

Repositório Institucional do IF Goiano - RIIF Goiano Sistema Integrado de Bibliotecas

- Profissional de Educação do IF Goiano -

Com base no disposto na Lei Federal nº 9.610/98, e manual sobre a Produção Técnica, publicado pela DAV/CAPES/MEC*, AUTORIZO o Instituto Federal de Educação, Ciência e Tecnologia Goiano, a disponibilizar gratuitamente o documento no Repositório Institucional do IF Goiano (RIIF Goiano), sem ressarcimento de direitos autorais, conforme permissão assinada eletronicamente abaixo, em formato digital para fins de leitura, download e impressão, a título de divulgação da produção técnico-científica no IF Goiano.

Identificação da Produção Técnica – DAV/CAPES

[ ] Editoria [ ] Material Didático

[ ] Curso de Formação Profissional [ ] Projetos de Extensão à Comunidade

[ ] Relatório Técnico Conclusivo [ ] Atividade Técnica/Tecnológica

[ ] Disseminação do Conhecimento Técnico/Tecnológico [ ] Produto Bibliográfico

[X] Outras Produções Técnicas - Tipo: TCC (Graduação)

Nome Completo do Autor/a: Marcos Winicyus Borges Lima

Matrícula: 2017102201910331

Título do Trabalho: SUPERANDO O DESAFIO PEDAGÓGICO NO ENSINO DE PO: DESENVOLVIMENTO DE UM LABORATÓRIO DIGITAL PARA O MÉTODO SIMPLEX E BRANCH-AND-BOUND

Restrições de Acesso ao Documento

Documento confidencial: [X] Não [ ] Sim

Justifique: _____

Informe a data que poderá ser disponibilizado no RIIF Goiano: 10 / 08 / 2026

O documento está sujeito a registro de patente? [ ] Sim [X] Não

O documento pode vir a ser publicado como livro e/ou artigo? [ ] Sim [X] Não

DECLARAÇÃO DE DISTRIBUIÇÃO NÃO-EXCLUSIVA

O/A referido/a docente e/ou autor/a declara que:

1 - o documento é seu trabalho original, detém os direitos autorais da produção técnica e não infringe os direitos de qualquer outra pessoa ou entidade;

2 - obteve autorização de quaisquer materiais inclusos no documento do qual não detém os direitos de autor/a, para conceder ao Instituto Federal de Educação, Ciência e Tecnologia Goiano os direitos requeridos e que este material cujos direitos autorais são de terceiros, estão claramente identificados e reconhecidos no texto ou conteúdo do documento entregue;

3 - cumpriu quaisquer obrigações exigidas por contrato ou acordo, caso o documento entregue seja baseado em trabalho financiado ou apoiado por outra instituição que não o Instituto Federal de Educação, Ciência e Tecnologia Goiano.

Rio Verde, 6 de agosto de 2026.

(Assinado Eletronicamente)

Marcos Winicyus Borges Lima (Autor)

(Assinado Eletronicamente)

Marcio Antonio Ferreira Belo Filho (Orientador)

1223649

(Assinatura do Docente, Autor e/ou Detentor dos Direitos Autorais)

Documento assinado eletronicamente por:

- **Marcio Antonio Ferreira Belo Filho**, **PROFESSOR ENS BASICO TECN TECNOLOGICO** , em 06/08/2026 09:34:19.
- **Marcos Winicyus Borges Lima**, **2017102201910331 - Discente**, em 06/08/2026 14:38:44.

Este documento foi emitido pelo SUAP em 06/08/2026. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifgoiano.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 845308
Código de Autenticação: 060185e3a4

INSTITUTO FEDERAL GOIANO
Campus Rio Verde
Rodovia Sul Goiana, Km 01, Zona Rural, 01, Zona Rural, RIO VERDE / GO, CEP 75901-970
(64) 3624-1000

Regulamento de Trabalho de Curso (TC) – IF Goiano - Campus Rio Verde

ANEXO V - ATA DE DEFESA DE TRABALHO DE CURSO

Aos **17** dias do mês de **junho** de **dois mil e vinte e seis**, às 14 horas, reuniu-se a Banca Examinadora composta por: Prof. **Márcio Antônio Ferreira Belo Filho** (orientador), Prof. **André da Cunha Ribeiro** e Prof. **Marlus Dias Silva**, para examinar o Trabalho de Curso (TC) intitulado **SUPERANDO O DESAFIO PEDAGÓGICO NO ENSINO DE PO: DESENVOLVIMENTO DE UM LABORATÓRIO DIGITAL PARA O MÉTODO SIMPLEX E BRANCH-AND-BOUND**, de **Marcos Winicyus Borges Lima**, estudante do curso de **Bacharelado em Ciência da Computação** do IF Goiano – Campus Rio Verde, sob Matrícula nº **2017102201910331**. A palavra foi concedida ao estudante para a apresentação oral do TC, em seguida houve arguição do candidato pelos membros da Banca Examinadora. Após tal etapa, a Banca Examinadora decidiu pela **APROVAÇÃO** do estudante. Ao final da sessão pública de defesa foi lavrada a presente ata, que, foi assinada pelos membros da Banca Examinadora e Mediador de TC.

Rio Verde, 17 de junho de 2026.

Márcio Antônio Ferreira Belo Filho

Orientador

André da Cunha Ribeiro

Membro da Banca Examinadora

Marlus Dias Silva

Membro da Banca Examinadora

Douglas Cedrim Oliveira

Mediador de TC

Documento assinado eletronicamente por:

- **Marcio Antonio Ferreira Belo Filho**, **PROFESSOR ENS BASICO TECN TECNOLOGICO** , em 17/06/2026 14:56:51.
- **Marlus Dias Silva**, **PROFESSOR ENS BASICO TECN TECNOLOGICO** , em 17/06/2026 14:59:00.
- **Andre da Cunha Ribeiro**, **PROFESSOR ENS BASICO TECN TECNOLOGICO** , em 17/06/2026 14:59:59.
- **Douglas Cedrim Oliveira**, **PROFESSOR ENS BASICO TECN TECNOLOGICO** , em 17/06/2026 21:05:35.

Este documento foi emitido pelo SUAP em 17/06/2026. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifgoiano.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 832114
Código de Autenticação: cce649f7cf

Dedico esse trabalho à minha família e professores, que me deram todo o suporte e apoio para concluir essa longa jornada.

RESUMO

A Pesquisa Operacional (PO) é uma disciplina fundamental para a área de ciência da computação, mas seu ensino enfrenta barreiras pedagógicas significativas. Algoritmos centrais, como o Método Simplex e o *Branch-and-Bound* (B&B), são abstratos e os alunos frequentemente falham em conectar a execução mecânica dos algoritmos (álgebra) com a intuição conceitual (geometria) que os fundamenta. Ferramentas educacionais existentes, como o SIMO, embora relevantes, apresentam limitações críticas: operam em arquiteturas *client-side*, impõem uma barreira de entrada ao exigir a inserção manual de problemas na forma matricial e promovem um engajamento passivo (Nível 2, *Viewing*). Este trabalho propõe o desenvolvimento de uma ferramenta educacional web, interativa e de código aberto, para auxiliar no ensino e aprendizagem do Método Simplex e do *Branch-and-Bound*. A solução ataca as lacunas identificadas através de três inovações centrais: (1) Inovação Tecnológica, com uma arquitetura *server-side* moderna usando Python e Streamlit, que desacopla a lógica computacional da interface; (2) Inovação Pedagógica, com a implementação de uma Biblioteca de Problemas Clássicos Editáveis que fomenta a Aprendizagem Baseada na Investigação (IBL); e (3) Integração Visual-Analítica, que sincroniza a execução algébrica do *tableau* com a visualização geométrica da região factível em 2D e 3D. Adicionalmente, o sistema incorpora ferramentas de Análise de Pós-Otimização, gerando automaticamente o Problema Dual e relatórios de Análise de Sensibilidade. Esta abordagem fornece os recursos necessários para fomentar um engajamento de Nível 4 (*Changing*) na taxonomia de Naps, superando a barreira de entrada da modelagem e permitindo uma compreensão holística dos problemas de otimização.

LIMA, Marcos Winicyus Borges.**Superando o Desafio Pedagógico no Ensino de PO: Desenvolvimento de um Laboratório Digital para o Método Simplex e *Branch-and-Bound*.** Junho, 2026. 38 f. Monografia (Curso de Bacharel em Ciência da Computação), Instituto Federal de Educação, Ciência e Tecnologia Goiano - Campus Rio Verde. Rio Verde, GO, Junho, 2026.

Palavras-chave: Pesquisa Operacional (PO), Método Simplex, *Branch-and-Bound* (B&B), Aprendizagem Baseada na Investigação (IBL), Visualização de Algoritmos

ABSTRACT

Operational Research (OR) is a fundamental discipline for the computer science field, but its teaching faces significant pedagogical barriers. Central algorithms, such as the Simplex Method and *Branch-and-Bound* (B&B), are abstract, and students often fail to connect the mechanical execution of the algorithms (algebra) with the conceptual intuition (geometry) that underpins them. Existing educational tools, like SIMO, although relevant, present critical limitations: they operate on *client-side* architectures, impose a barrier to entry by requiring the manual insertion of problems in matrix form, and promote passive engagement (Level 2, *Viewing*). This work proposes the development of an interactive, open-source web-based educational tool to assist in the teaching and learning of the Simplex Method and *Branch-and-Bound*. The solution addresses the identified gaps through three central innovations: (1) Technological Innovation, with a modern *server-side* architecture using Python and Streamlit, which decouples the computational logic from the interface; (2) Pedagogical Innovation, through the implementation of a Library of Editable Classic Problems that fosters Inquiry-Based Learning (IBL); and (3) Visual-Analytical Integration, which synchronizes the algebraic execution of the *tableau* with the geometric visualization of the feasible region in 2D and 3D. Additionally, the system incorporates Post-Optimization Analysis tools, automatically generating the Dual Problem and Sensitivity Analysis reports. This approach provides the necessary resources to foster student engagement at Level 4 (*Changing*) of Naps' taxonomy, overcoming the modeling barrier to entry and enabling a holistic understanding of optimization problems.

LIMA, Marcos Winicyus Borges. **Overcoming the Pedagogical Challenge in OP Teaching: Development of a Digital Laboratory for Simplex and *Branch-and-Bound* Methods**. Junho, 2026. 38 f. Trabalho de Conclusão de Curso Bacharel em Ciência da Computação, Instituto Federal de Educação, Ciência e Tecnologia Goiano - Campus Rio Verde. Rio Verde, GO, Junho, 2026.

Keywords: Operations Research (OP), Simplex Method, *Branch-and-Bound* (B&B), Inquiry-Based Learning (IBL), Algorithm Visualization (AV)

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AV	*Algorithm Visualization* (Visualização de Algoritmos)
BFS	*Breadth-First Search* (Busca em Largura)
BLAS	*Basic Linear Algebra Subprograms*
B&B	*Branch-and-Bound*
C4	*Context, Containers, Components, and Code*
CSS	*Cascading Style Sheets*
DFS	*Depth-First Search* (Busca em Profundidade)
DOM	*Document Object Model*
GILP	*Geometric Interpretation of Linear Programs*
GNU	*GNU's Not Unix!*
GPL	*General Public License*
HTML	*HyperText Markup Language*
IA	Inteligência Artificial
IBL	*Inquiry-Based Learning* (Aprendizagem Baseada na Investigação)
IDE	*Integrated Development Environment* (Ambiente de Desenvolvimento Integrado)
ILESA	*Intelligent Learning Environment for the Simplex Algorithm*
JS	*JavaScript*
JSON	*JavaScript Object Notation*
LAPACK	*Linear Algebra PACKage*
MVC	*Model-View-Controller*
PLI	Programação Linear Inteira
PL	Programação Linear
PO	Pesquisa Operacional

PPL	Problema de Programação Linear
RF	Requisitos Funcionais
RNF	Requisitos Não-Funcionais
SIGCSE	*Special Interest Group on Computer Science Education*
SIMO	Sistema Interativo para Métodos de Otimização
SVG	*Scalable Vector Graphics*
TORA	*TORA Optimization System*
UI	*User Interface* (Interface de Usuário)
URL	*Uniform Resource Locator*
UX	*User Experience* (Experiência do Usuário)

Lista de Algoritmos

*

SUMÁRIO

1 Introdução

De acordo com Taha (2017), a Pesquisa Operacional (PO) atua como uma ferramenta de tomada de decisão que é "tanto uma ciência quanto uma arte", aliando técnicas matemáticas rigorosas à criatividade da equipe de modelagem. Seu objetivo é fornecer subsídios racionais para otimizar processos, com aplicações na indústria, logística, finanças e serviços.

A Programação Linear (PL) pode ser vista como um modelo de alocação de recursos que busca maximizar a receita sob recursos limitados, sendo estritamente projetada para modelos com funções objetivo e de restrição lineares. Por sua vez, a Programação Linear Inteira (PLI) estende e aplica essa mesma estrutura de otimização a cenários onde a natureza da situação impede a atribuição de valores fracionários às variáveis do modelo, forçando as variáveis de decisão a assumirem valores discretos.

Na solução de problemas de PL, o Método Simplex foi criado em 1947. Este algoritmo que tornou os (grandes) problemas de PL solúveis na prática. De maneira análoga, para a Programação Linear Inteira (PLI), destaca que o primeiro algoritmo B&B foi desenvolvido em 1960 por A. Land e G. Doig para o problema geral de PLI puro e misto, consolidando-se como o método de resolução padrão.

Existe uma interdependência algorítmica entre esses métodos. O B&B utiliza o Simplex como sub-rotina em seu passo de limitação (*limitação*). Em cada nó da árvore de busca, o algoritmo resolve uma relaxação linear do problema inteiro, tornando o domínio do Simplex um pré-requisito para a compreensão do *Branch-and-Bound*.

Esse acoplamento técnico gera uma dependência pedagógica. Na prática, constata-se uma barreira recorrente durante o estudo de Pesquisa Operacional: a dificuldade em conectar a álgebra do Simplex com a sua geometria espacial dificulta o aprendizado subsequente do *Branch-and-Bound*. Essa lacuna justifica o desenvolvimento de uma ferramenta unificada que aborde os dois algoritmos de forma integrada.

O ensino dos algoritmos de PO enfrenta barreiras devido à sua abstração matemática. A literatura da área reconhece essa limitação pedagógica; o próprio Taha (2017), em sua obra fundamental, expressa a necessidade de adotar abordagens que tornem o estudo da disciplina menos árida. Essa dificuldade se acentua na execução algébrica matricial, onde os cálculos de Gauss-Jordan são extensos, repetitivos e propensos a erros operacionais, desviando o foco do aluno da compreensão intuitiva para a mera repetição mecânica.

Este trabalho foca na lacuna cognitiva entre a execução mecânica dos algoritmos e a intuição espacial que os fundamenta. Essa divergência manifesta-se de duas formas:

1. **No Método Simplex:** A literatura aponta a dificuldade de conectar a manipulação algébrica do quadro (*tableau*) com o seu significado geométrico: o deslocamento

entre vértices adjacentes da região viável. Esta lacuna é evidenciada por Robbins et al. (2023) que ressaltam que embora a motivação geométrica seja intuitiva, o algoritmo é mecanicamente complexo, e a forte manipulação algébrica pode ofuscar a compreensão de suas ideias centrais. O processo de pivoteamento, desprovido de interpretação espacial, torna-se uma aplicação de regras arbitrárias (LAZARIDIS; SAMARAS; ZISSOPOULOS, 2003). É por essa razão que a literatura instrui o estudante a ler o ponto de solução resultante diretamente no quadro do Simplex e, em seguida, localizar seu ponto extremo correspondente no espaço de solução gráfico, garantindo assim a compreensão da essência do método (TAHA, 2017).

2. **No Algoritmo *Branch-and-Bound*:** No caso do *Branch-and-Bound*, Zumaytis e Karnalim (2017) pontuam que a maioria dos estudantes sente-se sobrecarregada pelos numerosos passos necessários para resolver um problema. A falta de imaginação visual torna ainda mais difícil projetar mentalmente a árvore lógica de busca, exigindo do aluno o rastreio manual da árvore, das múltiplas relaxações lineares (onde cada nó é um problema Simplex) e das regras de poda por limite, inviabilidade ou integralidade.

Ferramentas de Visualização de Algoritmos (AV) são essenciais para reverter esse cenário. Conforme o meta-estudo conduzido por Hundhausen, Douglas e Stasko (2002), "os usos educacionais mais bem-sucedidos da tecnologia de AV são aqueles em que ela atua como um veículo para engajar ativamente os alunos no processo de aprendizagem".

O software desenvolvido neste trabalho opera como um laboratório digital fundamentado na Aprendizagem Baseada na Investigação (IBL). O IBL direciona o aluno da recepção passiva para a participação ativa, promovendo a formulação de hipóteses em um ambiente controlado (LAPPAS; KRITIKOS, 2018). Essa abordagem alinha-se às necessidades do ensino de otimização, uma vez que, como ressalta Taha (2017), "os cálculos de Gauss-Jordan são extensos, repetitivos e propensos a erros operacionais", sendo o verdadeiro objetivo pedagógico fazer com que o aluno entenda como o método Simplex funciona.

A construção da ferramenta surge da análise de sistemas preexistentes, com destaque para o Sistema Interativo para Métodos de Otimização (SIMO) (MALTA et al., 2016). O SIMO apresenta restrições que este projeto procura resolver:

1. **Limitação Tecnológica:** O SIMO adota uma arquitetura *client-side*, com lógica baseada em JavaScript (`glpk.js` e `jQuery`) (MALTA et al., 2016). Essa estrutura acopla o motor matemático à interface (DOM) e restringe o desempenho computacional aos recursos do navegador do usuário.
2. **Limitação Pedagógica e Usabilidade:** A aplicação exige a inserção manual do problema na forma matricial (matrizes A, b e c) (MALTA et al., 2016), assumindo que o aluno já superou as etapas de formulação. De fato, a solução do modelo é a fase operacionalmente menos complexa de todas as fases da PO porque envolve o uso de

algoritmos de otimização bem definidos, o que transfere o peso cognitivo inicial para a fase da modelagem. A interface restringe alterações dimensionais sem reiniciar o problema, carece de explicações textuais passo a passo e não possui integração de estado entre os módulos Simplex e *Branch-and-Bound*.

Para mitigar essas limitações, o projeto implementa as seguintes abordagens:

Abordagem Pedagógica: A **Biblioteca de Problemas Clássicos Editáveis** fornece modelos pré-formulados (ex: Problema da Dieta, Problema da Mochila). Essa estrutura insere o usuário no Nível 4 (*Changing*) da taxonomia de Naps et al. (2003), superando o Nível 2 (*Viewing*) do SIMO. O aluno altera coeficientes ou restrições e acompanha o impacto matemático em tempo real.

Abordagem Tecnológica: Adoção da **Arquitetura Python e Streamlit** (Streamlit Inc., 2026). O sistema processa os cálculos em *server-side*. A decisão de evitar frameworks web *client-side* e adotar uma infraestrutura em Python apoia-se na necessidade de um processamento robusto e na manipulação matricial otimizada via NumPy (NumPy Developers, 2026). As propriedades dessa arquitetura englobam:

- **Desacoplamento:** A lógica computacional opera independente da camada de renderização visual (QUANSIGHT, 2020; DATACAMP, 2024).
- **Implementação *white-box*:** O algoritmo Simplex foi reimplementado do zero, evitando *solvers* em *black-box* (como `scipy.optimize.linprog`). Isso permite a extração granular de dados em cada iteração matricial, viabilizando o detalhamento didático das operações.

1.1 Objetivo Geral

Desenvolver uma ferramenta web educacional *open-source* em Python, denominada Solver LP (mantendo a sigla LP por se tratar do nome próprio do software), para auxiliar o ensino do Método Simplex e *Branch-and-Bound*, aplicando a Aprendizagem Baseada na Investigação (IBL) para superar as restrições pedagógicas e de arquitetura de softwares existentes.

1.2 Objetivos Específicos

- Analisar ferramentas de visualização para PO, identificando limitações pedagógicas do sistema SIMO (MALTA et al., 2016).
- Desenvolver o motor adotando a mesma estrutura defendida por Deshpande et al. (2025), onde a linguagem Python é utilizada para operações de *backend* robustas e processamento de dados, enquanto o *framework* Streamlit cria uma interface intuitiva e interativa.
- Desenvolver o motor computacional do Método Simplex, documentando iterações e expondo a motivação geométrica das decisões algébricas, partindo do princípio de

que o desenvolvimento do método simplex algébrico é baseado em ideias transmitidas pela solução gráfica de PL.

- Desenvolver o motor do algoritmo *Branch-and-Bound* com renderização interativa da árvore de busca, detalhando as relaxações e os critérios de poda.
- Integrar uma Biblioteca de Problemas Clássicos Editáveis para incentivar a exploração ativa.
- Construir componentes de visualização da região viável em 2D e 3D, correlacionando poliedros ao *tableau*.
- Incorporar módulos de pós-otimização para apresentar o problema Dual e calcular a Análise de Sensibilidade.
- Garantir a persistência do estado da aplicação durante transições de módulos e alterações dimensionais.

O Capítulo 2 consolida os fundamentos de Programação Linear e métodos pedagógicos associados. O Capítulo 3 delineia as escolhas tecnológicas e a arquitetura desenvolvida. O Capítulo 4 documenta a codificação dos algoritmos e da camada de interação visual. A avaliação do sistema é o foco do Capítulo 5, enquanto o Capítulo 6 sumariza os resultados obtidos e apresenta diretrizes para expansões futuras da pesquisa.

2 Revisão Bibliográfica

Este capítulo revisa os fundamentos de Pesquisa Operacional e discute o uso da tecnologia educacional no ensino de algoritmos. Em seguida, analisa o estado da arte das ferramentas de visualização para identificar lacunas pedagógicas e justifica a escolha da arquitetura tecnológica proposta para este trabalho.

2.1 Fundamentos de Pesquisa Operacional

Um Problema de Programação Linear (PPL) otimiza uma função objetivo linear sujeita a restrições lineares. Neste trabalho, adota-se a notação matemática padrão onde n representa o número de variáveis de decisão e m representa o número de restrições do modelo. A aplicação do Método Simplex exige a conversão do PPL para a *forma padrão*, um formato que impõe equações nas restrições e não-negatividade nos lados direitos e variáveis. Para alcançar tal formato estrutural, o próprio autor demonstra a necessidade de introduzir variáveis de folga (*variáveis de folga*), de excesso (*variáveis de excesso*) ou artificiais (*variáveis artificiais*), garantindo assim a viabilidade inicial para execução do algoritmo (TAHA, 2017).

O *tableau* Simplex organiza o PPL na forma padrão em uma matriz, viabilizando a execução iterativa do algoritmo. Cada iteração ou pivoteamento é definido pelo autor como uma troca (*swapping*) entre a variável que entra e a que sai da base no *tableau*, cujos cálculos seguem estritamente as operações de linha de Gauss-Jordan:

1. **Verificação de Otimalidade e Regra de Entrada:** A linha de custos reduzidos ($C_j - Z_j$) determina a otimalidade. Em maximização, se todos $C_j - Z_j \leq 0$, o algoritmo termina. Caso contrário, a variável não-básica com maior $C_j - Z_j$ entra na base (coluna pivô).
2. **Regra de Saída (Teste da Razão Mínima):** Para manter a viabilidade ($\mathbf{x} \geq 0$), calcula-se a razão entre o valor da solução (X_{Bi}) e o coeficiente na coluna pivô (y_{ik}), para $y_{ik} > 0$. A menor razão positiva define a variável básica que sai (linha pivô).
3. **Atualização do Quadro:** O elemento na interseção (pivô) orienta operações de linha (Gauss-Jordan) para transformá-lo em 1 e zerar os demais elementos da coluna, gerando um novo *tableau*.

Taha (2017) evidencia que ferramentas didáticas precisam explicar **casos especiais** do Simplex. Estes incluem: **Inviabilidade**, quando variáveis artificiais continuam na base ao final da Fase I; **Solução Ilimitada**, quando a coluna pivô não possui coeficientes positivos; **Ótimos Alternativos**, quando uma variável não-básica apresenta custo reduzido zero no ótimo; e **Degenerescência**, que ocorre em empates no Teste da Razão Mínima.

Geometricamente, as soluções que satisfazem as restrições formam um conjunto

convexo, frequentemente denominado espaço de soluções viáveis. O Teorema Fundamental da Programação Linear estabelece que a solução ótima de um PL, quando existe, está sempre associada a um ponto extremo (vértice) do espaço de soluções limitando a busca por otimização a um número finito de candidatos. A genialidade algorítmica reside no fato de que o caminho do algoritmo Simplex sempre conecta pontos extremos [vértices], movendo-se iterativamente ao longo das arestas do espaço de soluções para melhorar a função objetivo (TAHA, 2017). O algoritmo Simplex apoia-se em ideias geométricas intuitivas, porém a mecânica computacionalmente intrincada do algoritmo pode ofuscar a compreensão geométrica (ROBBINS et al., 2023). Essa dificuldade motiva o desenvolvimento de ferramentas recentes como o GILP, publicadas na SIGCSE, que buscam restaurar a ligação visual e evidenciam uma lacuna pedagógica.

A teoria da Dualidade pareia todo problema de PL (Primal) a um problema simétrico (Dual). Pelo Teorema da Dualidade Fraca, o valor da função objetivo no problema de minimização estabelece um limite superior para o valor da função objetivo no problema de maximização. Por sua vez, o Teorema Forte da Dualidade garante que, se houver ótimo, ambos compartilham o mesmo valor de função objetivo ($Z_{primal} = W_{dual}$).

Pedagogicamente, a teoria da dualidade introduz o *Preço Sombra* (ou preço dual), definido como a taxa de variação da função objetivo por unidade de mudança de um recurso. Em paralelo, a Análise de Sensibilidade investiga a robustez do modelo ao determinar os limites e intervalos em que os parâmetros podem variar sem alterar a solução. O domínio integrado desses conceitos capacita o aluno a ir além do cálculo abstrato, habilitando-o a tomar decisões econômicas fundamentadas sobre o sistema.

A Programação Inteira restringe variáveis a valores inteiros. O *Branch-and-Bound* (B&B) estrutura a busca por essas soluções como uma árvore, baseando-se em três operações:

1. **limitação:** A relaxação linear (Simplex sem restrição de integralidade) gera um limite superior. A melhor solução inteira conhecida forma o limite inferior (incumbente).
2. **ramificação:** Se a relaxação resultar em valor fracionário v_i, o problema ramifica com as restrições $x_i \leq \lfloor v_i \rfloor$ e $x_i \geq \lceil v_i \rceil$.
3. **poda:** Sub-árvores são podadas por limite (resultado pior que o incumbente), inviabilidade (relaxação sem solução), ou integralidade (nova melhor solução inteira encontrada).

2.2 Tecnologia Educacional para Algoritmos Abstratos

O ensino de algoritmos como o Simplex sofre com cálculos manuais lentos que dificultam a autoaprendizagem. Ferramentas de visualização automatizam a matemática para garantir que o educador não gaste mais um tempo valioso em cálculos entediantes, enquanto o aluno compreende conceitos matemáticos complexos (LAZARIDIS; SAMARAS; ZISSOPOULOS, 2003). Além disso, a estratégia *Branch-and-Bound* (B&B) é conside-

rada significativamente difícil de aprender quando comparada a outras estratégias, pois sobrecarrega os alunos com numerosos passos lógicos e árvores de decisão (ZUMAYTIS; KARNALIM, 2017).

A eficácia da Visualização de Algoritmos (AV) correlaciona-se profundamente com o engajamento cognitivo, partindo da premissa de que a forma como os estudantes usam a tecnologia AV tem um impacto maior na sua eficácia do que aquilo que a tecnologia lhes mostra (HUNDHAUSEN; DOUGLAS; STASKO, 2002). Naps et al. (2003) classificam esse engajamento:

- **Nível 1 (No Viewing):** Sem visualização.
- **Nível 2 (Viewing):** Aluno assiste passivamente.
- **Nível 3 (Responding):** Aluno responde a perguntas sobre a execução.
- **Nível 4 (Changing):** Aluno altera parâmetros e observa o impacto no algoritmo (NAPS et al., 2003).
- **Nível 5 (Constructing) e Nível 6 (Presenting):** Aluno cria e apresenta visualizações.

O aprendizado melhora ao avançar do *Viewing* (Nível 2) para o *Changing* (Nível 4), pois estudantes sem conhecimento prévio parecem ganhar mais utilizando visualizações em um nível de engajamento mais alto (MYLLER; LAAKSO; KORHONEN, 2007). A funcionalidade de edição de problemas clássicos proposta neste projeto eleva a interação para o Nível 4.

A biblioteca de problemas editáveis implementa a Aprendizagem Baseada na Investigação (IBL). Esta abordagem concentra-se no engajamento dos estudantes em uma aprendizagem de descoberta ativa e em ambientes que incluem características de participação, ação própria, observação, exploração e experimentação (LAPPAS; KRITIKOS, 2018). Na matemática aplicada, isso permite testar hipóteses sem a barreira do cálculo manual.

2.3 Estado da Arte em Ferramentas de Visualização para PO

O SIMO (Sistema Interativo para Métodos de Otimização) implementa visualizadores *web* passo a passo para o Simplex e o B&B (MALTA et al., 2016).

A arquitetura do SIMO é *client-side*, baseada em JavaScript, jQuery e `glpk.js`. Ela acopla lógica e interface, dificultando a manutenção e limitando a complexidade dos problemas resolvidos pelo navegador.

Existem barreiras instrucionais na usabilidade do SIMO que travam a progressão do aluno:

- **Fluxo Inflexível:** A dimensionalidade inicial é fixa; alterá-la exige reiniciar todo o problema.
- **Superficialidade Didática:** As mensagens detalham as operações, mas omitem a motivação (por que uma linha foi escolhida).
- **Desconexão entre Módulos:** Simplex e B&B não compartilham o problema em

memória, forçando o recadastramento manual dos dados ao trocar de algoritmo.

- **Baixa Granularidade:** A árvore do B&B não expõe a execução do Simplex dentro de cada nó.
- **Ausência de Estado:** Falta persistência da sessão (SHAFFER et al., 2010), exigindo que o aluno controle o histórico de variações externamente.

O GILP (*Geometric Interpretation of Linear Programs*) foca em conectar o Simplex à sua interpretação em 2D e 3D via Python e Plotly (Plotly Technologies Inc., 2026; ROBBINS et al., 2023). Enquanto o GILP é construído ao redor da geometria, este trabalho integra a geometria à execução algébrica passo a passo e à análise de sensibilidade em uma única interface.

O ILESA (*Intelligent Learning Environment for the Simplex Algorithm*) adota uma abordagem de tutoria ativa. O sistema em Java propõe problemas e diagnostica erros do aluno passo a passo (Nível 3) (LÓPEZ et al., n.d.). Esta abordagem difere do foco no laboratório de investigação livre (Nível 4) desta pesquisa.

Visualizadores de análise de performance como VBnB e BAK focam em ajudar pesquisadores a reconhecer padrões de execução e o progresso das árvores de *Branch-and-Bound* (OZALTIN; HUNSAKER; RALPHS, 2007). *Solvers* didáticos clássicos, como o TORA (TAHA, 2017), automatizam o cálculo do *tableau* e chegam a oferecer uma opção iterativa guiada pelo usuário (*user-guided option*), onde o estudante escolhe as variáveis que entram e saem da base. Contudo, essas ferramentas limitam-se ao escopo puramente algébrico; carecem de integração bidirecional em tempo real com representações geométricas e não oferecem persistência de estado visual para experimentação ramificada (*cenários hipotéticos*), justificando a necessidade de abordagens modernas.

A Tabela 1 posiciona a contribuição deste trabalho.

Tabela 1 – Quadro comparativo das ferramentas de visualização de PO.

Ferramenta	**Foco Principal**	**Arquitetura**	**Engajamento Pedagógico**
SIMO (MALTA et al., 2016)	Mecânica passo a passo e árvore B&B.	*client-side* (JS, jQuery, glpk.js)	**Nível 2.** Requer entrada manual completa e sem histórico.
GILP (ROBBINS et al., 2023)	Conexão Álgebra-Geometria	Python e Plotly	**Nível 4.** Focado na visualização geométrica.
ILESA (LÓPEZ et al., n.d.)	Sistema Tutor Inteligente	*Web-based* (Java)	**Nível 3.** Abordagem tutelada.
PROPOSTA	Integração algébrica e geométrica com pós-otimização	***server-side*** (Python, Streamlit, NumPy)	**Nível 4.** Foco em IBL via problemas editáveis e transparência matricial.

Fonte: Autoria própria (2026).

2.4 A Arquitetura Python/Streamlit como Evolução Tecnológica

Sistemas web que dependem de pesada manipulação de DOM via JavaScript frequentemente apresentam limites estruturais e alta carga de manutenção. A adoção de uma arquitetura *server-side* com Python e Streamlit atua para contornar o gargalo tecnológico identificado no SIMO (MALTA et al., 2016).

O Streamlit permite construir aplicações web utilizando lógica Python pura (DATACAMP, 2024). Esta escolha arquitetural entrega três vantagens:

1. **Foco no Domínio Numérico:** Ao evitar código *front-end* (HTML/JS), o desenvolvimento foca na lógica dos algoritmos de Pesquisa Operacional (DATACAMP, 2024; QUANSIGHT, 2020).
2. **Processamento Vetorizado (*white-box*):** A utilização da biblioteca NumPy explora a eficiência nativa do processamento vetorizado para reimplementar os algoritmos do Simplex e *Branch-and-Bound*, substituindo o uso de *solvers* comerciais (*black-box*) que ocultam as operações matriciais.
3. **Isolamento de Estado:** A computação dos algoritmos roda isolada no servidor. O cliente atua estritamente como visualizador dinâmico (KIM, 2025).

3 Metodologia e Arquitetura

A estruturação do projeto traduz as lacunas pedagógicas identificadas no Capítulo 2 em especificações técnicas funcionais e não-funcionais. O percurso metodológico abrange a classificação da pesquisa, a engenharia de requisitos e a definição arquitetural da ferramenta desenvolvida.

3.1 Classificação Metodológica

A pesquisa classifica-se como *Aplicada*, pois objetiva gerar conhecimentos práticos voltados à solução de problemas específicos. De acordo com Gil (2002), as pesquisas dessa natureza são movidas por razões de ordem prática e "decorrem do desejo de conhecer com vistas a fazer algo de maneira mais eficiente ou eficaz". O artefato resultante a plataforma Solver LP atua diretamente nesta eficácia, auxiliando na mitigação das dificuldades de ensino-aprendizagem em Pesquisa Operacional.

Do ponto de vista dos objetivos, a investigação inicia-se como *Exploratória*, fundamentada no mapeamento bibliográfico e na análise de ferramentas correlatas para delimitar as limitações vigentes. Na fase de desenvolvimento, assume caráter *Descritivo* ao especificar a arquitetura, a gestão de estados e a implementação algorítmica da solução proposta.

O processo de desenvolvimento seguiu um ciclo iterativo e incremental, composto pelas seguintes etapas:

1. **Levantamento de Requisitos:** Análise das necessidades pedagógicas e mapeamento das funcionalidades baseadas nas deficiências do benchmark (SIMO).
2. **Prototipagem de Baixa Fidelidade:** Definição dos fluxos de navegação e interfaces principais.
3. **Implementação Algorítmica (*white-box*):** Desenvolvimento do núcleo matemático em Python/NumPy, independente da interface.
4. **Integração e Interface:** Construção da camada de apresentação utilizando o *framework* Streamlit.
5. **Validação:** Testes de usabilidade e verificação da correção matemática dos algoritmos.

3.2 Engenharia de Requisitos

A definição de requisitos do Solver LP estrutura-se para superar as limitações de usabilidade mapeadas no estado da arte. Os requisitos dividem-se em Funcionais (RF), que especificam os comportamentos operacionais do sistema, e Não-Funcionais (RNF), que balizam os atributos de qualidade estrutural e desempenho.

3.2.1 Requisitos Funcionais (RF)

Os requisitos funcionais detalham o escopo da ferramenta, conforme sumarizado na Tabela 2.

Tabela 2 – Lista de Requisitos Funcionais do Solver LP.

ID	Descrição do Requisito
RF01	**Biblioteca de Problemas e Sessão:** O sistema deve prover um acervo de problemas clássicos (ex: Dieta, Mix de Produção) editáveis e manter um histórico de sessão que permita restaurar problemas resolvidos anteriormente.
RF02	**Resolução Passo a Passo (Simplex):** O sistema deve executar o Método Simplex exibindo cada iteração matricial (*tableau*), detalhando o cálculo do elemento pivô, as variáveis de entrada/saída e identificando casos especiais (inviabilidade, solução ilimitada).
RF03	**Visualização Geométrica:** O sistema deve gerar representações gráficas da região factível em 2D (plano cartesiano) e 3D (gráfico interativo rotacionável), destacando as restrições e o vértice ótimo.
RF04	**Árvore *Branch-and-Bound*:** Para problemas de Programação Inteira, o sistema deve construir dinamicamente o grafo da árvore de decisão, permitindo a inspeção de nós e suportando estratégias de busca em Largura (BFS), Profundidade (DFS) e Melhor Limitante (*Melhor Limite*).
RF05	**Análise de Pós-Otimização:** O sistema deve realizar a conversão automática Primal-Dual, calcular Preços Sombra (*Preços Sombra*) e determinar os intervalos de estabilidade para os coeficientes da função objetivo e restrições.
RF06	**Conversão para Forma Padrão:** O sistema deve transformar automaticamente inequações em equações através da inserção de variáveis de folga e excesso, tratando restrições de igualdade e termos independentes negativos.

Fonte: Autoria própria (2026).

3.2.2 Requisitos Não-Funcionais (RNF)

Os requisitos não-funcionais garantem que a ferramenta atenda aos critérios de desempenho e usabilidade pedagógica.

- **RNF01 - Arquitetura Web *server-side*:** O processamento dos algoritmos de otimização deve ocorrer no servidor, desacoplando a lógica da capacidade de processamento do dispositivo do usuário. Isso assegura que problemas complexos de Programação Inteira não sobrecarreguem a capacidade de processamento do navegador do cliente, superando a limitação arquitetural do SIMO.
- **RNF02 - Persistência de Estado (*State Management*):** O sistema deve manter o contexto do problema (matrizes e parâmetros) ativo na memória da sessão. Isso

permite que o usuário navegue entre diferentes módulos (ex: do Simplex para a Dualidade) sem a necessidade de reinserir os dados, reduzindo a fricção de uso.

- **RNF03 - Didática e Transparência (*white-box*):** A aplicação não deve atuar como uma *black-box*. Todas as operações devem ser expostas visualmente. O tempo de resposta deve priorizar a clareza da explicação (texto + visual) em detrimento da velocidade pura de execução.
- **RNF04 - Internacionalização (i18n):** A arquitetura deve suportar múltiplos idiomas (inicialmente Português, Inglês e Espanhol) com troca dinâmica, visando alcance global.

3.3 Arquitetura do Sistema

A arquitetura do Solver LP adota o padrão *Model-View-Controller* (MVC) adaptado para o paradigma reativo do *framework* Streamlit. Diferentemente de aplicações web tradicionais baseadas em ciclo de requisição-resposta (Request-Response), o Streamlit opera em um modelo de execução de *script*, onde a interface é re-renderizada reativamente a alterações no estado.

Um diferencial arquitetural em relação às ferramentas analisadas é a gestão de estado. Em vez de depender do DOM (*client-side*) para persistir dados ao longo das interações, o Solver LP utiliza o `Session State` do Streamlit. Esse recurso opera como um armazenamento lógico no servidor, preservando as matrizes A e os vetores b e c entre os ciclos de re-renderização da interface.

Essa escolha garante reatividade sem perda de contexto: um problema instanciado no módulo "Biblioteca" transita para o "Solver Simplex" e para o gerador de gráficos em memória, mitigando a necessidade de passagem explícita de parâmetros via URL ou processamento local excessivo no navegador.

Os diagramas estruturados no modelo C4 detalham o fluxo de dados, a divisão lógica e a interação entre os componentes do sistema.

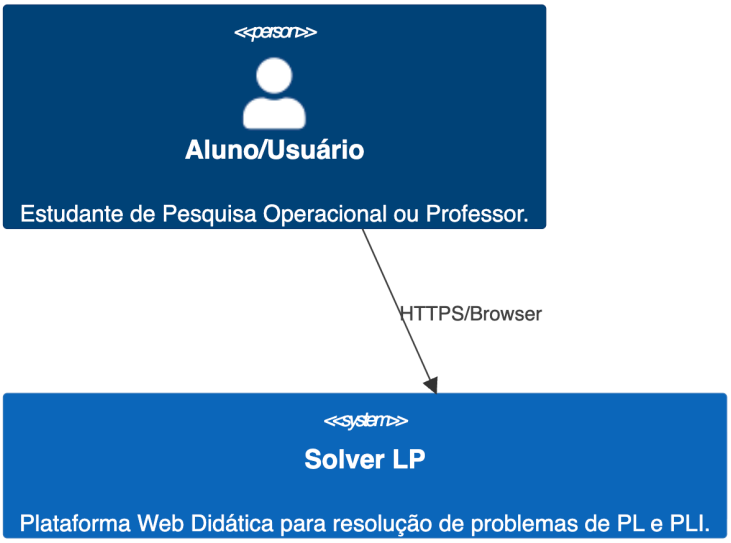
O Diagrama de Container (Figura 2) detalha a separação lógica. O sistema é composto por um *Entrypoint* (`app.py`) que gerencia o roteamento, uma Camada de UI (Componentes Streamlit) que consome o estado, e o Núcleo Matemático (*Core Solvers*) isolado da interface.

O fluxo de execução do algoritmo Simplex (Figura 3) demonstra a abordagem interativa: o usuário solicita um passo, o controlador invoca o método `step()` da classe `SimplexSolver`, e o estado atualizado do *tableau* é retornado para renderização.

3.4 Tecnologias e Ferramentas

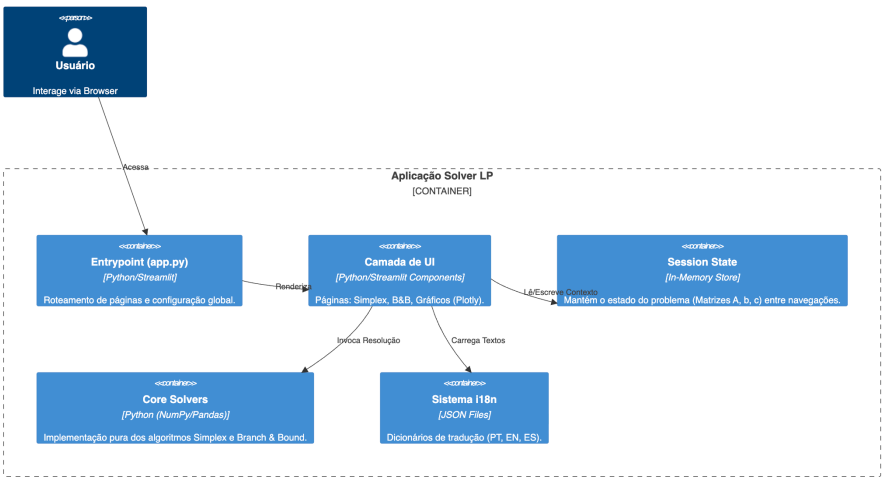
A *stack* tecnológica priorizou manutenibilidade, eficiência no desenvolvimento e desempenho no processamento algorítmico numérico:

Figura 1 – Diagrama de Contexto (Nível 1) do Solver LP.



Fonte: Autoria própria (2026).

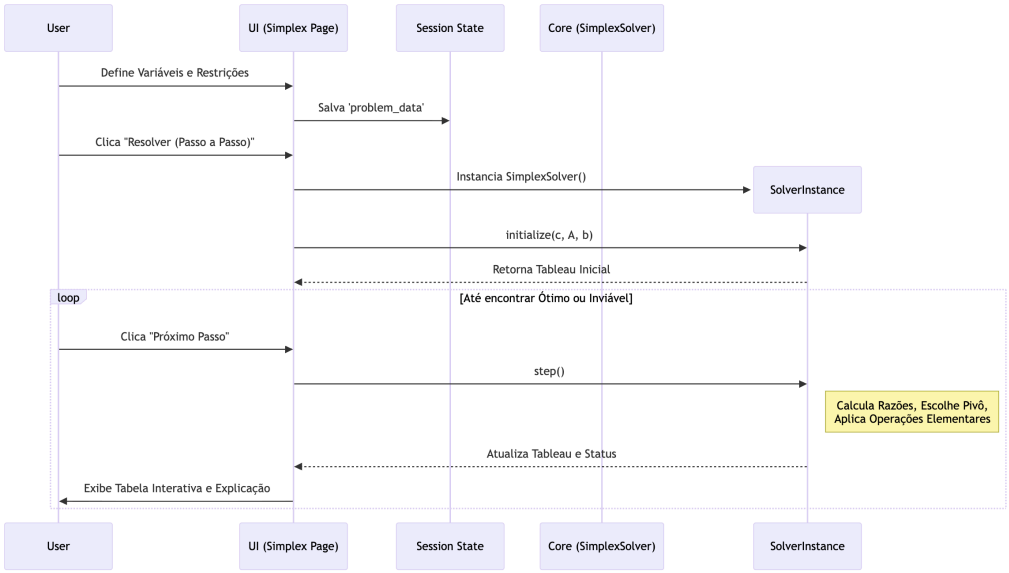
Figura 2 – Diagrama de Container (Nível 2) evidenciando a arquitetura modular.



Fonte: Autoria própria (2026).

- **Linguagem Python (3.10+):** Fornece um ecossistema nativo consolidado para cálculo científico e uma sintaxe que favorece a legibilidade, critério essencial para um projeto educacional de código aberto.
- ***framework* Streamlit:** O Streamlit isola a complexidade de roteamento e renderização *frontend* tradicional (HTML/CSS/JS). Seu modelo de reatividade permite

Figura 3 – Diagrama de Sequência do fluxo de execução passo a passo do Simplex.



Fonte: Autoria própria (2026).

concentrar o esforço de engenharia na integridade dos cálculos do núcleo matemático, viabilizando interfaces dinâmicas voltadas a dados.

- **NumPy** (HARRIS et al., 2020): Atua como o motor de álgebra do sistema. A utilização de matrizes alocadas contiguamente em memória (`ndarrays`) suporta operações vetorizadas de forma altamente otimizada, mandatórias para as manipulações intensivas das linhas do *tableau* no método Simplex.
- **Plotly** (Plotly Technologies Inc., 2026): Adotada para a renderização geométrica (RF03). Diferentemente de saídas rasterizadas estáticas, gera gráficos vetoriais (SVG/WebGL) interativos em tempo de execução. Isso permite rotacionar a região factível em 3D e mapear vértices inspecionando pontos do plano via eventos de foco (*hover*).

3.5 Estratégia de Implementação Algorítmica (*white-box*)

A não utilização de *solvers* consolidados em arquiteturas *black-box* (como `scipy.optimize` e `PuLP`) foi uma decisão intencional. Embora apresentem alta performance, essas bibliotecas abstraem as etapas intermediárias e retornam unicamente a solução ótima convergida, inviabilizando a proposta educacional de acompanhar a evolução algébrica.

Para satisfazer a Transparência exigida pelo RNF03, desenvolvemos uma arquitetura ***white-box***. Os métodos Simplex e *Branch-and-Bound* foram programados do zero sobre a infraestrutura do NumPy, garantindo exposição visual total das matrizes iteradas sem encapsular regras de parada algorítmica.

A classe `SimplexSolver` manipula diretamente uma matriz NumPy que representa o *tableau*. A cada iteração, o algoritmo:

1. Calcula os custos reduzidos ($C_j - Z_j$) acessando a última linha da matriz.
2. Identifica a coluna pivô via `argmax` (para maximização).
3. Realiza o Teste da Razão Mínima dividindo o vetor b pela coluna pivô (operações vetorizadas).
4. Executa o pivoteamento (Operações Elementares de Linha) atualizando toda a matriz via *propagação vetorial* do NumPy.

Esta abordagem permite extrair o estado exato do sistema em qualquer ponto da execução, viabilizando a visualização passo a passo.

Para a Programação Inteira, a classe `BranchBoundSolver` gerencia a árvore de busca. A estrutura de dados utiliza uma fila de prioridade para armazenar os nós ativos. A cada passo, o algoritmo:

- Seleciona um nó com base na estratégia de busca (DFS, BFS ou Melhor Limite).
- Invoca uma instância da classe `SimplexSolver` para resolver a relaxação linear daquele nó específico.
- Analisa o resultado para decidir pela ramificação (criação de restrições de corte $x_i \leq \lfloor v \rfloor$ e $x_i \geq \lceil v \rceil$) ou pela poda (por limite, inviabilidade ou integralidade).

Essa implementação modular assegura que o aluno possa visualizar não apenas a árvore final, mas o raciocínio de ramificação e poda em tempo real.

4 Implementação e Desenvolvimento

A implementação da arquitetura técnica (Capítulo 3) priorizou o atendimento aos requisitos do sistema com foco na modularidade, manutenibilidade e experiência do usuário (UX). O desenvolvimento adotou uma abordagem ágil, assistida por ferramentas de inteligência artificial generativa para prototipagem e refatoração.

4.1 Ambiente de Desenvolvimento e Ferramentas

O ecossistema de ferramentas uniu a eficiência no processamento matemático (*backend*) à interatividade da camada de apresentação (*frontend*).

A linguagem **Python** (versão 3.10+) suportou a construção do sistema, escolhida por sua forte aderência à computação científica e facilidade de prototipagem. O **Antigravity** (Google) atuou como ambiente de desenvolvimento integrado (IDE), orquestrando o uso de agentes de IA para refatorar os componentes complexos do *backend*.

O `pip` gerenciou as dependências, fixadas em um arquivo `requirements.txt` para assegurar a reprodutibilidade. O controle de versão utilizou o **Git**, aplicando *commits* atômicos para documentar a evolução do código.

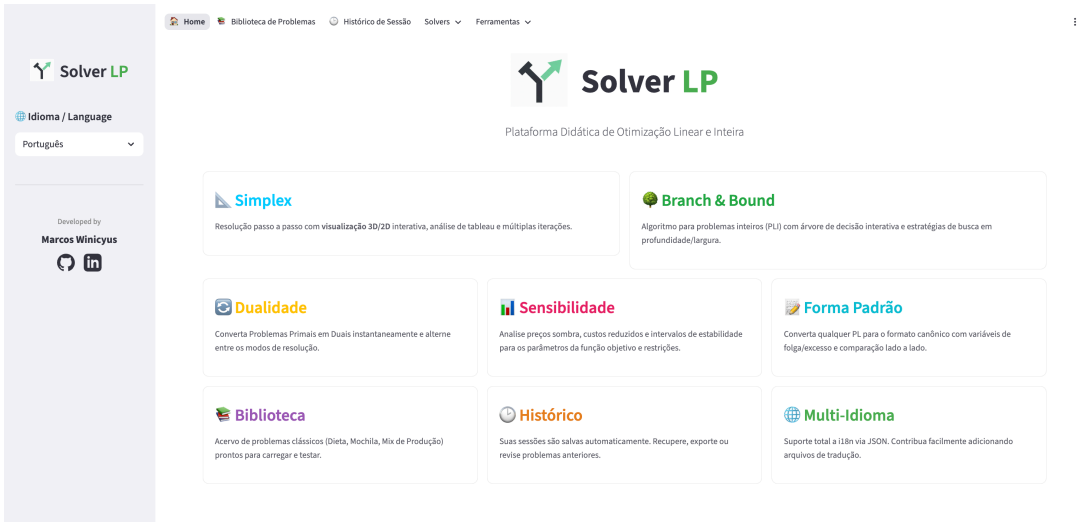
O código-fonte completo encontra-se disponibilizado publicamente sob licença de código aberto, alinhado aos princípios da Ciência Aberta, no repositório:

<https://github.com/MarcosWinicyus/solver-lp>

A aplicação apoia-se em quatro bibliotecas principais do ecossistema Python:

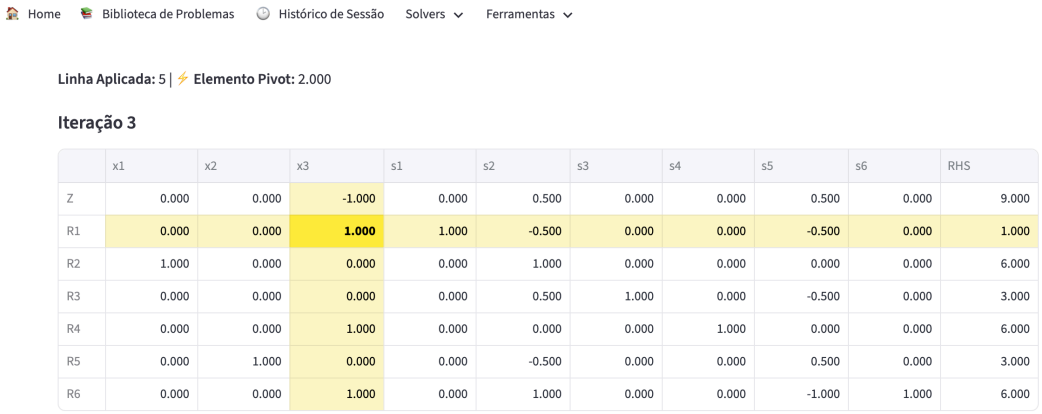
- **Streamlit (*frontend*):** Gerencia a interface web. Ao permitir a criação de uma aplicação reativa a partir de *scripts* Python, a ferramenta eliminou a escrita de HTML/CSS/JavaScript, acelerando o ciclo de implementação da UI (Figura 4).
- **NumPy (Motor Matemático)** (HARRIS et al., 2020): Executa a álgebra linear. A representação matricial do *tableau* Simplex e as operações de pivoteamento utilizam a vetorização do NumPy, o que assegura um desempenho superior ao dos laços de repetição nativos da linguagem.
- **Pandas (Estruturação de Dados)** (MCKINNEY, 2010): Formata resultados e tabelas na camada de apresentação. A biblioteca viabilizou a estilização condicional (como o destaque visual da linha e coluna pivô) na interface (Figura 5).
- **Plotly (Visualização)** (Plotly Technologies Inc., 2026): Renderiza as regiões factíveis em 2D e 3D interativamente. Essa escolha permite inspeções geométricas diretas, como a identificação em tempo real de coordenadas e vértices (Figura 6).

Figura 4 – Tela inicial do sistema Solver LP.



Fonte: Autoria própria (2026).

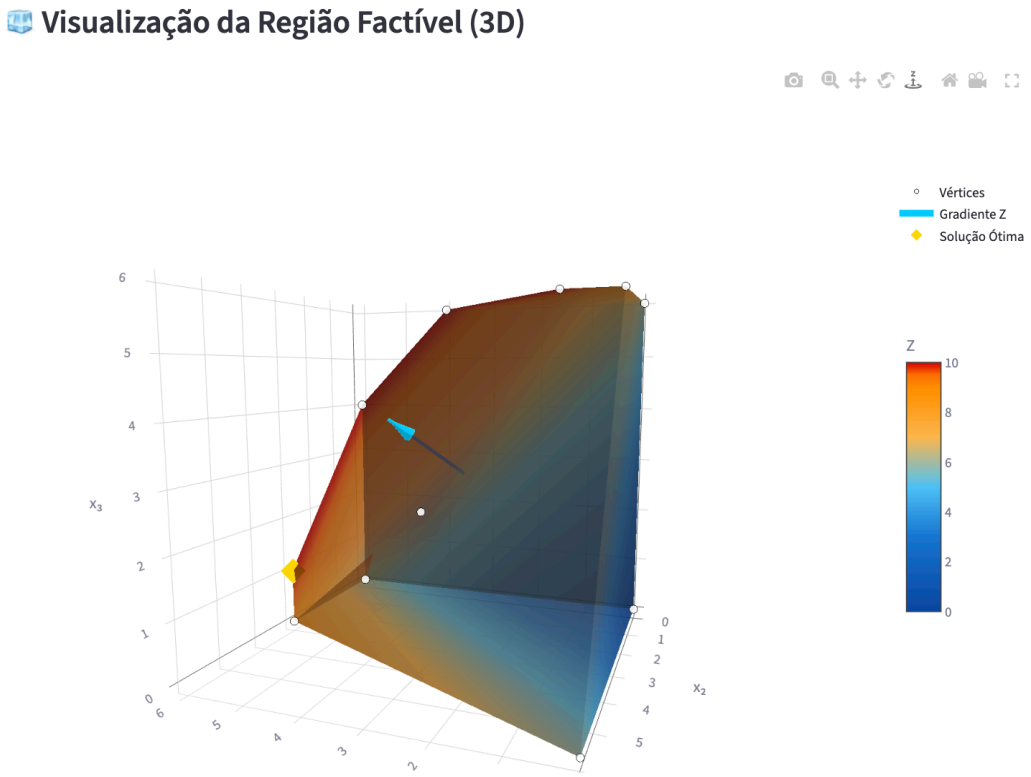
Figura 5 – Tabela do Simplex renderizada com Pandas e marcação de pivôs.



	x1	x2	x3	s1	s2	s3	s4	s5	s6	RHS
Z	0.000	0.000	-1.000	0.000	0.500	0.000	0.000	0.500	0.000	9.000
R1	0.000	0.000	**1.000**	1.000	-0.500	0.000	0.000	-0.500	0.000	1.000
R2	1.000	0.000	0.000	0.000	1.000	0.000	0.000	0.000	0.000	6.000
R3	0.000	0.000	0.000	0.000	0.500	1.000	0.000	-0.500	0.000	3.000
R4	0.000	0.000	1.000	0.000	0.000	0.000	1.000	0.000	0.000	6.000
R5	0.000	1.000	0.000	0.000	-0.500	0.000	0.000	0.500	0.000	3.000
R6	0.000	0.000	1.000	0.000	1.000	0.000	0.000	-1.000	1.000	6.000

Fonte: Autoria própria (2026).

Figura 6 – Visualização 3D da região factível e gradiente com Plotly.



Fonte: Autoria própria (2026).

4.2 O Motor de Otimização "*white-box*" (*backend*)

A pasta `core/` concentra os algoritmos de otimização, construídos a partir de princípios básicos (*from scratch*). Optamos por essa abordagem *white-box* no lugar de pacotes comerciais de prateleira (*black-box*) justamente para garantir a introspecção de cada etapa do cálculo, um requisito fundamental para as demonstrações didáticas da ferramenta.

O algoritmo Simplex processa uma matriz com os coeficientes numéricos do problema de otimização linear. A implementação representa esse *tableau* por meio de um *array* bidimensional do NumPy (`np.ndarray`). O Código 1 ilustra a inicialização dinâmica dessa matriz, com a alocação do tamanho correspondente às variáveis de decisão (x_n), de folga (s_m) e artificiais (a_m).

Algoritmo 1: Pseudocódigo de Inicialização do *Tableau*

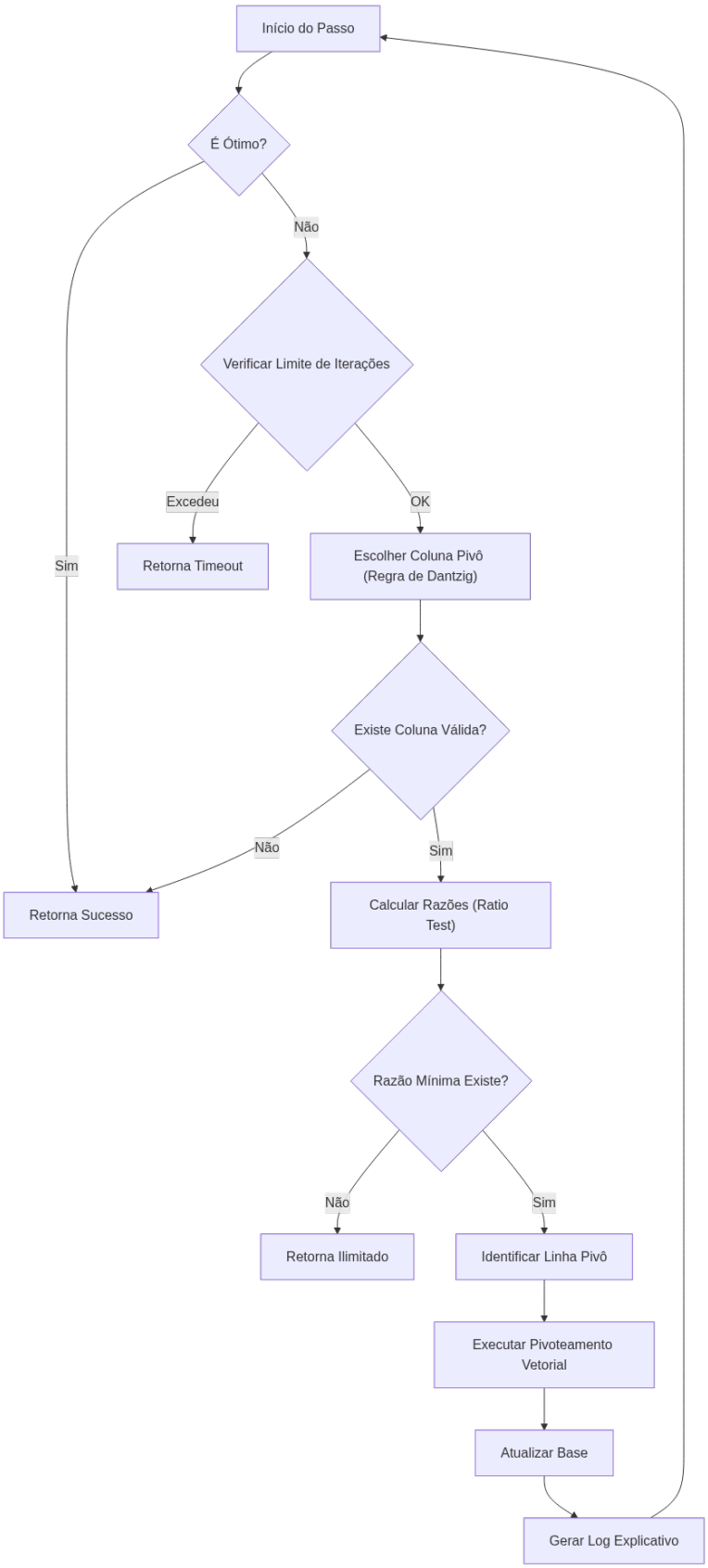
1: $total\_vars \leftarrow n + n\_slack + n\_surplus + n\_artificial$
2: Criar matriz T preenchida com zeros de tamanho $(m + 1) \times (total\_vars + 1)$
3: **for** CADA variável de decisão j de 0 até $n - 1$ FAÇA **do**
4: $\quad T[0, j] \leftarrow -c[j]$ {Preencher Linha do Objetivo}
5: **end for**
6: **for** CADA índice idx NA lista de variáveis artificiais FAÇA **do**
7: $\quad T[0, idx] \leftarrow M$ {Penalidade do Método Big-M}
8: **end for**

Fonte: Autoria própria (2026).

4.2.1 Implementação do Algoritmo Simplex

A classe `SimplexSolver` gerencia o ciclo de vida da otimização. A função `step()` executa uma única iteração da Eliminação de Gauss-Jordan, permitindo que o usuário avance passo a passo e compreenda a mecânica da tabela. O fluxo de execução, que inclui o teste da razão e os critérios de parada, é documentado na Figura 7.

Figura 7 – Fluxograma do método `step()` detalhando a lógica de decisão a cada iteração.



Fonte: Autoria própria (2026).

A atualização do *tableau* exige atenção especial quanto à eficiência. Evitamos a lentidão natural dos laços `for` aninhados (que, no Python puro, custariam $O(m \times n)$) utilizando o *propagação vetorial* e as operações matriciais do NumPy. O Código 2 mostra como o trecho computa e recarrega os dados numéricos inteiros na memória de uma só vez, ao invés de atualizar as células individualmente.

Algoritmo 2: Pseudocódigo da Operação de Pivoteamento Vetorial

$T2 \leftarrow$ Cópia da Matriz T
$ElementoPivô \leftarrow T[pr, pc]$
$T2[pr] \leftarrow T[pr] \,/\, ElementoPivô$ {Normaliza a linha do pivô}
for CADA linha i NA matriz T FAÇA **do**
  if $i \neq pr$ **then**
    $Fator \leftarrow T[i, pc]$
    $T2[i] \leftarrow T[i] - (Fator \times T2[pr])$ {Operação vetorial na linha}
  end if
end for
return $T2$

Fonte: Autoria própria (2026).

O método `_log_iteration` gera a narrativa textual exigida pelo **RF02 (Explicação Passo a Passo)**. Ele mapeia e processa os índices matemáticos das variáveis de entrada e saída, convertendo-os em matrizes de caracteres descritivos.

4.2.2 Implementação do *Branch-and-Bound*

A classe `BranchBoundSolver` implementa a árvore de busca para problemas de Programação Inteira. Uma fila (`self.queue`) armazena os nós pendentes, abstraindo a navegação em largura (BFS) e em profundidade (DFS).

A etapa de ramificação (*ramificação*) localiza a primeira variável fracionária e gera dois subproblemas paralelos, adicionando as restrições de limite inferior ($\lfloor x_i \rfloor$) e superior ($\lceil x_i \rceil$) (Código 3).

Algoritmo 3: Pseudocódigo da Lógica de Ramificação (*Branch-and-Bound*)

1: $indice\_frac \leftarrow$ Buscar a primeira variável com valor fracionário na solução do nó atual
2: $x\_val \leftarrow Solucao[indice\_frac]$
3: **for** CADA limite em $\{(\leq, \lfloor x\_val \rfloor), (\geq, \lceil x\_val \rceil)\}$ FAÇA **do**
4: $novos\_limites \leftarrow$ Cópia dos limites atuais do nó
5: $novos\_limites[indice\_frac] \leftarrow$ limite
6: $sub\_A, sub\_b \leftarrow$ Aplicar $novos\_limites$ às matrizes globais A e b
7: Resolver a relaxação linear para o novo subproblema
8: **end for**

Fonte: Autoria própria (2026).

A etapa de poda (*poda*) descarta o nó investigado caso o custo ótimo relaxado seja inferior à melhor solução inteira identificada anteriormente na mesma árvore de busca, abortando cálculos inúteis.

4.2.3 Análise de Desempenho e Complexidade Computacional

A integração do modelo *white-box* com a biblioteca NumPy resultou no balanço essencial entre desempenho algorítmico e rastreabilidade didática. Enquanto a adoção do Python nativo com listas esbarraria em gargalos significativos de repetição iterativa na memória, o pacote NumPy delega as instruções matriciais densas para rotinas otimizadas em C e Fortran internamente (BLAS/LAPACK).

Para instâncias acadêmicas ($n \leq 10, m \leq 10$), a vetorização do pivoteamento ocorre em milissegundos. Do ponto de vista analítico, o Método Simplex apresenta uma complexidade iterativa de $\mathcal{O}(m \times n)$. Esse custo computacional advém das operações matriciais necessárias a cada iteração, especificamente na execução do teste da razão (percorrendo as m linhas) e na posterior atualização dos coeficientes do *tableau* (processando m linhas e n colunas) via eliminação de Gauss-Jordan. Embora seja eficiente na prática, o Simplex possui um pior caso assintótico exponencial, comportamento classicamente demonstrado pelo hipercubo de Klee-Minty, onde o algoritmo é forçado a percorrer iterativamente os vértices da região factível.

No que tange à otimização discreta, os problemas de Programação Linear Inteira (PLI) enquadram-se na classe matemática NP-Difícil. A árvore de busca do algoritmo *Branch-and-Bound* possui complexidade exponencial no pior caso, podendo atingir até 2^n subproblemas explorados. Essa explosão combinatória justifica a necessidade crítica dos algoritmos de poda (*pruning*), que descartam ramificações subótimas para viabilizar a resolução computacional.

Dessa forma, a latência perceptível relatada na interface (Seção 5.6) não advém

da complexidade teórica do cálculo matricial. O atraso advém, na realidade, da latência de comunicação da rede e da renderização integral da página requisitada a cada recálculo, sendo um reflexo inevitável do estilo arquitetural de persistência adotado pelo Streamlit.

4.3 Desenvolvimento da Interface e Interatividade (*frontend*)

O paradigma de controle utilizado pelo Streamlit reexecuta todo o escopo do arquivo *script* a cada iteração do usuário com os elementos gráficos. Esse comportamento de renderização demandou o desenho e implementação de uma camada adicional de persistência para travar as variáveis e evitar a perda de processamento em tela.

Para preservar e controlar os dados do problema (**RNF01** e **RNF02**), o estado atual do objeto `solver` fica protegido na memória do dicionário `st.session_state` da interface web. Essa estratégia técnica anula os problemas de contexto que frequentemente limitavam ferramentas em estágios iniciais de desenvolvimento, garantindo fluidez (Código 4).

Algoritmo 4: Pseudocódigo para Persistência de Estado e Histórico

if usuário clicou em Resolver **then**
  $solver \leftarrow$ Instanciar algoritmo Simplex
  Executar $solver.resolver(c, A, b)$
  $Sessao[\text{“}solver\text{”}] \leftarrow solver$ {Salva o estado na memória web}
  if *solver* encontrou solução ótima **then**
    $sol, z \leftarrow solver.obter\_solucao()$
    Adicionar $\{\text{“}metodo\text{”} : \text{“}Simplex\text{”}, \text{“}solucao\text{”} : sol, \text{“}z\text{”} : z\}$ ao $Sessao[\text{“}historico\text{”}]$
  end if
end if

Fonte: Autoria própria (2026).

Os controles numéricos do usuário acompanham em tela a dimensão real da modelagem parametrizada. O laço de renderização produz os campos dinamicamente conforme a proporção de variáveis (n) exigida.

4.4 Implementação da Biblioteca de Problemas

A Biblioteca de Problemas Editáveis (RF01) oferece a base de *dados simulados* de dados da ferramenta para aplicar o modelo da Aprendizagem Baseada na Investigação (IBL, Seção 2.2). A configuração de *configurações predefinidas* clássicos (como o modelo do Mix de Produção) ocorre mediante dicionários que guardam tanto as dimensões dos vetores A, b, c quanto a respectiva carga de texto educativo.

Uma vez invocado o dicionário pelo usuário na tela inicial, os registros enchem

o espaço no `session_state` e levam a sessão imediatamente para o painel de otimização interativo. Lá, o aluno possui plena autoridade de edição para alterar, ajustar e testar limites dos parâmetros de cada variável, testando hipóteses matemáticas (referente ao Nível 4 de exploração da taxonomia de Naps).

4.5 Módulo de Visualização Geométrica

A conversão lógica e visual de inequações no gráfico da região factível (RF03) baseou-se na manipulação matricial associada ao Plotly. O desafio do desenvolvimento resumiu-se à captura geométrica dos cortes para demarcar corretamente uma área inteira.

A rotina de validação na função `find_vertices` explora e mapeia as interseções exatas de *restrições* simultâneas na matriz principal, computando assim os cantos limítrofes.

Algoritmo 5: Pseudocódigo do Algoritmo para Encontrar Vértices da Região Factível

vertices ← Lista Vazia
for CADA combinação de n_vars linhas da matriz A_all e do vetor b_all FAÇA **do**
    A_sub ← submatriz quadrada formada pelas linhas da combinação
    b_sub ← subvetor formado pelas linhas da combinação
    if determinante de $A\_sub \neq 0$ **then**
        x ← solução do sistema linear $A\_sub \times x = b\_sub$
        if $A\_all \times x \leq b\_all + \epsilon$ **then**
            Adicionar x à lista *vertices* {O ponto é factível}
        end if
    end if
end for
return *vertices*

Fonte: Autoria própria (2026).

Após o filtro matricial descartar os vértices infratores do sistema, os pontos que persistem geram polígonos validados no escopo do gráfico 2D ou 3D. Essa abordagem renderiza imediatamente as mudanças dos limites no Plotly, fornecendo assim uma conexão contínua de entendimento entre a lógica computacional subjacente e a malha gráfica interativa.

5 Resultados e Discussão

Este capítulo valida a plataforma Solver LP através de estudos de caso que acompanham o fluxo algébrico e gráfico de resolução. Apresentamos também um comparativo com o SIMO, destacando as diferenças de arquitetura e usabilidade que afetam a aplicação de metodologias de ensino.

5.1 Validação Funcional por Estudo de Caso

A validação simula o roteiro de um estudante utilizando o clássico "Problema do Mix de Produção". O exemplo permite verificar o funcionamento do algoritmo Simplex integrado à visualização geométrica bidimensional.

O problema selecionado visa maximizar o lucro Z da produção de dois produtos (x_1 e x_2), sujeitos a restrições de disponibilidade de recursos (horas de máquina e matéria-prima). O modelo matemático formal é definido como:

$$\begin{aligned} \text{Maximizar} \quad & Z = 100x_1 + 150x_2 \\ \text{Sujeito a:} \quad & 2x_1 + 3x_2 \leq 120 \\ & 1x_1 + 0.5x_2 \leq 50 \\ & x_1, x_2 \geq 0 \end{aligned} \tag{1}$$

Ferramentas tradicionais frequentemente iniciam com uma grade matricial vazia. No Solver LP, a funcionalidade Biblioteca de Problemas (RF09) permite carregar modelos pré-configurados, conforme a Figura 8. A partir daí, o aluno edita os parâmetros c_j ou b_i livremente.

Figura 8 – Interface da Biblioteca de Problemas, ilustrando a seleção e o carregamento de modelos pré-definidos.

Fonte: Autoria própria (2026).

Isso facilita a aplicação da Aprendizagem Baseada na Investigação (IBL). O estudante concentra-se em modificar as restrições e observar o comportamento do modelo em vez de configurar matrizes manualmente.

5.1.1 Execução do Simplex e Explicações Didáticas

Durante a resolução, o Solver LP exibe o *tableau* a cada iteração. A Figura 9 mostra a terceira iteração do problema de teste. Em vez de apresentar apenas a matriz numérica atualizada, o sistema gera uma explicação textual fundamentada na abordagem *white-box* detalhada no Capítulo 4.

Figura 9 – Visualização do *Tableau* Simplex com destaque para a explicação textual gerada automaticamente sobre a escolha do pivô.

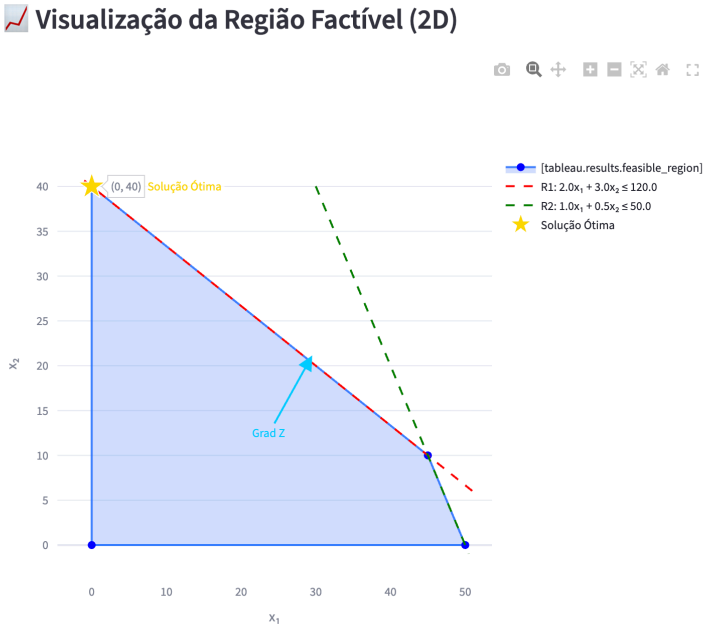
Fonte: Autoria própria (2026).

O sistema identifica explicitamente:

- A **Variável de Entrada** (baseada no critério de maior custo reduzido positivo).
- A **Variável de Saída** (baseada no Teste da Razão Mínima).
- O **Elemento Pivô** e as operações elementares realizadas.

Simultaneamente à tabela, o módulo visual plota a região factível e o trajeto do algoritmo pelos vértices (RF03), conforme a Figura 10.

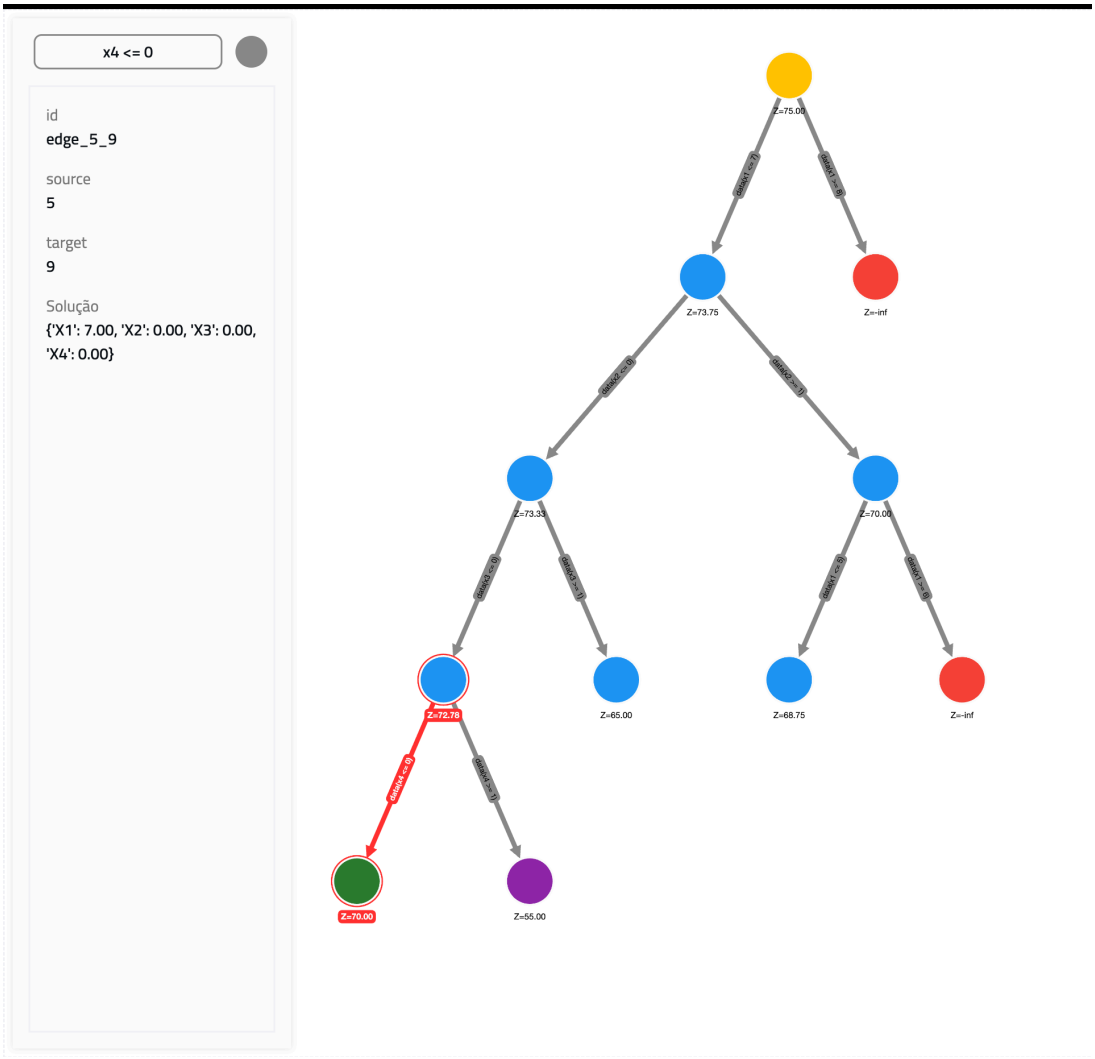
Figura 10 – Representação geométrica da região factível, evidenciando o percurso do algoritmo pelos vértices do poliedro.



Fonte: Autoria própria (2026).

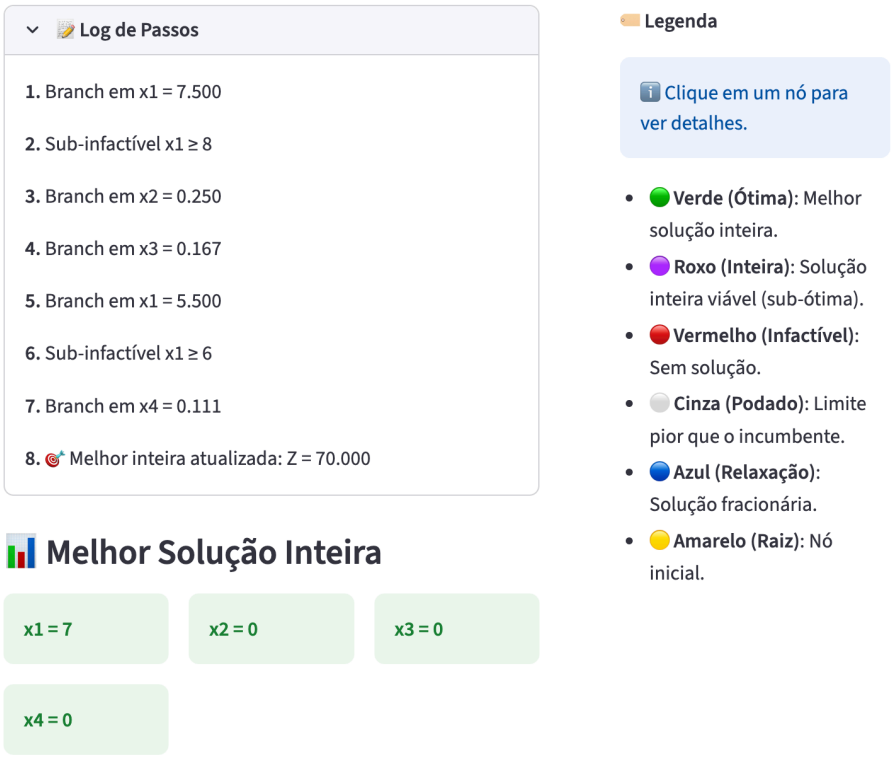
Para testar o módulo de Programação Inteira (RF02 e RF05), o modelo foi restrito ao domínio inteiro ($x_1, x_2 \in \mathbb{Z}$). A Figura 11 ilustra a árvore de ramificação gerada. A navegação interativa e a legenda de cores (Figura 12) permitem inspecionar o estado de cada nó. O acesso às justificativas dos nós podados apoia a compreensão de critérios técnicos, como poda por limite e por viabilidade integral.

Figura 11 – Árvore de busca do *Branch-and-Bound* gerada pelo sistema, ilustrando ramificações e podas.



Fonte: Autoria própria (2026).

Figura 12 – Detalhe da interface de auxílio ao aluno: convenção semântica de cores para os estados dos nós e painel de feedback textual justificando o critério de poda aplicado.



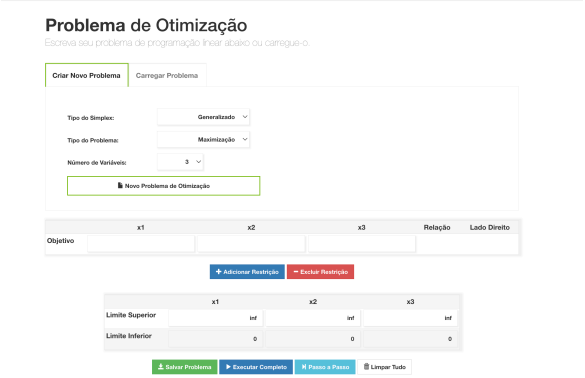
Fonte: Autoria própria (2026).

5.2 Análise Comparativa: Solver LP e SIMO

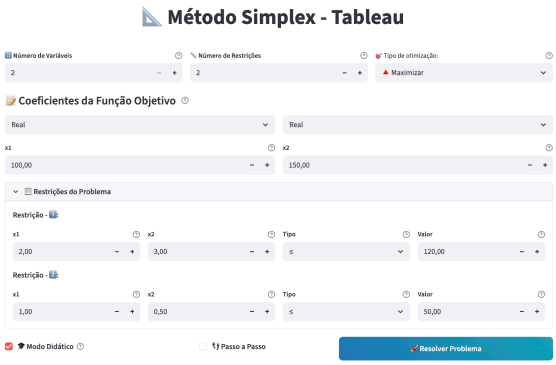
A evolução em relação às limitações estruturais do SIMO representa uma das contribuições técnicas deste projeto. A seção contrasta as abordagens de usabilidade e arquitetura de ambas as ferramentas.

O SIMO exige abstração matricial prévia e impõe um fluxo de dados rígido. A Figura 13 compara as duas interfaces de modelagem inicial.

Figura 13 – Comparativo de Usabilidade: (a) A interface do SIMO exige abstração matricial prévia; (b) O Solver LP oferece formulários dinâmicos e intuitivos.



(a) SIMO: Entrada Matricial Rígida



(b) Solver LP: Entrada Parametrizada

Fonte: Autoria própria (2026).

No Solver LP, a adoção de formulários dinâmicos e a gestão de estado do Streamlit (RNF01) permitem adicionar variáveis ou restrições a qualquer momento. Isso evita a perda de dados durante a modelagem, uma limitação comum no ambiente *client-side* estático do SIMO.

A Tabela 3 sumariza as diferenças estruturais e funcionais, relacionando a pilha tecnológica baseada em Python aos recursos de ensino implementados.

Tabela 3 – Quadro comparativo técnico-pedagógico: SIMO vs. Solver LP.

Aspecto	**SIMO (Benchmark)**	**Solver LP (Proposto)**
Arquitetura	*client-side* (JavaScript/jQuery). Processamento limitado pelo navegador do cliente.	*server-side* (Python/Streamlit). Processamento robusto com NumPy e suporte a sessões.
Entrada de Dados	Matriz estática. Alterar dimensões apaga os dados inseridos.	Formulário dinâmico. Permite adição incremental de variáveis mantendo o estado (RNF01).
Feedback Didático	Exibe apenas o resultado numérico da operação.	Exibe o *porquê* da operação (ex: razão do teste mínimo), gerando texto explicativo (RF02).
Visualização	Inexistente.	Gráficos interativos (Plotly) sincronizados com a iteração atual do algoritmo (RF03/RF04).
Pós-Otimização	Inexistente.	Relatórios completos de sensibilidade e dualidade (RF06/RF07).

Fonte: Autoria própria (2026).

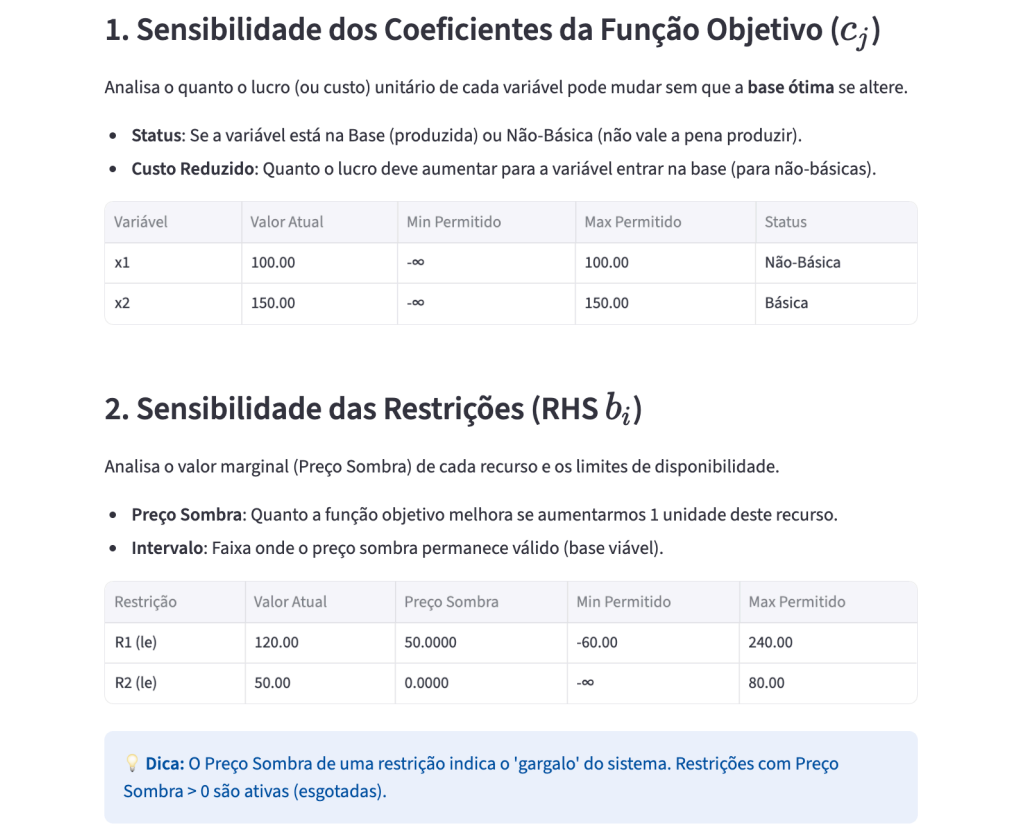
5.3 Discussão Pedagógica: Nível de Engajamento

Os testes funcionais corroboram o suporte da ferramenta ao Nível 4 (*Changing*) da taxonomia de Naps et al. (2003). Segundo os autores, este nível de engajamento "implica em modificar a visualização [...] permitindo ao aprendiz alterar os dados de entrada do algoritmo sob estudo para explorar o seu comportamento em casos diferentes". No Solver LP, a manipulação de parâmetros (c_j, b_i) com resposta visual imediata materializa essa premissa, incentivando a formulação de hipóteses por parte do usuário. Ao invés de focar no valor exato da função objetivo, o estudante pode analisar o impacto geométrico da restrição de um recurso.

Isso viabiliza cenários práticos onde o objetivo é entender a estrutura do problema, reduzindo a carga cognitiva associada ao recálculo manual ou reconfiguração de interfaces rígidas.

A etapa de pós-otimização complementa a interpretação econômica dos modelos. A Figura 14 apresenta a análise de sensibilidade para o Mix de Produção.

Figura 14 – Relatório de Análise de Sensibilidade, apresentando os Preços Sombra e os intervalos de estabilidade dos parâmetros.



Fonte: Autoria própria (2026).

O cálculo automático dos **Preços Sombra** (*Preços Sombra*) e dos limites de esta-

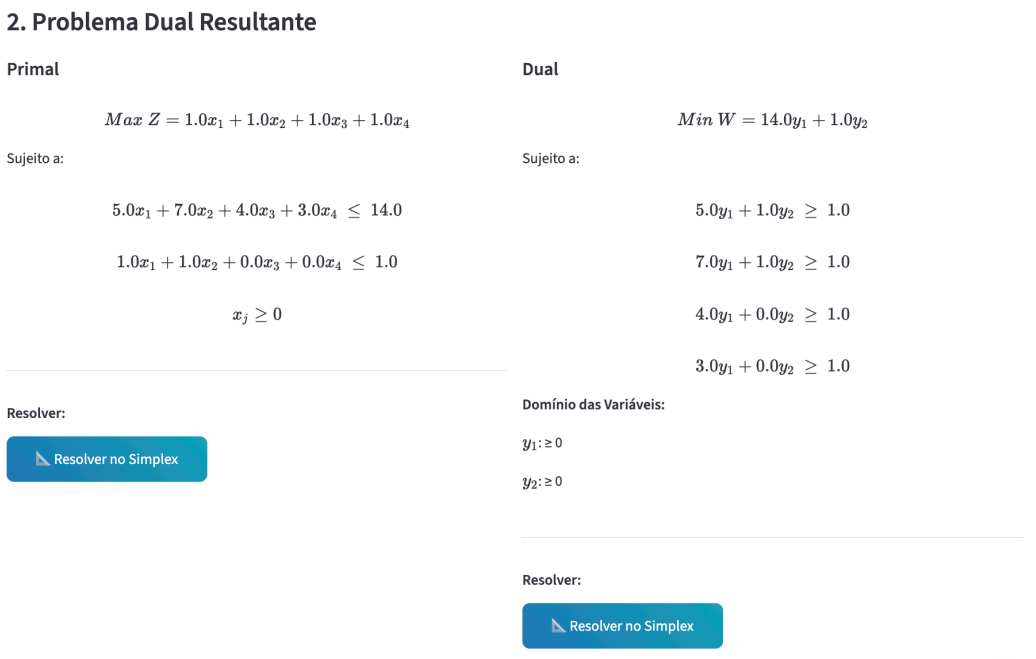
bilidade quantifica o impacto marginal dos recursos. A ferramenta formata esses valores em relatórios tabulares, eliminando a necessidade de extração manual a partir da matriz final do Simplex.

5.4 Conversão Automática Primal-Dual

A transcrição manual do modelo Dual muitas vezes acarreta erros de conversão algébrica. O **Conversor Automático Primal-Dual** (RF06) agiliza essa etapa aplicando as relações da tabela de Tucker para transpor o sistema.

A Figura 15 ilustra os dois modelos gerados e exibidos em conjunto.

Figura 15 – Visualização comparativa entre o modelo Primal e o Dual gerado automaticamente.



Fonte: Autoria própria (2026).

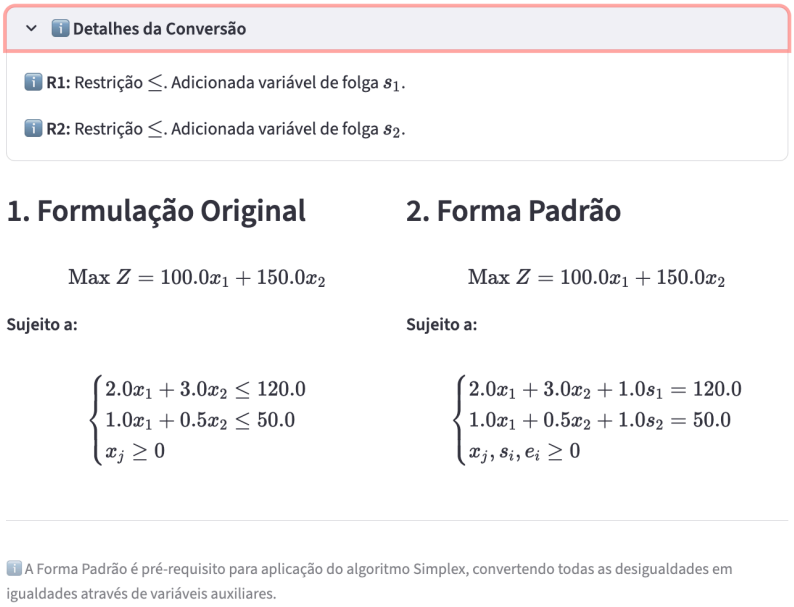
No contexto do SIMO, o usuário deve reiniciar a aplicação e reescrever os coeficientes se desejar analisar o problema Dual correspondente. O Solver LP utiliza a sessão em memória (RNF01) para manter e converter os dados automaticamente, integrando o fluxo de estudo.

5.5 Pré-Processamento e Forma Padrão

O algoritmo de pré-processamento detalha a inserção de variáveis de folga e artifícios algébricos (RF08) antes da execução do Simplex. Essa abordagem de caixa-branca

exibe o sistema de equações expandido (Figura 16), destacando as variáveis adicionadas (s_i, e_i, a_i) para assegurar igualdades e limites não-negativos ($b_i \geq 0$).

Figura 16 – Exibição do problema convertido para a Forma Padrão, evidenciando a inserção de variáveis de folga e artificiais.



Fonte: Autoria própria (2026).

Omitir a fase de padronização, como observado em outras ferramentas, esconde conceitos estruturais como a ativação de restrições de desigualdade. Apresentar o PPL convertido fortalece o elo entre a modelagem textual inicial e a matriz do primeiro *tableau*.

5.6 Limitações da Ferramenta

As escolhas técnicas do projeto incorrem em algumas limitações mapeadas:

1. **Conectividade:** O *backend* do Streamlit exige acesso à internet contínuo para atualizar componentes, limitando o uso local isolado caso não seja executado um servidor na máquina do estudante.
2. **Restrição Dimensional:** O escopo está bloqueado para PPLs de até 10 variáveis e restrições. Embora a infraestrutura NumPy seja escalável, matrizes maiores perdem legibilidade na interface, desvirtuando o foco didático do *software*.
3. **Latência de *WebSocket*:** A comunicação contínua necessária para gestão de estado pode apresentar atrasos na renderização do *frontend* se exposta a instâncias com dezenas de ramificações ou tabelas enormes simultâneas, uma limitação do encapsulamento Web utilizado.

6 Considerações Finais

O desenvolvimento do Solver LP alcança o objetivo central proposto no início desta pesquisa: a construção de um laboratório digital de Pesquisa Operacional focado em mitigar as limitações arquiteturais e pedagógicas de ferramentas anteriores, como o sistema SIMO.

A pesquisa evidencia que a dificuldade no aprendizado do Método Simplex e do algoritmo *Branch-and-Bound* não decorre apenas da complexidade matemática intrínseca, mas da desconexão prática entre o cálculo matricial mecânico e a intuição geométrica na avaliação dos vértices do poliedro. A ferramenta desenvolvida reduz essa lacuna ao sincronizar visualmente as etapas algébricas com suas representações gráficas.

A literatura documenta que ferramentas de resolução estáticas limitam a curva de aprendizado do aluno. A implementação da abordagem *white-box* no sistema desenvolvido garante a transparência do processo de resolução a cada passo algorítmico.

A validação da arquitetura confirma o desenvolvimento de um ambiente que:

- Reduz a fricção na entrada de dados por meio da Biblioteca de Problemas Editáveis;
- Mantém a persistência do contexto de estudo via gestão de estado (*Session State*);
- Sincroniza as alterações nos parâmetros do modelo com a atualização gráfica da região factível e da árvore de decisão.

A integração da Biblioteca de Problemas Editáveis altera a forma como o aluno interage com os modelos matemáticos. A manipulação de coeficientes e restrições com atualização visual simultânea atende aos requisitos do Nível 4 (*Changing*) da taxonomia de Naps et al. (2003). Esse formato viabiliza a Aprendizagem Baseada na Investigação (IBL), transferindo o foco cognitivo da execução aritmética para a análise de sensibilidade e exploração das variáveis do problema.

A adoção de uma arquitetura *server-side* com Python e Streamlit resolve limitações de processamento matemático encontradas em implementações *client-side* baseadas em JavaScript. Centralizar a execução algorítmica no servidor com a biblioteca NumPy assegura maior estabilidade no cálculo matricial e viabiliza uma base de código modular, facilitando a manutenção e a escalabilidade da aplicação.

Para eliminar barreiras de configuração e compatibilidade de *hardware*, o sistema adota o modelo de aplicação web. A plataforma está implantada no *Streamlit Community Cloud*, permitindo acesso direto via navegador sem a necessidade de instâncias locais.

Link de Acesso Oficial: <https://solver-linear-programing.streamlit.app>

Como parte dos resultados, o código-fonte integral da aplicação foi disponibilizado publicamente, configurando um artefato de *software* documentado e extensível.

A aplicação opera sob a licença **GNU General Public License v3.0 (GPLv3)**. A

adoção dessa licença de *copyleft* garante que o projeto e quaisquer trabalhos derivados mantenham o código aberto. Essa estrutura legal viabiliza o uso, estudo e modificação do software por outros pesquisadores e instituições.

Além da transparência na interface do usuário (*white-box*), a implementação em Python nativo permite que alunos de graduação inspecionem a tradução da álgebra linear computacional em código de produção, contrastando com ferramentas comerciais de código fechado.

O uso de Python e Streamlit reduz a curva de aprendizado para a manutenção do projeto, facilitando contribuições de outros desenvolvedores.

Repositório GitHub: <https://github.com/MarcosWinicyus/solver-lp>

A arquitetura do sistema suporta nativamente a internacionalização (i18n). A gestão de traduções é feita através de dicionários isolados em arquivos JSON, permitindo a adição de novos idiomas sem interferência no núcleo algorítmico de otimização.

6.1 Trabalhos Futuros

As características arquiteturais do projeto viabilizam as seguintes evoluções na ferramenta:

- **Módulos de Gamificação:** Desenvolvimento de desafios progressivos onde o aluno deve otimizar o número de iterações do Simplex ou identificar falhas lógicas injetadas em modelos pré-definidos.
- **Expansão de Idiomas:** Mapeamento e tradução colaborativa da interface para incluir novos idiomas via arquivos de internacionalização.
- **Integração de *Solvers* Externos:** Implementação de adaptadores para motores de otimização industriais (como HiGHS, IBM CPLEX, COIN, GLPK e Gurobi), capacitando a ferramenta a processar instâncias massivas que ultrapassam a capacidade didática do cálculo com NumPy.

6.2 Conclusão

A aplicação de técnicas de engenharia de software no contexto educacional possibilita a abstração de operações matemáticas repetitivas em favor do raciocínio analítico. O Solver LP atua como um recurso complementar ao estudo teórico da Pesquisa Operacional, delegando o cálculo exaustivo ao computador para que o aluno se concentre na modelagem e interpretação dos dados. O código aberto e a arquitetura visual do projeto oferecem uma infraestrutura prática para a compreensão profunda dos algoritmos lineares e sua aplicação na resolução de problemas estruturados.

Referências

DATACAMP. *What Is Streamlit? A Beginner's Guide to Building Interactive Web Apps in Python*. 2024. Tutorial. Disponível em: <https://www.datacamp.com/tutorial/streamlit>. Acesso em: jun. 2026. Citado 2 vezes nas páginas 3 e 9.

DESHPANDE, P. et al. Automated result analysis using python and streamlit. *ResearchGate*, May 2025. Preprint. Disponível em: <https://www.researchgate.net/publication/392263000>. Citado na página 3.

GIL, A. C. *Como elaborar projetos de pesquisa*. 4. ed. São Paulo: Atlas, 2002. ISBN 9788522431694. Citado na página 10.

HARRIS, C. R. et al. Array programming with NumPy. *Nature*, Springer Science and Business Media LLC, v. 585, n. 7825, p. 357–362, setembro 2020. Citado 2 vezes nas páginas 14 e 16.

HUNDHAUSEN, C. D.; DOUGLAS, S. A.; STASKO, J. T. A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages & Computing*, v. 13, n. 3, p. 259–290, 2002. Citado 2 vezes nas páginas 2 e 7.

KIM, M. Integrating streamlit into english teacher education: Pedagogical possibilities and practice. *Studies in Foreign Language Education*, v. 39, n. 3, p. 1–25, 2025. Disponível em: <https://www.researchgate.net/publication/395944906>. Citado na página 9.

LAPPAS, P. Z.; KRITIKOS, M. N. Teaching and learning numerical analysis and optimization: A didactic framework and applications of inquiry-based learning. *Higher Education Studies*, v. 8, n. 1, p. 46–61, 2018. Citado 2 vezes nas páginas 2 e 7.

LAZARIDIS, V.; SAMARAS, N.; ZISSOPOULOS, D. Visualization and teaching simplex algorithm. In: *Proceedings. IEEE International Conference on Advanced Learning Technologies (ICALT 2003)*. [S.l.: s.n.], 2003. p. 270–271. Citado 2 vezes nas páginas 2 e 6.

LÓPEZ, J. M. et al. ILESA: a web-based intelligent learning environment for the simplex algorithm. In: *Artigo de Conferência não especificada (Fonte: ResearchGate)*. [S.l.: s.n.], n.d. Disponível em: <https://www.researchgate.net/publication/2518479>. Acesso em: jun. 2026. Citado na página 8.

MALTA, A. F. R. et al. SIMO: Uma ferramenta online para auxiliar o ensino de otimização. In: *Anais do XLVIII Simpósio Brasileiro de Pesquisa Operacional (SBPO)*. Vitória, ES, Brasil: [s.n.], 2016. p. 742–752. Citado 5 vezes nas páginas 2, 3, 7, 8 e 9.

MCKINNEY, W. Data structures for statistical computing in python. In: WALT, S. van der; MILLMAN, J. (Ed.). *Proceedings of the 9th Python in Science Conference*. [S.l.: s.n.], 2010. p. 56–61. Citado na página 16.

MYLLER, N.; LAAKSO, M.-J.; KORHONEN, A. Analyzing engagement taxonomy in collaborative algorithm visualization. In: *Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education (ITiCSE '07)*. [S.l.: s.n.], 2007. p. 251–255. Citado na página 7.

NAPS, T. L. et al. Exploring the role of visualization and engagement in computer science education. *ACM SIGCSE Bulletin*, v. 35, n. 2, p. 131–152, 2003. Citado 4 vezes nas páginas 3, 7, 32 e 35.

NumPy Developers. *NumPy: The fundamental package for scientific computing with Python*. 2026. Página inicial. Disponível em: <https://numpy.org/>. Acesso em: jun. 2026. Citado na página 3.

OZALTIN, O. Y.; HUNSAKER, B.; RALPHS, T. K. Visualizing branch-and-bound. In: *8th INFORMS Computing Society Conference*. [S.l.: s.n.], 2007. p. 1–15. Disponível em: <http://coral.ie.lehigh.edu/~ted/files/papers/BBVis.pdf>. Citado na página 8.

Plotly Technologies Inc. *Plotly: Low-Code Data App Development*. 2026. Página inicial. Disponível em: <https://plotly.com/>. Acesso em: jun. 2026. Citado 3 vezes nas páginas 8, 14 e 16.

QUANSIGHT. *Dash, Voila, Panel, Streamlit: Our thoughts on the big four dashboarding tools*. 2020. Blog Post. Disponível em: <https://quansight.com/post/dash-voila-panel-streamlit-our-thoughts-on-the-big-four-dashboarding-tools>. Acesso em: jun. 2026. Citado 2 vezes nas páginas 3 e 9.

ROBBINS, H. W. et al. GILP: An interactive tool for visualizing the simplex algorithm. In: *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2023)*. Toronto, ON, Canada: ACM, 2023. p. 1–12. Citado 3 vezes nas páginas 2, 6 e 8.

SHAFFER, C. A. et al. Algorithm visualization: The state of the field. *ACM Transactions on Computing Education*, v. 10, n. 3, p. 1–22, 2010. Citado na página 8.

Streamlit Inc. *Streamlit: A faster way to build and share data apps*. 2026. Página inicial. Disponível em: <https://streamlit.io/>. Acesso em: jun. 2026. Citado na página 3.

TAHA, H. A. *Operations Research: An Introduction*. 10. ed. [S.l.]: Pearson, 2017. Citado 5 vezes nas páginas 1, 2, 5, 6 e 8.

ZUMAYTIS, S.; KARNALIM, O. Introducing an educational tool for learning branch & bound strategy. In: *2017 International Conference on Soft Computing, Intelligent System and Information Technology (ICSIIT)*. [S.l.: s.n.], 2017. p. 1–6. Citado 2 vezes nas páginas 2 e 7.