UNIVERSIDADE DE CAXIAS DO SUL
ÁREA DO CONHECIMENTO DE CIÊNCIAS EXATAS E ENGENHARIAS

LEONARDO MARTELLI OLIVEIRA

DESENVOLVIMENTO DE UMA APLICAÇÃO WEB PARA O ENSINO DE MÉTODOS DE OTIMIZAÇÃO TOPOLÓGICA

CAXIAS DO SUL

2023

LEONARDO MARTELLI OLIVEIRA

DESENVOLVIMENTO DE UMA APLICAÇÃO WEB PARA O ENSINO DE MÉTODOS DE OTIMIZAÇÃO TOPOLÓGICA

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Ciência da Computação na Área do Conhecimento de Ciências Exatas e Engenharias da Universidade de Caxias do Sul.

Orientador: Prof. Dr. André Luis Martinotto
Coorientador: Prof. Dr. Leandro Luis Corso

CAXIAS DO SUL

2023

LEONARDO MARTELLI OLIVEIRA

DESENVOLVIMENTO DE UMA APLICAÇÃO WEB PARA O ENSINO DE MÉTODOS DE OTIMIZAÇÃO TOPOLÓGICA

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Bacharel em Ciência da Computação na Área do Conhecimento de Ciências Exatas e Engenharias da Universidade de Caxias do Sul.

Aprovado em 28/11/2023

BANCA EXAMINADORA

_____

Prof. Dr. André Luis Martinotto
Universidade de Caxias do Sul - UCS

_____

Prof. Dr. Leandro Luis Corso
Universidade de Caxias do Sul - UCS

_____

Prof. Me. Giovanni Ely Rocco
Universidade de Caxias do Sul - UCS

_____

Prof. Dr. Alexandre Vieceli
Universidade de Caxias do Sul - UCS

Aos meus pais Janete e Ricardo

“Lembre-se de olhar para as estrelas e não para os pés. Tente entender o que você vê e se pergunte sobre o que faz o universo existir. Seja curioso.”

Stephen Hawking

RESUMO

A otimização estrutural busca definir uma estrutura ótima, por meio de diferentes metodologias, destacando-se a entre elas a Otimização Topológica. Esse método visa obter uma topologia ótima cumprindo requisitos estruturais e distribuindo material em regiões de maior demanda. As etapas que compõem a Otimização Topológica utilizam conceitos matemáticos complexos e de difícil compreensão, gerando dificuldades nos processos de aprendizado. Uma das abordagens mais utilizadas para o auxílio do ensino destes conceitos, é a utilização de ferramentas computacionais que ofereçam recursos gráficos e interativos. Entretanto, existem poucas ferramentas acessíveis para tal fim. Portanto, neste trabalho foi desenvolvida uma ferramenta educacional que auxilia no ensino da Otimização Topológica. Essa ferramenta foi desenvolvida utilizando uma arquitetura *Backend For Frontend*. O *Backend* é responsável pelo processo de otimização, e foi desenvolvido na linguagem de programação *Python*. A otimização topológica foi executada por meio da utilização do pacote *TopOpt*, que implementa o método SIMP. O *Frontend* é responsável pela interface e interação com usuário. Esse é executado em um navegador *Web* e foi desenvolvido utilizando a linguagem *TypeScript* em conjunto com *React.js*. Foram realizados testes utilizando as estruturas de viga MBB, viga suspensa e suporte em L. Os resultados obtidos foram comparados com os resultados do código educacional *99 lines*, desenvolvido por Sigmund O, estando de acordo no que se refere a critérios estruturais e numéricos.

Palavras-chave: Otimização Topológica. Ferramentas de Aprendizagem.

LISTA DE FIGURAS

LISTA DE ALGORITMOS

LISTA DE ABREVIATURAS E SIGLAS

PL	Programação Linear
PNL	Programação Não Linear
OT	Otimização Topológica
MEF	Método dos Elementos Finitos
SIMP	*Solid Isotropic Material with Penalization*
LSM	*Level-Set Method*
OE	Otimização Evolutiva
RBF	*Radial Basis Function*
MQI	Multi Quadrática Inversa
ESO	*Evolutionary Structural Optimization*
BESO	*Bidirectional Evolutionary Structural Optimization*
DSC	*Deformable Simplicial Complex*
FIFO	*First In, First Out*
API	*Application Programming Interface*
JSON	*JavaScript Object Notation*
SVG	*Scalable Vector Graphics*
MBB	*Messerschmitt-Bolkow-Blohm*

LISTA DE SÍMBOLOS

$f(\mathbf{x})$	Função objetivo
$\mathbf{x}$	Vetor de variáveis de projeto Vetor de tensores de rigidez
$g(\mathbf{x})$	Restrição de desigualdade
$h(\mathbf{x})$	Restrição de igualdade
S	Região viável
$f(\mathbf{x}, \mathbf{u})$	Função de flexibilidade
$h(\mathbf{x}, \mathbf{u})$	Restrição de equilíbrio
$g(\mathbf{x}, \mathbf{u})$	Restrição de volume
$\mathbf{u}$	Vetor de deslocamentos
$\mathbf{f}$	Vetor de forças aplicadas
V	Limite fixo de volume
e	Índice do elemento
x_e	Tensor de rigidez do elemento e
K	Matriz de rigidez
x_0	Propriedades base do material
Ω	Domínio de referência
Ω^{mat}	Subdomínio material
Ω^{vaz}	Subdomínio vazio
ω	Ponto no domínio Ω Coordenada do nodo
$\text{Vol}(x)$	Função de volume
ρ	Densidade variável
$\rho(\omega)$	Função de densidade

t	Fator de penalização
$\rho_{\min}$	Densidade miníma
ρ_e^{novo}	Densidade atualizada
B_e	Valor de condição de otimização
ζ	Limite de máximo de mudança de densidade
η	Parâmetro de redução de oscilação
Λ	Multiplicador de Lagrange
$\tilde{\rho}_e$	Densidade filtrada
$\hat{W}$	Fator de ponderação
$r_{\min}$	Raio de filtragem mínimo
dist(e, i)	Distância entre os elementos e e i
$\phi(\omega)$	Função *Level-set*
Γ	Limites da geometria
c	Constante de nível
$r(\omega)$	Função base
s	Constante de forma
ψ	Coeficiente de expansão
τ	Pseudo tempo
R	Matriz de funções base
Ψ	Vetor de coeficiente de expansão
$\hat{\boldsymbol{B}}$	Vetor de velocidades
ε	Tensão dos elementos
$\delta(\phi)$	Esquema de controle de evolução
Δ	Constante balizadora de nível
γ	Índice do elemento no limite Γ
$\nabla\phi$	Norma do gradiente

$H(\phi)$	Função degrau
Vol_0	Volume inicial
$\text{Vol}_{\max}$	Volume máximo
σ_e	Tensão do elemento e
$\sigma_{\max}$	Tensão máxima
FR	Fator de remoção
FI	Fator de inserção
r_{EE}	Constante do estado estável EE de remoção
r_{NO}	Constante do estado de oscilação NO de remoção
i_{EE}	Constante do estado estável EE de inserção
i_{NO}	Constante do estado de oscilação NO de inserção
EE	Estado estável
NO	Estado de oscilação
IP	Índice de performance
V_e	Volume do elemento e
L	Dimensão de referência
X_e	Conjunto de valores discretos
γ_e	Multiplicador de inserção
$\mathbf{q_a}$	Número de elementos ligados ao elemento e pré-validação
$\mathbf{q_p}$	Número de elementos ligados ao elemento e pós-modificação

SUMÁRIO

1 INTRODUÇÃO

A engenharia é uma das ciências mais importantes para o desenvolvimento de uma sociedade, contribuindo significativamente para a solução de problemas complexos nas mais diferentes áreas como, por exemplo, a construção civil, tecnologia da informação, medicina, produção industrial, entre outras (BAILLIE, 2009).

De acordo com Cocian (2016), a solução de problemas complexos é formada por diferentes etapas, que incluem a formulação, análise, pesquisa, decisão e especificação. Entre essas destacam-se as etapas de pesquisa e de decisão, visto que são nestas etapas que a solução é estipulada e selecionada. Em cada uma dessas etapas, diferentes métodos podem ser utilizados para a seleção das possíveis soluções e para uma comparação entre elas. Neste sentido, uma das técnicas mais utilizada para esse propósito, é a otimização (FLOUDAS, 2009).

Amplamente utilizada nas áreas de engenharia, finanças, indústria, logística e biologia molecular, a otimização busca a definir uma solução ótima, seguindo critérios pré-definidos (TSAI *et al.*, 2012). De acordo com Floudas (2009), o desenvolvimento das técnicas de otimização dividiu-se em três períodos. No primeiro período, não existia um método geral para minimizar ou maximizar uma função, existindo apenas técnicas especiais, como por exemplo, para funções quadráticas. O segundo período teve início em 1646, quando Pierre de Fermat, determinou que os pontos de mínimo e máximo de uma função poderiam ser obtidos considerando os pontos onde a derivada de uma função é igual a zero (COE, 1942).

No entanto, foi a partir do terceiro período, com o desenvolvimento da Programação Linear (PL), que surgiram as técnicas de otimização mais utilizadas atualmente. A PL consiste em um conjunto de métodos para a solução de problemas de minimização e maximização de uma função linear (função objetivo), sujeita a restrições também lineares (DANTZIG, 1963). A partir da metade do século 20, deu-se o início do desenvolvimento dos métodos de Programação Não Linear (PNL), que introduzem a utilização de funções não lineares para a definição da função objetivo e das restrições.

A PNL é uma ferramenta amplamente utilizada em problemas de otimização estrutural. A solução de um problema de otimização estrutural consiste na definição da melhor estrutura, entre todas as possíveis, considerando os objetivos e as restrições geométricas e/ou comportamentais previamente definidas (OLHOFF; TAYLOR, 1983).

Um problema de otimização estrutural pode ser classificado, considerando as mudanças estruturais, em: paramétricos, de forma ou topológicos. Em uma otimização paramétrica as características do contorno não são modificadas, ou seja, apenas as dimensões que definem a geometria são alteradas. Em uma otimização de forma, há mudança no contorno e nas dimensões da estrutura. Por fim, em uma otimização topológica, ocorre uma alteração na sua topologia, gerando vazios na estrutura (PERINI, 2013).

A Otimização Topológica (OT) consiste em calcular a distribuição espacial mais favorável de um material, de forma a satisfazer os requisitos estipulados por uma aplicação (carga, pressão e/ou torção, por exemplo), criando vazios em áreas menos solicitadas e acúmulo de material em áreas mais solicitadas (BENDSØE; SIGMUND, 2004). Como exemplo, pode-se citar a utilização da OT na industria aeronáutica para determinar a topologia ótima de um pilone[1] de um Airbus A350, objetivando reduzir a sua massa (REMOUCHAMPS *et al.*, 2011).

Apesar do conceito geral de OT (otimizar o uso de material em regiões de maior demanda) ser de fácil entendimento, a compreensão e a utilização desses métodos é considerada complexa, devido principalmente a base matemática que sustenta a área (OLIVEIRA *et al.*, 2019). De acordo com Tyflopoulos, Haskins e Steinert (2021) uma das metodologias mais eficientes para a aprendizagem desses conceitos matemáticos é utilização de ferramentas interativas e gráficas que exemplifiquem e ilustrem a OT.

Atualmente, existem ferramentas educacionais voltadas para o ensino da OT, no entanto o acesso a essas ferramentas é restrito. De fato, frequentemente essas ferramentas são fornecidas como *softwares* independentes ou *plugins* de *softwares* comerciais, apresentando um elevado custo (REDDY *et al.*, 2016). Assim, neste trabalho será desenvolvida uma aplicação *Web*, de acesso gratuito, para facilitar o ensino de OT.

1.1 OBJETIVOS

O principal objetivo deste trabalho consiste no desenvolvimento de uma aplicação *Web* para o ensino e testes de Otimização Topológica. Com base no objetivo geral, foram definidos os seguintes objetivos específicos:

- definir os métodos de OT que serão implementados;
- escolher as ferramentas e bibliotecas que serão utilizadas para a implementação da aplicação;
- desenvolver uma aplicação *Web* para a execução e visualização de projetos de Otimização Topológica;
- definir casos de teste para validação da aplicação *Web*;

[1] Suporte responsável por fixar a turbina da aeronave.

- validar a aplicação *Web* a partir dos casos de teste.

1.2 ESTRUTURA DO TRABALHO

O presente trabalho está organizado da seguinte forma:

- Este capítulo apresentou uma introdução sobre o trabalho, incluindo a sua motivação, os objetivos geral e específicos.
- No Capítulo 2 são apresentados métodos de solução para um problema de Otimização Topológica.
- No Capítulo 3 é apresentada uma implementação para uma aplicação para o ensino de Otimização Topológica e os resultados obtidos.
- Por fim, no Capítulo 4 são apresentadas considerações finais e sugestões de trabalho futuro.

2 OTIMIZAÇÃO TOPOLÓGICA

O conceito de Otimização Topológica foi introduzido por Bendsøe e Kikuchi (1988), como um método de distribuição de material, gerando topologias ótimas para estruturas contínuas. Os métodos de OT alteram a topologia da estrutura, ao determinar quais pontos do espaço terão um acúmulo de material e quais serão vazios (BENDSØE; SIGMUND, 2004).

Em um projeto de OT, assim como em qualquer projeto de otimização, deve-se, primeiramente definir o problema a ser otimizado perante o modelo geral de otimização. Desta forma, na Seção 2.1 é apresentado o modelo geral de otimização. Posteriormente, na Seção 2.2, é descrito o procedimento de definição de um problema de OT. Por fim, na Seção 2.3 são apresentados os métodos mais utilizados para a solução de problemas de OT.

2.1 MODELO GERAL DE OTIMIZAÇÃO

A otimização consiste na definição de uma ou mais funções objetivo, as quais são submetidas a uma ou mais restrições (ARORA, 2016). A função objetivo é uma representação do que será otimizado, sendo utilizada como uma medida de efetividade (HAFTKA; GÜRDAL, 1992). A função objetivo pode ser de minimização, onde busca-se o menor valor possível (como por exemplo o custo de um processo) ou de maximização, onde busca-se o maior valor possível (como por exemplo o lucro em uma operação).

O modelo geral da otimização pode ser descrito por meio de uma função objetivo $f(\mathbf{x})$, parametrizada por n variáveis de projetos, representadas por um vetor $\mathbf{x} = (x_1, x_2, ..., x_n)$. Assim, a função objetivo pode ser escrita como

$$f(\mathbf{x}) = f(x_1, x_2, ..., x_n) \tag{2.1}$$

sujeito a um conjunto de restrições de igualdade $h(\mathbf{x})$ e um conjunto de restrições de desigualdade $g(\mathbf{x})$, onde $h_j(\mathbf{x})$ representa a j-ésima restrição de igualdade e $g_i(\mathbf{x})$ representa a i-ésima restrição de desigualdade

$$\begin{aligned} h_j(\mathbf{x}) &= h_j(x_1, x_2, ..., x_n) = 0; \quad j = 1, 2, ..., l \\ g_i(\mathbf{x}) &= g_i(x_1, x_2, ..., x_n) \leq 0; \quad i = 1, 2, ..., m \end{aligned} \tag{2.2}$$

Em um problema de otimização estrutural, as variáveis de projeto são os parâmetros que controlam os fatores geométricos, propriedades dos materiais e dimensões de uma estrutura. Essas variáveis podem ser separadas em contínuas e discretas. As variáveis contínuas podem assumir qualquer valor contínuo dentro de um intervalo, como por exemplo, a quantidade de massa da estrutura. As variáveis discretas correspondem a valores isolados dentro de um conjunto finito de opções, como por exemplo, a espessura de uma peça em um projeto, que pode assumir os valores de 10, 15 ou 20 milímetros (HAFTKA; GÜRDAL, 1992).

As restrições em um modelo de otimização são os limites das variáveis de projeto (HAFTKA; GÜRDAL, 1992). Essas restrições são separadas em restrições de igualdade, como por exemplo, o deslocamento de uma peça que deve ser igual a 0, e restrições de desigualdade, onde, por exemplo, a quantidade de um material não pode ser maior do que um limite estipulado. Uma solução é considerada viável somente se todos os requisitos forem satisfeitos (ARORA, 2016).

A região viável de um problema de otimização compreende em todas as possíveis soluções de um conjunto S que satisfazem as restrições de igualdade $h_j(\mathbf{x})$ e de desigualdade $g_i(\mathbf{x})$ (Equação 2.3). Assim, ao adicionar restrições ao sistema, a região viável diminui e, consequentemente, menos soluções possíveis podem ser consideradas.

$$S = (\mathbf{x}|h_j(\mathbf{x}) = 0, j = 1, ..., l; g_i(\mathbf{x}) \leq 0, i = 1, ..., m;). \tag{2.3}$$

De acordo com Arora (2016), na definição de um modelo de otimização devem ser considerados:

- As funções $f(\mathbf{x})$, $h_j(\mathbf{x})$ e $g_i(\mathbf{x})$ devem ser dependentes implicitamente ou explicitamente de uma ou mais variáveis de projeto. As restrições que não apresentam dependência das variáveis de projeto podem ser ignoradas.
- O número de condições de igualdade l deve ser menor ou igual ao número de variáveis de projeto n ($l \leq n$). O sistema é considerado sobre-determinado caso o número de condições l seja maior que o número de variáveis de projeto n ($l > n$). Se for possível remover as restrições redundantes, de forma atender à condição $l \leq n$, o mesmo será considerado válido. Em caso contrário, esse será considerado inconsistente e o problema precisa ser reformulado.
- O número de restrições de desigualdade $g(\mathbf{x})$ deve ser menor ou igual ao número de variáveis de projeto ($m \leq n$).

- Um modelo pode não apresentar nenhuma restrição, $l = 0$ e $m = 0$, sendo classificando como um modelo irrestrito. Caso apresente restrições esse é classificado como um modelo restrito.

- Se todas funções $f(\mathbf{x})$, $h_j(\mathbf{x})$ e $g_i(\mathbf{x})$ forem lineares, então o problema de otimização pode ser resolvido utilizando um método de Programação Linear. No caso da existência de uma ou mais funções não lineares, esse deve ser resolvido utilizando um método de Programação Não Linear.

- A função objetivo $f(\mathbf{x})$ pode ser multiplicada por uma constante positiva sem modificar a solução ótima. O mesmo é válido se for adicionada uma constante ao modelo. As condições de desigualdade podem ser multiplicadas por constantes positivas e as condições de igualdade por qualquer constante, sem afetar a região viável e a solução ótima. Entretanto, essas transformações influenciam os multiplicadores de Lagrange[1] e o desempenho dos métodos numéricos utilizados para a solução do problema.

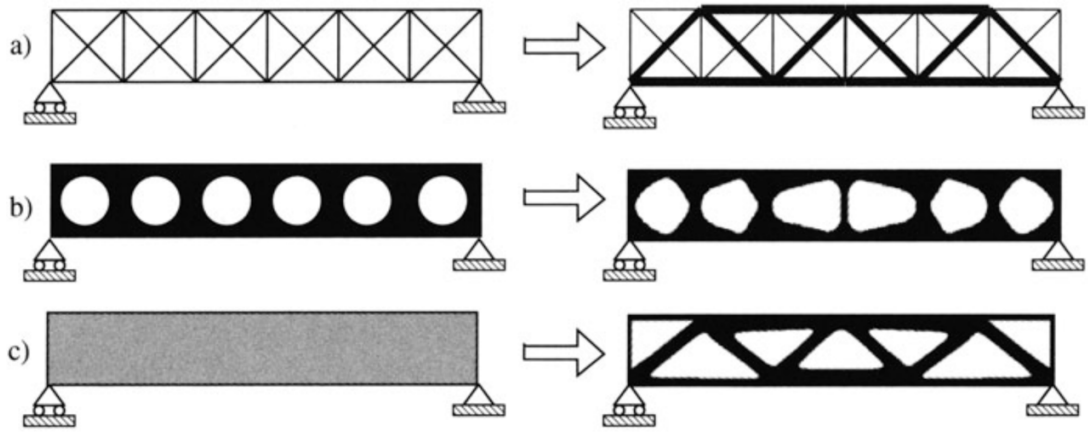
2.2 DEFINIÇÃO DE UM PROBLEMA DE OTIMIZAÇÃO TOPOLÓGICA

A otimização estrutural pode ser classificada em três grupos: paramétrica, de forma (geométrica) e topológica (BENDSØE; SIGMUND, 2004). Em um problema de otimização paramétrica procura-se encontrar as dimensões ótimas para uma estrutura. Na Figura 1a tem-se um exemplo de otimização paramétrica, onde pode ser observado um aumento nas dimensões de uma estrutura de treliça. Em uma otimização de forma ou geométrica, o contorno da estrutura é modificado, porém sua topologia é conservada. Como pode ser observado na Figura 1b, tem-se uma mudança na forma dos vazios internos da estrutura, porém a quantidade de vazios é mantida. Por fim, em uma Otimização Topológica, a estrutura sofre mudanças em sua topologia[2], ou seja, a quantidade, posição, conectividade e o formato dos vazios são alterados, conforme pode ser observado na Figura 1c.

[1] Técnica utilizada para encontrar valores mínimos ou máximos de uma função $f(\mathbf{x})$ aplicada sob uma restrição $h(\mathbf{x}) = 0$.

[2] Estudo das propriedades de uma geometria que são preservadas em transformações continuas, como por exemplo dobramento, torção e encolhimento. Ações como corte, perfuração e rasgamento são consideradas transformações descontínuas.

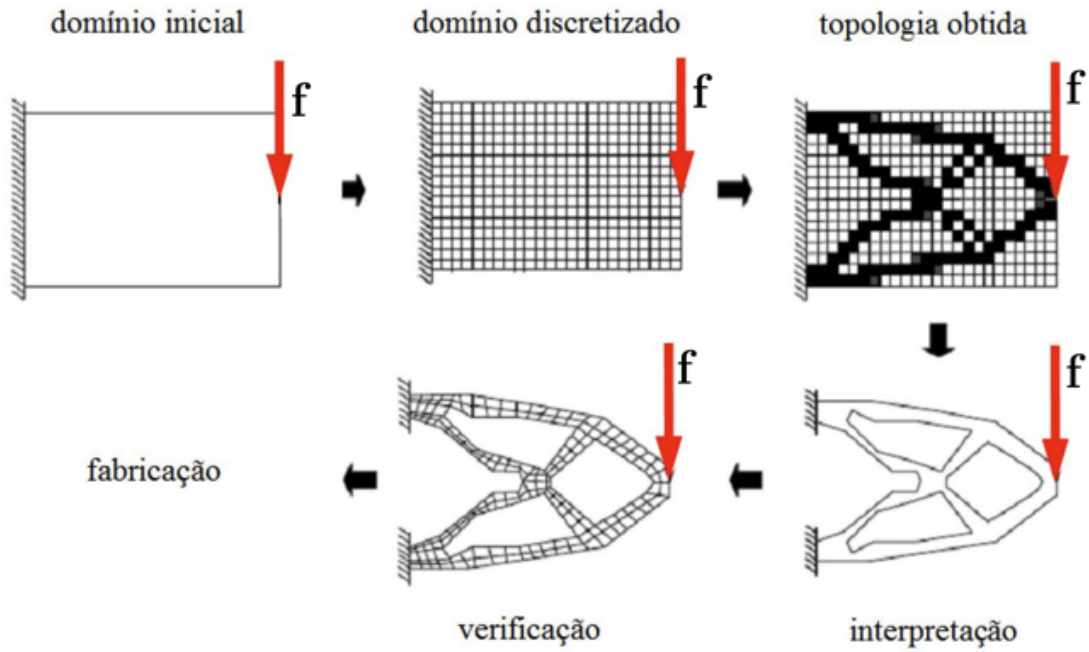
Figura 1 – Exemplos de otimizações estruturais



Fonte: Bendsøe e Sigmund (2004).

A solução de um problema de OT objetiva encontrar um valor de rigidez ótimo, diminuindo a flexibilidade do sistema. Uma das abordagens mais utilizadas para a solução de um problema de OT é a utilização do Método dos Elementos Finitos (MEF). Neste, o domínio inicial é dividindo-as em partes menores, chamadas de elementos, os quais representam o domínio contínuo do problema. Assim, as cargas de tensão, deformação e deslocamento são calculadas a partir do comportamento de cada um dos elementos (BENDSØE; SIGMUND, 2004). A discretização do domínio e demais etapas da OT são ilustradas na Figura 2.

Figura 2 – Etapas da Otimização Topológica



Fonte: Adaptado de Perini (2013).

Pode-se formalizar a solução de um problema de OT a partir da minimização da função de flexibilidade $f(\mathbf{x}, \mathbf{u})$, restrita às condições de equilíbrio $h(\mathbf{x}, \mathbf{u})$ e de volume $g(\mathbf{x}, \mathbf{u})$. Essas possuem como parâmetros, o conjunto de tensores de rigidez $\mathbf{x}$. Desta forma, o problema de OT é dado por meio da equação

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}, \mathbf{u}) = \mathbf{f}^{\mathrm{T}}\mathbf{u} \\ \text{sujeito a} \quad & h(\mathbf{x}, \mathbf{u}) \\ & g(\mathbf{x}, \mathbf{u}) \leq V \end{aligned} \tag{2.4}$$

onde $\mathbf{x}$ é o vetor de tensores de rigidez, $\mathbf{u}$ é o vetor de deslocamentos e $\mathbf{f}$ é o vetor de forças aplicadas. A condição de volume $g(\mathbf{x}, \mathbf{u})$ é posta pelo limite fixo V.

A condição de equilíbrio $h(\mathbf{x}, \mathbf{u})$ é definida como

$$h(\mathbf{x}, \mathbf{u}) = \mathbf{K}(x_e)\mathbf{u} = \mathbf{f}$$

sendo x_e um tensor de rigidez, onde $x_e \in \mathbf{x}$, e $\mathbf{K}$ é a matriz de rigidez, a qual pode ser representada por meio de

$$\mathbf{K} = \sum_{e=1}^{N} \mathbf{K}_e(x_e), \tag{2.5}$$

onde em um sistema de N elementos, cada elemento é representado por índice e com $e = 1, ..., N$.

Em uma distribuição ótima de um material isotrópico[3] em um espaço de domínio Ω, os tensores de rigidez x_e são definidos como

$$x_e = 1_\Omega x_0 \tag{2.6}$$

onde x_0 representa as propriedades base do material. Define-se Ω^{mat} como o subdomínio formado pela presença de material e Ω^{vaz} como o subdomínio formado por vazios. Portanto, considerando um ponto ω, tem-se que

$$1_\Omega = \begin{cases} 1 & \text{se } \omega \in \Omega^{\text{mat}}, \\ 0 & \text{se } \omega \in \Omega^{\text{vaz}}. \end{cases} \tag{2.7}$$

O cálculo da restrição de volume $g(\mathbf{x}, \mathbf{u})$ pode ser realizado por meio de

$$g(\mathbf{x}, \mathbf{u}) = \int_\Omega 1_\Omega \mathrm{d}\Omega = \text{Vol}(\Omega^{\text{mat}}) \tag{2.8}$$

[3] Material onde as propriedades mecânicas são as mesmas em todas direções.

2.3 SOLUÇÃO DE PROBLEMAS DE OTIMIZAÇÃO TOPOLÓGICA

Diversos métodos podem ser utilizados para a solução de um problema Otimização Topológica, sendo que entre eles destacam-se os métodos: Material Isotrópico Sólido com Penalização (SIMP - *Solid Isotropic Material with Penalization*), Método de Limites Variáveis (LSM - *Level-Set Method*) e Otimização Evolutiva (OE).

2.3.1 Material Isotrópico Sólido com Penalização

Um modelo de OT apresenta uma distribuição discreta com a presença ou não de material em um ponto ω (Equação 2.7). No entanto trabalhar com variáveis discretas diminui a quantidade de soluções possíveis. Um dos métodos mais utilizados para resolver esse problema, consiste em substituir o uso de variáveis discretas por valores contínuos, representando a densidade dos elementos. A fim de converter os valores contínuos para valores discretos (0 ou 1), é introduzido um formato de penalização. O método SIMP é o método mais utilizado, dentre os métodos que utilizam os conceitos de densidade variável e penalização (BENDSØE; SIGMUND, 2004).

O método SIMP foi proposto por Bendsøe em 1989, e baseia-se na definição de um material artificial, com uma densidade variável ρ, onde $\rho = 0$ corresponde ao vazio e $\rho = 1$ a presença total de material. Assim, a Equação 2.6 pode ser redefinida por meio de uma função de densidade $\rho(\omega)$, como

$$x_e(\omega) = \rho(\omega)^t x_0 \tag{2.9}$$

onde t representa um parâmetro de penalização ($t > 1$) e x_o representa as propriedades base do material, como por exemplo o módulo de Young e o coeficiente de Poisson (BENDSØE; SIGMUND, 2004). O parâmetro de penalização t é utilizado para converter os valores intermediários da função de densidade ρ, para valores discretos, evitando densidades intermediárias na solução. Segundo Perini (2013), materiais com densidade intermediária apresentam baixa contribuição estrutural para o modelo. A Figura 3 ilustra diferença da não utilização ($t = 1$) e a utilização de penalização ($t = 3$).

A função objetivo $f(\mathbf{x}, \mathbf{u})$ (Equação 2.4) pode ser reescrita considerando os elementos com densidades variáveis e penalização por meio de

$$f(\mathbf{x}, \mathbf{u}) = \sum_{e=1}^{N} \rho_e^t \mathbf{f}_e^{\mathrm{T}} \mathbf{u}_e \tag{2.10}$$

onde $\mathbf{f}_e$ e $\mathbf{u}_e$ são, respectivamente, os vetores de força e de deslocamentos que são aplicados a um elemento e.

Figura 3 – Comparação entre uma otimização gerada sem e com penalização

Fonte: O Autor (2023).

A Equação 2.8, que representa o cálculo da restrição de volume $g(\mathbf{x}, \mathbf{u})$, pode ser reescrita considerando a função $\rho(\omega)$ a partir da Equação 2.11, onde $\omega \in \Omega$ e $0 \leq \rho_{\min} \leq \rho(\omega) \leq 1$. Neste caso é introduzido um limite de densidade miníma $\rho_{\min}$ a fim de evitar possíveis problemas de singularidade na solução. Considera-se problemas de singularidade, quando elementos apresentam densidades que tendem a zero, causando tensões que extrapolam a restrição de tensionamento. Em aplicações normais utiliza-se um valor de $\rho_{\min} = 10^{-3}$ (BENDSØE; SIGMUND, 2004).

$$g(\mathbf{x}, \mathbf{u}) = \int_{\Omega} \rho(\omega) \mathrm{d}\Omega \leq V \tag{2.11}$$

O método SIMP é um método iterativo, onde a cada iteração as densidades são atualizadas para cada elemento, independentemente dos demais elementos. Após a distribuição do material e o cálculo das tensões, é realizado o cálculo da deformação do modelo. O processo iterativo é interrompido quando as condições necessárias da otimização foram atingidas. Diferentes condições podem ser utilizadas, como por exemplo, as mudanças mínimas nos volumes e na deformação da estrutura.

A variável de projeto relacionada a densidade ρ_e de cada elemento é atualizada por meio da Equação 2.12, onde ρ_e^{novo} é o valor de densidade atualizado, ρ_e o valor atual da densidade e B_e é o valor da condição de otimização. Os parâmetros ζ e η controlam as mudanças realizadas na atualização, podendo ser ajustados para uma melhor eficiência do método (BENDSØE; SIGMUND, 2004). O valor de ζ define um limite máximo na mudança das densidade e o coeficiente η é utilizado para uma diminuição das oscilação nos valores de B_e. Segundo Bendsøe e Sigmund (2004), os valores típicos para ζ e η são $0{,}2$ e $0{,}5$, respectivamente.

$$\rho_e^{\text{novo}} = \begin{cases} \max\{\rho_e - \zeta, \rho_{\min}\} & \text{se } \rho_e B_e^{\eta} \leq \max\{\rho_e - \zeta, \rho_{\min}\}, \\ \rho_e B_e^{\eta} & \text{se } \min\{\rho_e + \zeta, 1\} < \rho_e B_e^{\eta} < \max\{\rho_e - \zeta, \rho_{\min}\}, \\ \min\{\rho_e + \zeta, 1\} & \text{se } \min\{\rho_e + \zeta, 1\} \leq \rho_e B_e^{\eta} \end{cases} \tag{2.12}$$

O valor da condição de otimização B_e pode ser obtido por meio da Equação 2.13. Nesta, o multiplicador de Lagrange Λ_e é obtido por meio de um algoritmo de busca binária, o qual assumem-se dois valores arbitrários como limite inferior e superior, e iterativamente busca-se definir um valor que mantenha o volume da estrutura dentro das restrições volumétricas.

$$B_e = \frac{t\rho_e^{t-1}\mathbf{f}^{\mathrm{T}}\mathbf{u}}{\Lambda_e} \tag{2.13}$$

A implementação direta do método SIMP ocasiona a instabilidade de tabuleiro na geometria, gerando uma topologia final de falsa rigidez. Esse efeito é assim chamado, devido a similaridade da geometria resultante com um tabuleiro de xadrez (PERINI, 2013). A Figura 4 ilustra tal instabilidade.

Figura 4 – Exemplo de geometria com instabilidade de tabuleiro

Fonte: Adaptado de Sigmund e Petersson (1998)

A fim de mitigar a ocorrência da instabilidade de tabuleiro, é introduzido na solução um mecanismo de filtragem das densidades, descrito por meio da Equação 2.14, onde $\tilde{\rho_e}$ é a densidade do elemento e após o procedimento de filtragem. Neste processo, utilizam-se as densidades ρ_i para cada elemento i, onde $\{i \in N | \text{dist}(e,i) \leq r_{\min}\}$. O valor de $r_{\min}$ corresponde ao raio de filtragem mínimo e dist(e,i) é a distância dos elementos e e i.

$$\tilde{\rho_e} = \frac{1}{\rho_e \sum_{i=1}^{N} \hat{W}_i} \sum_{i=1}^{N} \hat{W}_i \, \rho_i \tag{2.14}$$

Define-se o fator de ponderação $\hat{W}$ como

$$\hat{W}_i = r_{\min} - \text{dist}(e,i),$$

Existem diversos *softwares* que implementam o método SIMP. Entre esses softwares destacam-se Solidworks, ANSYS e ABAQUS (TYFLOPOULOS; HASKINS; STEINERT, 2021). Além disso, considerações sobre a implementação do método SIMP podem ser encontradas nos trabalhos de Aage *et al.* (2012), Sigmund (2001), Andreassen *et al.* (2010) e Aranda, Bellido e Donoso (2020).

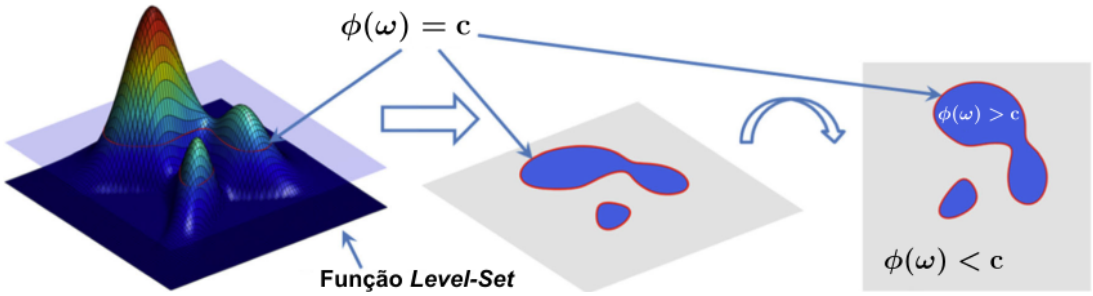
2.3.2 Método de Limites Variáveis

Os métodos baseados em densidade representam as bordas da estrutura como elementos de densidades intermediárias, comprometendo a definição da estrutura. A fim de evitar esse problema, o *Level-Set Method* (LSM) considera como variáveis de projeto os limites da estrutura, ao invés das densidades dos elementos. A distribuição de um ponto ω nos possíveis subdomínios é definida por meio de de uma função $\phi(\omega)$, que é comparada a um nível constante c, determinando a presença ou não de material (DIJK *et al.*, 2013). Ou seja, a distribuição do material é realizada por meio de

$$\begin{cases} \phi(\omega) > c \Leftrightarrow & \omega \in \Omega^{\text{mat}} \\ \phi(\omega) = c \Leftrightarrow & \omega \in \Gamma \\ \phi(\omega) < c \Leftrightarrow & \omega \in \Omega^{\text{vaz}} \end{cases}$$

onde o domínio Ω^{mat} representa a presença de material, Ω^{vaz} representa o vazio, Γ os limites e c é uma constante, usualmente é definida como 0. A Figura 5 ilustra um exemplo de uma função *Level-Set*.

Figura 5 – Exemplo de uma função *Level-Set*

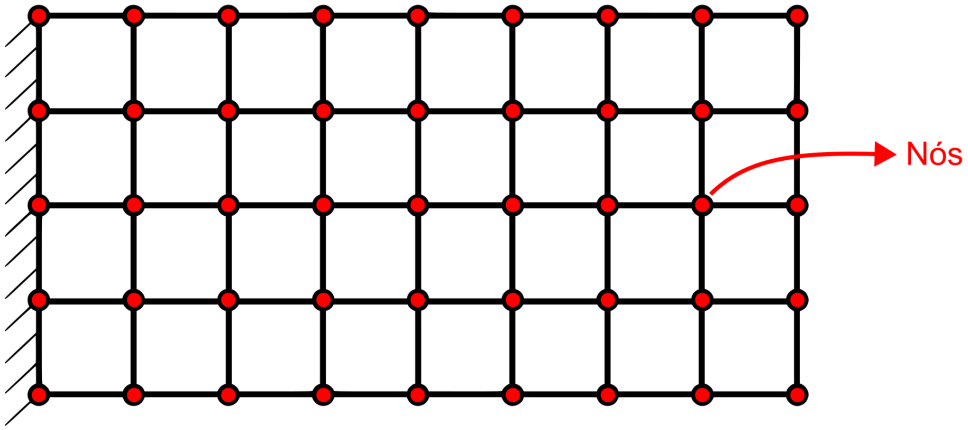


Fonte: Adaptado de Wei *et al.* (2018).

A função $\phi(\omega)$ representa a evolução da estrutura e é composta por um conjunto de funções base, podendo ser de diferentes tipos. Na implementação do LSM, usualmente utiliza-se as funções base radiais (RBF - *Radial Basis Function*) (DIJK *et al.*, 2013; WEI *et al.*, 2018). As RBF são um conjunto de funções simétricas, podendo ser geradas considerando a distância em relação a origem/centro. Diferentes tipos de funções podem ser utilizadas para a definição de uma RBF, sendo que as funções do tipo Multi Quadrática Inversa (MQI) são as mais adequadas para trabalhar no domínio matemático positivo do problema (WANG, 2006).

Para a geração das RBF, utiliza-se como origem a coordenada ω de cada nó. O nó é o ponto intermediário entre elementos de um domínio discretizado, conforme ilustrado na Figura 6. A função base r_i, para cada nó i, é definida a partir da Equação 2.15, onde ω_i corresponde a coordenada do nó i e s é o parâmetro constante de forma, que usualmente assume um valor igual a 1.

Figura 6 – Representação dos nós em um domínio discretizado



Fonte: Adaptado de Wang (2006).

$$r_i(\omega) = \frac{1}{\sqrt{(\omega - \omega_i)^2 + s^2}} \tag{2.15}$$

Define-se a função $\phi(\omega)$ como uma interpolação das funções base $r(\omega)$, onde ψ é o coeficiente de expansão da função, que é calculado por meio da Equação 2.16.

$$\phi(\omega) = \sum_{i=1}^{N} \psi r_i(\omega) \tag{2.16}$$

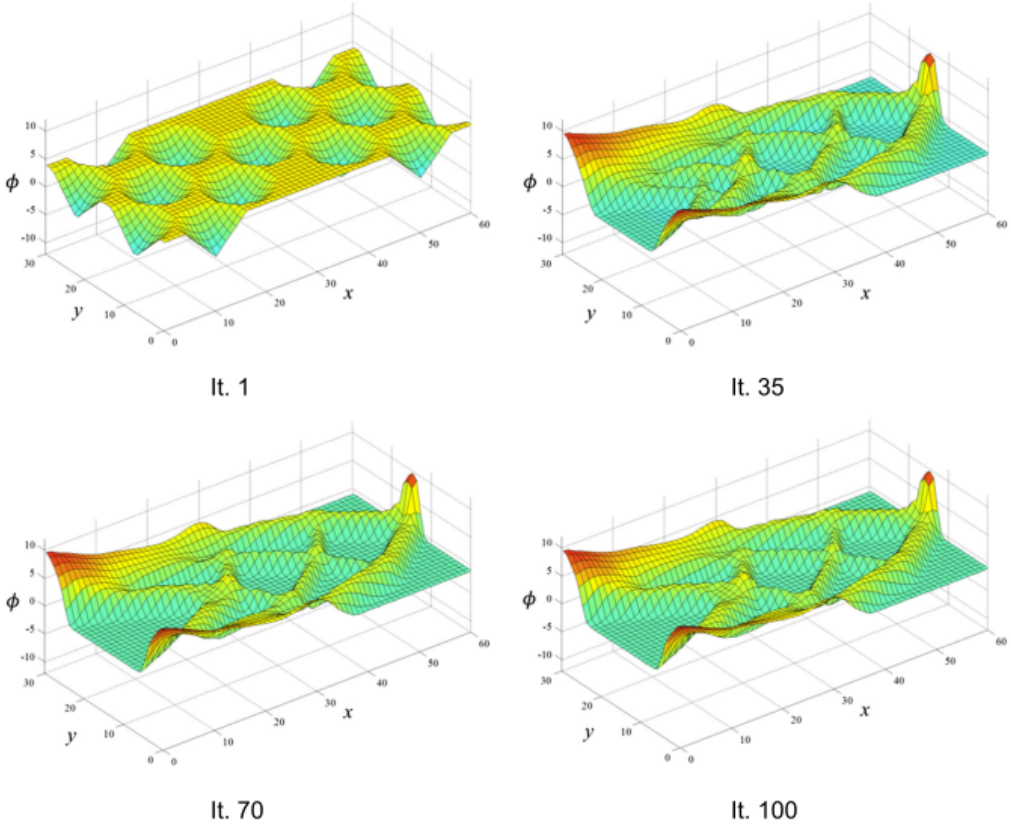
A fim de representar a evolução do modelo, representada na Figura 7, a Equação 2.16 pode ser reescrita por meio da Equação 2.17, onde τ é o pseudo-tempo representando a direção de evolução, R é uma matriz $[N \times N]$ com a funções base e $\Psi(\tau)$ é o vetor de coeficientes de expansão (Equação 2.19).

$$\phi(\tau) = \boldsymbol{R}\Psi(\tau) \tag{2.17}$$

$$\boldsymbol{R} = \begin{bmatrix} r_1(\omega_1) & \cdots & r_N(\omega_1) \\ \vdots & \ddots & \vdots \\ r_1(\omega_N) & \cdots & r_N(\omega_N) \end{bmatrix} \tag{2.18}$$

$$\Psi(\tau) = \{\psi_1(\tau) \cdots \psi_N(\tau)\} \tag{2.19}$$

Figura 7 – Evolução de uma função *Level-Set*



Fonte: Adaptado de Wei *et al.* (2018).

O método LSM utiliza um processo de otimização iterativo. A cada iteração, o conjunto de coeficientes Ψ é atualizado por meio da Equação 2.20, onde $\hat{\boldsymbol{B}}$ é o vetor de velocidades e $\Delta\tau$ é o passo de tempo, que usualmente recebe um valor de $0{,}5$. As tensões nos elementos são calculadas utilizando o MEF.

$$\Psi(\tau_{i+1}) = \Psi(\tau) + \Delta\tau \boldsymbol{R}^{-1}\hat{\boldsymbol{B}} \tag{2.20}$$

O vetor de velocidades $\hat{\boldsymbol{B}}$ é calculado por meio da Equação 2.21, onde ε é a tensão dos elementos, Λ é o multiplicador de Lagrange obtido a partir de uma iteração interna de regulação e $\delta(\phi)$ é o esquema de controle da evolução.

$$\hat{\boldsymbol{B}} = (\varepsilon - \Lambda)\delta(\phi) \tag{2.21}$$

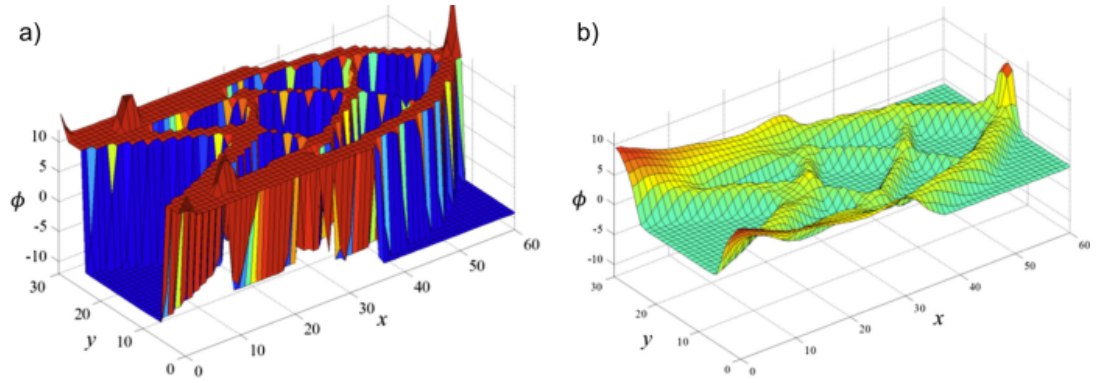
O controle da evolução $\delta(\phi)$ é definido por meio de Equação 2.22. A constante Δ é utilizada como balizador para a verificação do nível da função ϕ, e usualmente é utilizado um valor $\Delta > 5$.

$$\delta(\phi) = \begin{cases} \frac{3}{4\Delta}\left(1 - \frac{\phi^2}{\Delta^2}\right), & \text{caso } -\Delta < \phi < \Delta, \\ 0, & \text{caso contrário.} \end{cases} \tag{2.22}$$

A utilização de funções do tipo RBF podem gerar problemas de convergência, uma vez que elas podem apresentar valores altos. Em Wei *et al.* (2018) é apresentado um processo de reinicialização dos coeficientes de expansão Ψ^u, sendo realizado a cada iteração. Para tanto, é utilizada uma aproximação a partir da média dos gradientes dos elementos que compõem os limites Γ. Na Equação 2.23 é apresentada a atualização do coeficiente de expansão Ψ^u, onde $|\nabla\phi_\gamma^c|$ é a norma do gradiente do γ-ésimo elemento. Na Figura 8a é representada uma função sem a reinicialização dos coeficientes de expansão, já na Figura 8b é representada a função com a reinicialização dos coeficientes.

$$\Psi^u = \frac{\Psi}{\text{média}(|\nabla\phi_1^c|, \cdots, |\nabla\phi_\gamma^c|)} \tag{2.23}$$

Figura 8 – Resultado da reinicialização de coeficientes de expansão



Fonte: Adaptado de Wei *et al.* (2018).

No método LSM tem-se como restrição de otimização, o volume máximo da estrutura. Neste o cálculo do volume é realizado por meio de

$$\text{Vol}(\phi) = \int H(\phi)\text{d}\Omega - \left[\text{Vol}_0 - (\text{Vol}_0 - \text{Vol}_{\max})\frac{i}{n_R}\right]$$

onde $H(\phi)$ é a função degrau (Equação 2.24), Vol_0 é o volume inicial da estrutura, $\text{Vol}_{\max}$ é o volume máximo para a estrutura, i é a iteração atual e n_R é o número máximo de iterações.

$$H(\phi) = \begin{cases} 0 \text{ para } \phi < 0 \\ 1 \text{ para } \phi \geq 0. \end{cases} \tag{2.24}$$

Como critério de convergência, utiliza-se a variação do volume ou a deformação da estrutura. Por exemplo, considera-se que o método convergiu quando a variação de volume ou deformação é menor que $0{,}1\%$.

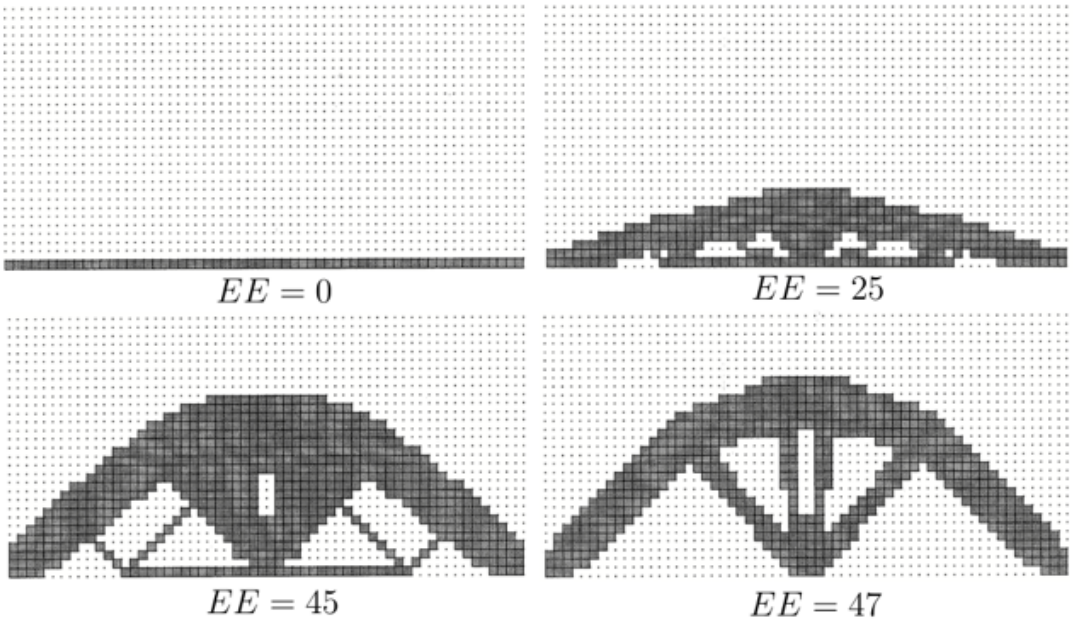
O método *Level-Set Method* não é usualmente utilizado em *softwares* comerciais, sendo apresentado majoritariamente em trabalhos acadêmicos, como por exemplo, os trabalhos desenvolvidos por Dunning e Kim (2014), Wei *et al.* (2018) e Andreasen, Elingaard e Aage (2020).

2.3.3 Métodos Evolutivos

A Otimização Evolutiva consiste em um conjunto de abordagens para a busca de uma topologia ótima utilizando critérios heurísticos (DEATON; GRANDHI, 2013). O desenvolvimento dos métodos evolutivos tiveram início com o trabalho de Xie e Steven (1993), onde foi proposto o método *Evolutionary Structural Optimization* (ESO).

O método ESO utiliza uma abordagem que consiste na remoção do material em regiões onde os elementos apresentam baixa tensão estrutural. Devido a baixa convergência do método, diferentes abordagens podem ser utilizadas para melhorar a busca por uma solução ótima (MUNK, 2019). Em Querin, Steven e Xie (1998) é proposto o método *Bidirectional Evolutionary Structural Optimization* (BESO), onde não é realizada somente a remoção do material em regiões de baixa tensão, mas também a adição de material nas regiões com tensão elevada. A Figura 9 ilustra uma otimização utilizando o método BESO.

Figura 9 – Exemplo de uma otimização realizada com o método BESO



Fonte: Adaptado de Querin, Steven e Xie (1998).

O BESO utiliza uma abordagem iterativa onde, a cada passo, os elementos são removidos ou adicionados ao sistema. A remoção ou inserção desses é realizada por meio das Equações 2.25 e 2.26, onde um elemento é removido caso a Equação 2.25 seja verdadeira ou inserido caso a Equação 2.26 seja verdadeira. O valor de $\sigma_{\max}$ corresponde a tensão máxima do sistema e FR e FI correspondem aos fatores de remoção e de inserção, respectivamente. A tensão de cada elemento e a tensão máxima da estrutura são recalculadas a cada iteração.

$$\sigma_e \leq FR \cdot \sigma_{\max} \tag{2.25}$$

$$\sigma_e \geq FI \cdot \sigma_{\max} \tag{2.26}$$

O fator de remoção FR e o fator de inserção FI são obtidos por meio das Equações 2.27 e 2.28, respectivamente.

Na Equação 2.27, r_{EE} corresponde a uma constante, comumente definida como 10^{-3}, e r_{NO} a uma constante para a variável de oscilação NO, usualmente definida como 10^{-2}. Os valores das constantes i_{EE} e i_{NO} existentes na Equação 2.28 são regularmente definidas, respectivamente, como 10^{-2} e como 10^{-1}.

Por fim, os valores de EE e NO correspondem a estado estável e o estado de oscilação.

$$FR = \mathrm{r}_{EE} \cdot EE + \mathrm{r}_{NO} \cdot NO \quad \text{onde } 0 \leq FR \leq 1 \tag{2.27}$$

$$FI = 1 - \mathrm{i}_{EE} \cdot EE - \mathrm{i}_{NO} \cdot NO \quad \text{onde } 0 \leq FI \leq 1 \tag{2.28}$$

O estado de oscilação NO é definido quando um elemento é adicionado ao sistema em uma iteração e removido na iteração seguinte, causando um número infinito de iterações. O estado de oscilação é verificado quando uma das Equações 2.25 e 2.26 são satisfeitas para um determinado elemento. Neste caso, o valor de NO é incrementado em 1 e as Equações 2.27 e 2.28 são recalculadas, permitindo que o sistema saia do estado oscilatório e continue evoluindo.

O valor EE corresponde a contagem de iterações que se mantêm em estado estável. Considera-se como estado estável quando nenhum elemento satisfaz as Equações 2.25 e 2.26. Ao final de cada iteração, verifica-se se a convergência do método foi realizada. Caso o sistema não tenha convergido, o valor da variável EE é incrementada em 1 e as Equações 2.27 e 2.28 são recalculadas para todos os elementos do sistema.

Segundo Querin, Steven e Xie (1998), pode-se definir como critério de convergência a minimização do índice de performance (IP), que pode ser calculado por meio da equação

$$f(\mathbf{x}) = IP = \frac{\sum \sigma_e V_e}{\mathbf{f} L} \tag{2.29}$$

onde σ_e e V_e correspondem a tensão e ao volume de elemento e, respectivamente. A força aplicada no sistema é definido por $\mathbf{f}$ e L representa a dimensão de referência.

A função objetivo f é condicionada as duas condições de desigualdade

$$\begin{aligned} X_e \cdot \sigma_e - FR \cdot \sigma_{\max} \geq 0 \\ FI \cdot \sigma_{\max} - \gamma_e \cdot \sigma_e \geq 0 \end{aligned} \tag{2.30}$$

onde X_e é conjunto de possíveis valores discretos, podendo assumir o valor de 0 ou 1 e γ_e é o multiplicador de inserção, onde $\gamma_e \in \{0, b\}$. O valor de b é calculado a partir de

$$b = 1 - \frac{\mathbf{q_a}}{\mathbf{q_p} + \mathbf{q_a}} \tag{2.31}$$

onde $\mathbf{q_a}$ é o número de elementos ligados a e, incluso, antes da validação das Equações 2.25 e 2.26. O valor de $\mathbf{q_p}$ corresponde ao número de elementos ligados a e, caso algum dos elementos seja removido ou adicionado.

Deaton e Grandhi (2013) afirma que os métodos OE não são amplamente utilizados no desenvolvimento de *softwares* comerciais.

3 IMPLEMENTAÇÃO DESENVOLVIDA

Neste capítulo é apresentada a solução desenvolvida neste trabalho. Os requisitos são apresentados na Seção 3.2. Esses, foram definidos a partir da análise de ferramentas equivalentes já existentes, apresentadas na Seção 3.1. Os detalhes de implementação são apresentados na Seção 3.3. Por fim, nas Seções 3.4 e 3.5 tem-se a apresentação dos testes realizados e os resultados obtidos.

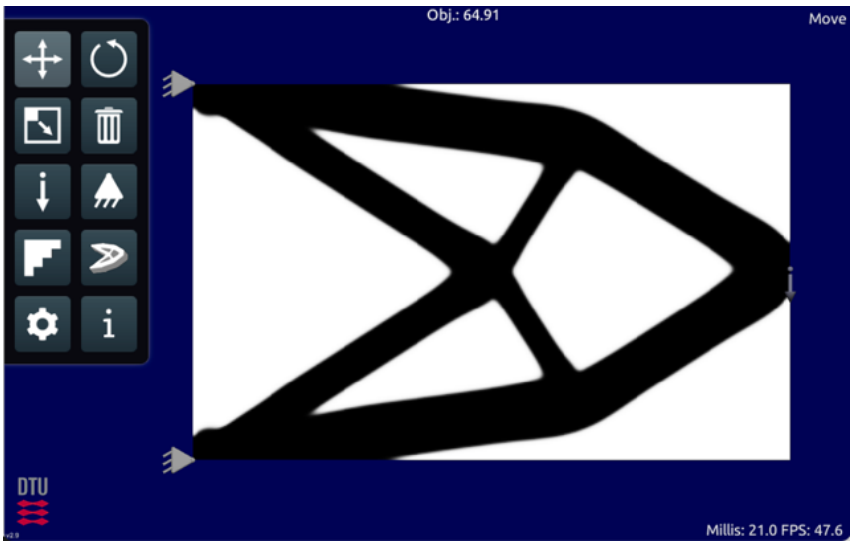
3.1 ANÁLISE DE FERRAMENTAS SIMILARES

Já existem algumas ferramentas que auxiliam na compreensão e na visualização dos conceitos dos métodos de Otimização Topológica. Dentre essas, destacam-se três aplicações gratuitas, desenvolvidas pelo Departamento de Engenharia Mecânica da Universidade Técnica da Dinamarca: *TopOpt*, *TopOpt 3D* e *TopOpt Shape*. Nesta seção é realizada uma análise das funcionalidades dessas ferramentas, visto que elas foram utilizadas como base para a definição dos requisitos da implementação desenvolvida neste trabalho.

3.1.1 *TopOpt*

O *TopOpt* é uma aplicação desenvolvida por Aage *et al.* (2012) com o objetivo de disponibilizar uma ferramenta gratuita para o ensino de OT. Essa foi desenvolvida na linguagem de programação C#, utilizando o *framework Unity3D*, e encontra-se disponível para o uso em computadores pessoais. Na Figura 10 é possível visualizar a tela inicial do *TopOpt*.

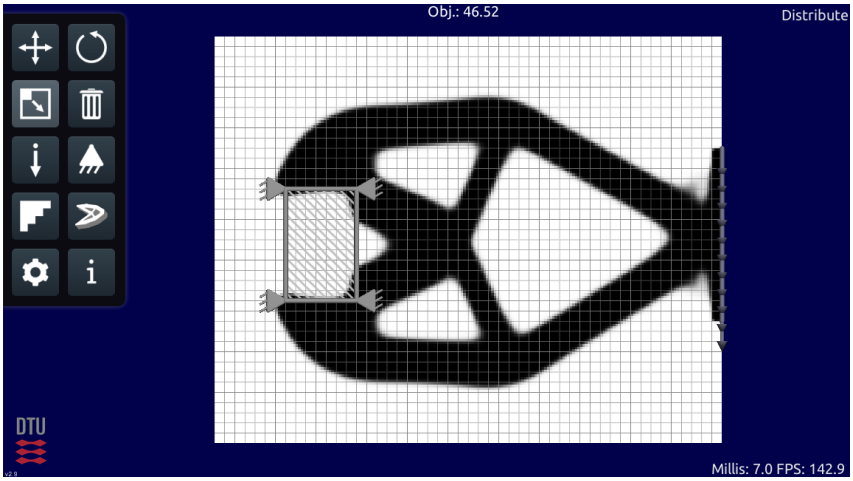
Figura 10 – Tela inicial do *TopOpt*



Fonte: O Autor (2023).

O *TopOpt* possibilita a otimização de estruturas bidimensionais e, para tanto, utiliza o método SIMP. O processo de otimização pode ser acompanhado em tempo real, sendo possível visualizar a evolução da estrutura em função do número de iterações. Essa permite adicionar, remover ou editar os parâmetros da simulação, como por exemplo, as condições de contorno. Na Figura 11 tem-se um exemplo de otimização onde os suportes fixos possuem um formato bidimensional e as forças são aplicadas de forma linear.

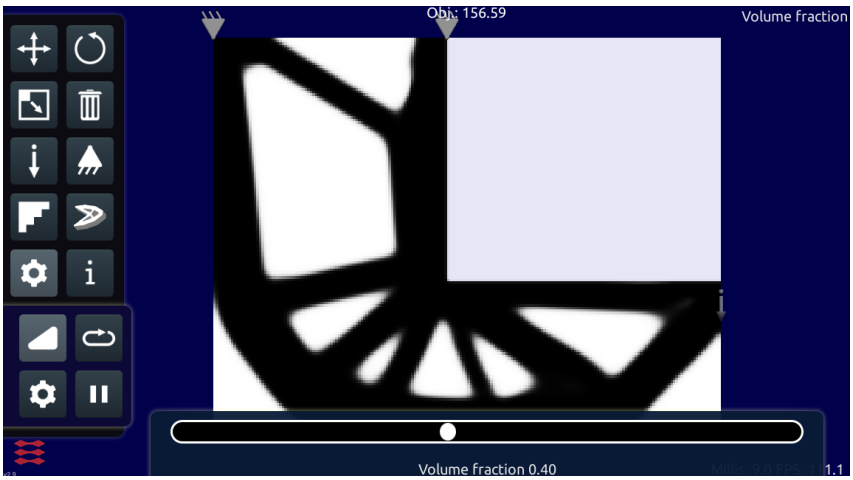
Figura 11 – Redimensionamento de parâmetros na aplicação *TopOpt*



Fonte: O Autor (2023).

As funcionalidades de reinicialização, exportação dos resultados, entre outras, podem ser acessadas por meio de submenus. Para a definição do volume máximo da estrutura, é possível selecionar o valor desejado por meio de uma barra existente na parte inferior da tela, conforme pode ser observado na Figura 12. Na parte superior é possível observar o valor atual da função objetivo, possibilitando a visualização do processo de minimização. Em relação a interface, destaca-se a falta de informações sobre as funcionalidades dos botões.

Figura 12 – Edição de volume da estrutura na aplicação *TopOpt*

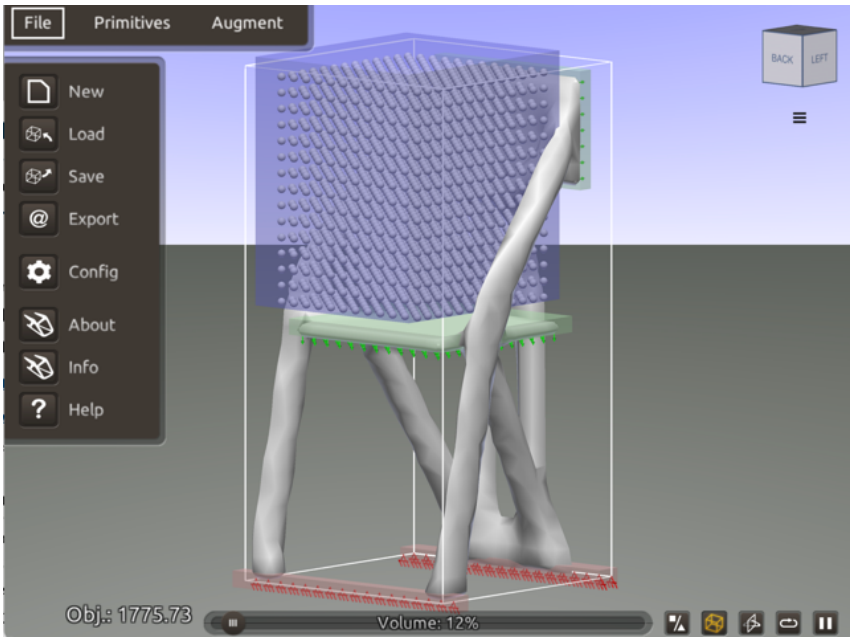


Fonte: O Autor (2023).

3.1.2 *TopOpt 3D*

Desenvolvido como uma evolução para o *TopOpt*, esse foi projetado inicialmente para dispositivos móveis, mas apresenta também uma versão para computadores pessoais (NOBEL-JØRGENSEN *et al.*, 2014). Diferentemente do *TopOpt*, esse possibilita a otimização e visualização de estruturas tridimensionais. Para a realização do processo de otimização é utilizado o método SIMP. Na Figura 13 é possível visualizar o processo de Otimização Topológica de uma cadeira.

Figura 13 – Exemplo de um problema na aplicação *TopOpt 3D*



Fonte: O Autor (2023).

O processo de otimização pode ser acompanhado em tempo real, com a possibilidade de uma visualização utilizando uma projeção ortográfica ou em perspectiva. Esse permite informar ainda os parâmetros de otimização de forma interativa. Por fim, o *TopOpt 3D* permite exportar a estrutura resultante como um arquivo tridimensional, possibilitando o uso da estrutura em aplicações externas. Na Figura 14 é possível observar a estrutura da cadeira otimizada em um *software* de visualização 3D.

Figura 14 – Visualização da estrutura ótima para uma cadeira

Fonte: O Autor (2023).

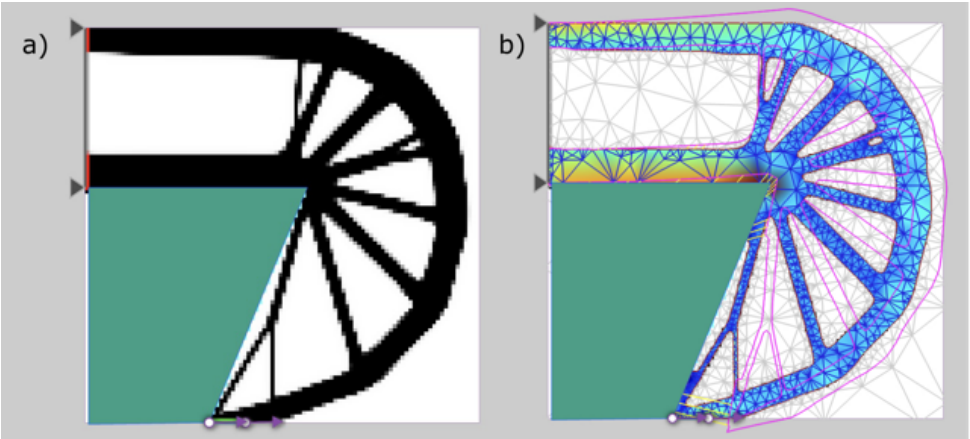
Dentre as configurações disponíveis, é possível alterar o volume máximo da estrutura, reiniciar ou pausar o processo de otimização. Essas opções podem ser acessadas por meio de botões existentes na parte inferior da tela. Também no setor inferior, estão presentes funcionalidades adicionais, como por exemplo, tornar visível os contornos do domínio, espelhar a geometria, e modificar a visualização da estrutura entre cúbica, transparente e convencional.

3.1.3 *TopOpt Shape*

O *TopOpt Shape* é um aplicativo para a OT de estruturas bidimensionais, desenvolvido para dispositivos móveis por Nguyen *et al.* (2020). Diferentemente das demais ferramentas, o *TopOpt Shape* realiza o processo de otimização utilizando um método híbrido, que combina os métodos SIMP e DSC (*Deformable Simplicial Complex*). O DSC é um método de otimização de forma que utiliza uma malha triangular, possibilitando resultados mais nítidos nas bordas da estrutura. Na Figura 15 são ilustradas as duas etapas desse processo híbrido de otimização. Na Figura 15a, tem-se o resultado do método SIMP, onde pode ser visualizado um efeito quadriculado nas bordas da estrutura. Na Figura 15b tem-se o resultado após a utilização do método DSC, onde pode ser observado que esse efeito é reduzido. O motor[1] que implementa os métodos otimização foi codificado na linguagem de programação C++.

[1] Módulo do *software* em que os cálculos matemáticos são realizados.

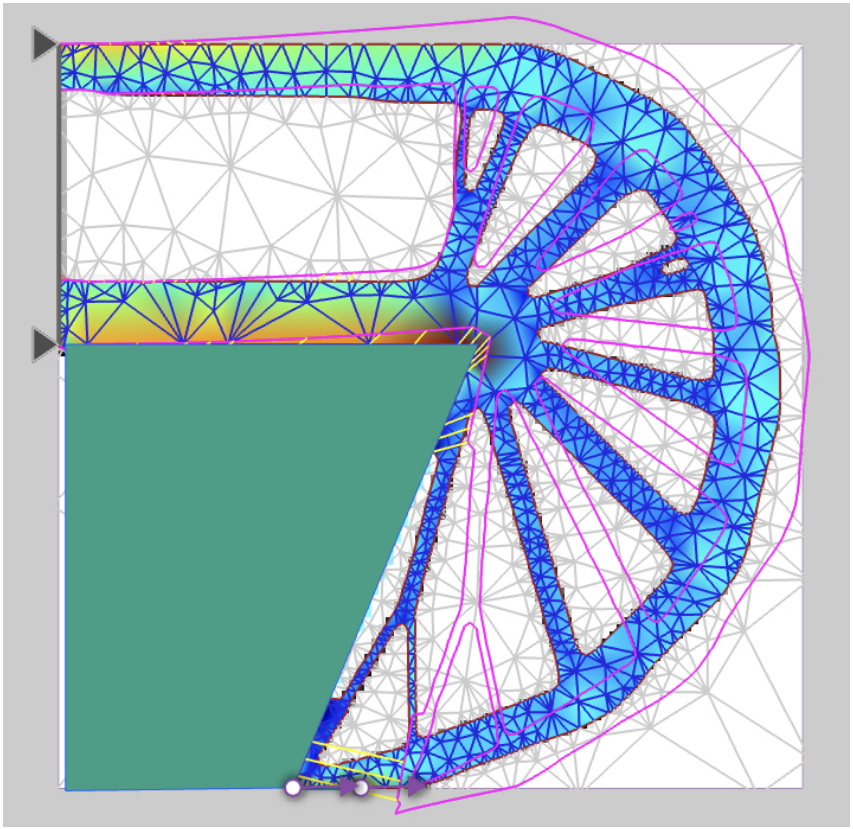
Figura 15 – Otimização realizada no *TopOpt Shape*



Fonte: O Autor (2023).

Ao utilizar o método SIMP, é possível editar alguns parâmetros, como por exemplo, os suportes, as forças aplicadas, materiais ou vazios constantes. Esses podem ser inseridos, removidos, movimentados e redimensionados. As dimensões do domínio e o volume máximo da estrutura não podem ser editados. Já os resultados da otimização DSC, podem ser visualizados com diferentes configurações, possibilitando uma visualização das tensões, arestas e deslocamentos (Figura 16). Por fim, os projetos podem ser salvos ou clonados.

Figura 16 – Estrutura ótima obtida por meio do método DSC



Fonte: O Autor (2023).

3.2 DEFINIÇÃO DOS REQUISITOS

A partir das análises dos *softwares TopOpt*, *TopOpt 3D* e *TopOpt Shape*, foram definidos os requisitos funcionais e não-funcionais da aplicação desenvolvida. Os requisitos funcionais, compõem o conjunto de funcionalidades que a aplicação deve suportar, e são:

- Otimização e visualização de estruturas bidimensionais: a possibilidade da otimização de estruturas tridimensionais é importante, no entanto a complexidade e o tempo de implementação extrapolariam o tempo limite para o desenvolvimento deste trabalho.
- Implementação do método SIMP: os resultados obtidos com uma abordagem híbrida são relevantes, no entanto a implementação do método DSC demandaria um tempo elevado.
- Possibilidade de configuração do domínio e das condições de contorno: a aplicação deve possibilitar a edição de parâmetros como, por exemplo, as cargas aplicadas à estrutura, regiões de material ou vazios e suportes fixos ou móveis.
- Otimização em tempo real: a aplicação deve apresentar o estado da estrutura otimizada a cada iteração. Dessa forma, é possível acompanhar as mudanças sofridas pela estrutura, auxiliando na compreensão do processo de OT.
- Definição das dimensões e volume do domínio: possibilidade de alterações nas dimensões do domínio, bem como o volume máximo da estrutura. Adicionalmente, deve suportar a possibilidade que sejam escolhidos diferentes coeficientes de penalização e raio de filtragem para o cálculo da densidade dos elementos.
- Definição das propriedades do material: a implementação deve possuir a possibilidade de edição das propriedades do material, como por exemplo, o módulo de Young e coeficiente de Poisson do material. Essas são responsáveis por descreverem o comportamento do material perante a aplicação de tensões. Não foi encontrada em nenhum dos *softwares* analisados a possibilidade de tal configuração.
- Visualização das variáveis de projeto: o valor da função objetivo e volume atual devem ser informados a cada iteração, tendo a evolução desses valores representada por meio de um gráfico.
- Salvar e carregar projetos: possibilidade de salvar um projeto. O usuário poderá realizar o *download* do arquivo contendo as informação do projeto, assim como o *upload* do mesmo.

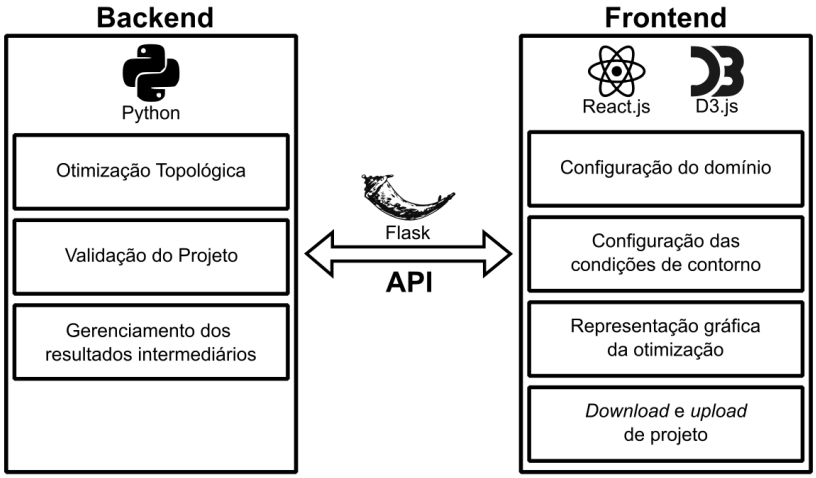
Os requisitos não funcionais compreendem os aspectos técnicos da implementação e são:

- Executar em um navegador *Web*: a implementação deverá ser disponibilizada em uma página *Web*, podendo ser utilizada por meio de um navegador. Todas aplicações analisadas necessitam da instalação, adicionando assim um processo adicional para o usuário, além de apresentar uma dependência das especificações e configurações do dispositivo do usuário.
- Interface gráfica intuitiva e simples: um fator observado é que as ferramentas analisadas não possuem uma descrição dos comandos, dificultando a utilização por usuários que não estão habituados às simbologias e nomenclaturas da OT. Desta forma, a interface da aplicação foi desenvolvida objetivando facilitar a utilização por usuários iniciantes na área.

3.3 ARQUITETURA DA APLICAÇÃO

A implementação foi desenvolvida utilizando uma arquitetura de *Backend For Frontend*. O *Backend* é responsável pela execução da OT propriamente dita e pode ser executado em um servidor com um poder computacional maior. Desta forma, o usuário pode acessar a aplicação por meio de um dispositivo com uma baixa capacidade computacional. O *Frontend* é responsável pela interação com o usuário e apresentação dos resultados, sendo executado em um navegador *Web* no dispositivo do usuário. Os resultados da otimização, realizada no *Backend*, são renderizados e mostrados no *Frontend*. Na Figura 17 tem-se uma representação da arquitetura desenvolvida. A Figura 34, presente no Apêndice A, contém o diagrama de sequência representando a execução da solução.

Figura 17 – Representação da arquitetura *Backend For Frontend*



Fonte: O Autor (2023).

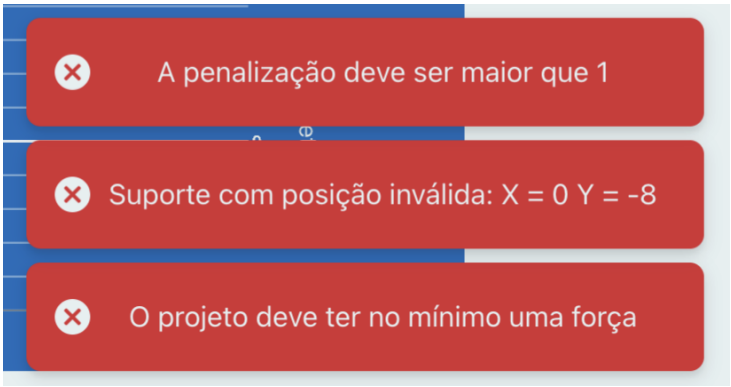
3.3.1 Implementação do *Backend*

Para o desenvolvimento do *Backend*, foi utilizada a linguagem de programação *Python*, devido ao grande número de pacotes disponíveis que auxiliam na implementação de métodos de Otimização Topológica. Para a realização do processo de otimização, foi utilizado o pacote *TopOpt*[2], visto que esse tem uma documentação completa no que se refere a detalhes de implementação. Além disso, por ser um pacote *open source*, esse pôde ser modificado de acordo com as necessidades do projeto. As alterações realizadas neste trabalho consistiram na inserção das propriedades do material como um parâmetro na otimização, sobrecarga nas classes que implementam as condições de contorno e geração dos resultados intermediários de maneira a ser consumida pelo *Frontend*.

O *Backend* é responsável também por gerenciar, manter e enviar os resultados intermediários na ordem em que foram gerados para o *Frontend*. Os resultados intermediários foram armazenados seguindo a ordem de geração, utilizando uma estrutura de dados do tipo fila, a qual segue o padrão FIFO (*First In, First Out*).

A fim de garantir a correta execução do processo de otimização, o *Backend* realiza um conjunto de validações sobre os parâmetros utilizados, como por exemplo, a presença de suportes válidos, quantidade mínima de forças, posicionamento e dimensão das condições de contorno. No caso de erros, são geradas mensagens que são enviadas para o *Frontend*, o qual informa as correções necessárias. Na Figura 18, são apresentados alguns exemplos das validações realizadas.

Figura 18 – Exemplos de validações realizadas no *Backend*



Fonte: O Autor (2023).

[2] https://pytopopt.readthedocs.io

A comunicação entre *Backend* e *Frontend* é feita por meio de uma API (*Application Programming Interface*), que disponibiliza três métodos, os quais realizam: o início da otimização, a consulta dos resultados e a finalização do processo. A API foi implementada utilizando o pacote *Flask*[3], o qual é de fácil utilização e não necessita de arquivos de configuração.

3.3.2 Implementação do *Frontend*

O *Frontend* é executado diretamente em um navegador *Web*. Para o desenvolvimento foi utilizada a linguagem de programação *TypeScript*[4], que consiste em um superconjunto da linguagem *JavaScript*. Utilizou-se também a biblioteca *React.js* para o desenvolvimento dos componentes gráficos, com suas respectivas funções e estilos. Esses componentes foram reutilizados nas diferentes partes da aplicação.

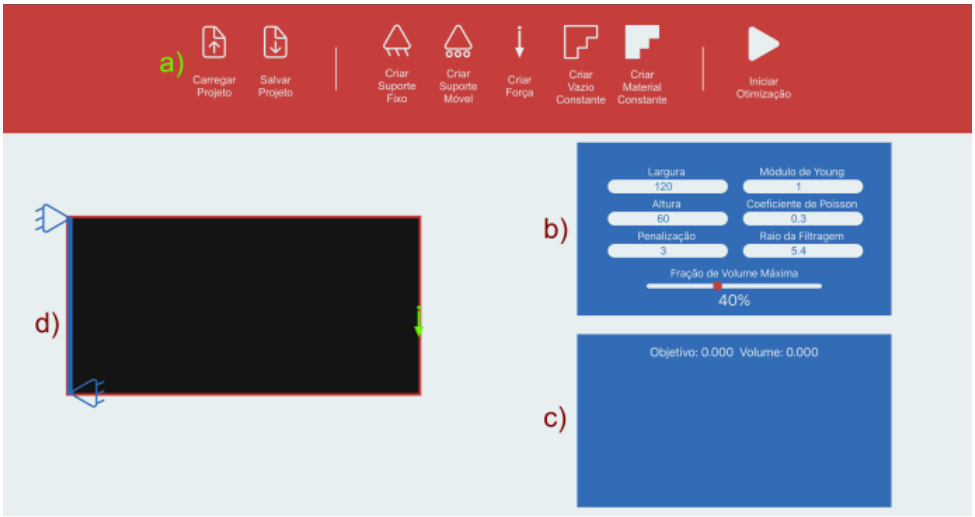
Optou-se pela utilização da linguagem *TypeScript* devido a grande quantidade de bibliotecas disponíveis. Entre essas bibliotecas, utilizou-se a *D3.js*[5], que auxilia no desenvolvimento de aplicações iterativas em ambientes *Web*. Nesta, os componentes visuais são representados por meio de imagens no formato *Scalable Vector Graphics* (SVG), o qual representa imagens no formato vetorial, possibilitando o escalonamento das figuras sem perda de definição. A biblioteca *D3.js* foi utilizada para a representação gráfica do processo de otimização juntamente com o acompanhamento dos valores das variáveis de projeto, além do gerenciamento da interação do usuário com a aplicação, como por exemplo, tratamento de cliques e movimentos de arrasto.

Para a implementação das funcionalidades de salvar e carregar os projetos foram utilizados arquivos no formato *JavaScript Object Notation* (JSON). Neste caso, são armazenadas as posições, as dimensões e as cargas dos parâmetros da estrutura, bem como as restrições e propriedades do material. Um exemplo de um arquivo JSON com os parâmetros de um projeto pode ser encontrado no Apêndice B.

A interface da aplicação é apresentada na Figura 19. Na parte superior da tela, encontram-se os controles para a adição das condições de domínio (Figura 19a). Os parâmetros da otimização, como dimensões de domínio, propriedades dos materiais e penalização, são editados em um painel existente no lado direito superior (Figura 19b). As variáveis de projeto podem ser visualizadas no quadro existente no lado direito inferior, junto com os gráficos de evolução do volume e da função objetivo (Figura 19c). Por fim, destaca-se na Figura 19d, o componente principal da aplicação, sendo esse responsável pela apresentação da estrutura otimizada e as condições de contorno.

[3] https://flask.palletsprojects.com/en/3.0.x/
[4] https://www.typescriptlang.org
[5] https://d3js.org

Figura 19 – Tela inicial do *Frontend*

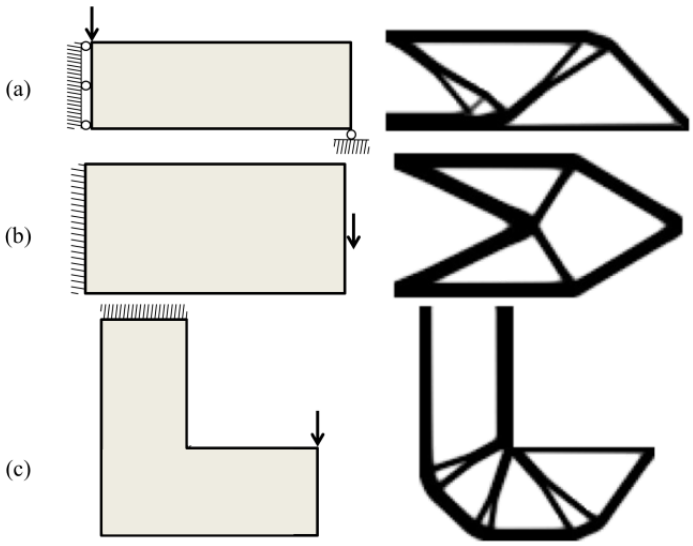


Fonte: O Autor (2023).

3.4 ESTRUTURAS DE TESTE

Dentro da área da OT existem problemas clássicos, que frequentemente são utilizados como estudo de caso e testes. Entre esses, foram selecionados três estruturas: viga MBB (*Messerschmitt-Bolkow-Blohm*), representada na Figura 20a), viga suspensa, representada na Figura 20b), e suporte em L, representado na Figura 20c). Nas próximas seções são apresentados os resultados obtidos utilizando essas estruturas. Os resultados são comparados com os resultados obtidos utilizando a implementação desenvolvida no trabalho de Sigmund (2001). Este trabalho apresenta uma implementação educacional, em linguagem Matlab, do processo de Otimização Topológica. Em todos os testes foi utilizada uma penalização de 3 e um raio de filtragem de $5{,}4$.

Figura 20 – Estruturas utilizadas para testes

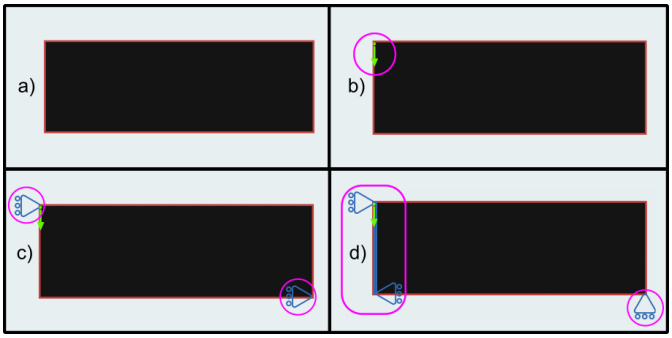


Fonte: Adaptado de Biyikli e To (2015).

3.4.1 Viga MBB

A viga MBB é um tipo de estrutura com dois apoios, um em cada extremidade. Essa pode apresentar diferentes dimensões e materiais, sendo muito comum em estruturas como pontes e edifícios. Na Figura 21, tem-se o processo de criação da estrutura para a inicialização da otimização. Inicialmente, a altura da estrutura é redimensionada para 40 unidades (Figura 21a). Em seguida, é adicionado uma força no canto superior da geometria, como destacado na Figura 21b). Dois suportes móveis são adicionados à estrutura, em cantos opostos, conforme destacado na Figura 21c). Por fim, a altura do suporte alinhado à esquerda é redimensionada e a direção do suporte inferior direto é trocada, como destacado na Figura 21d).

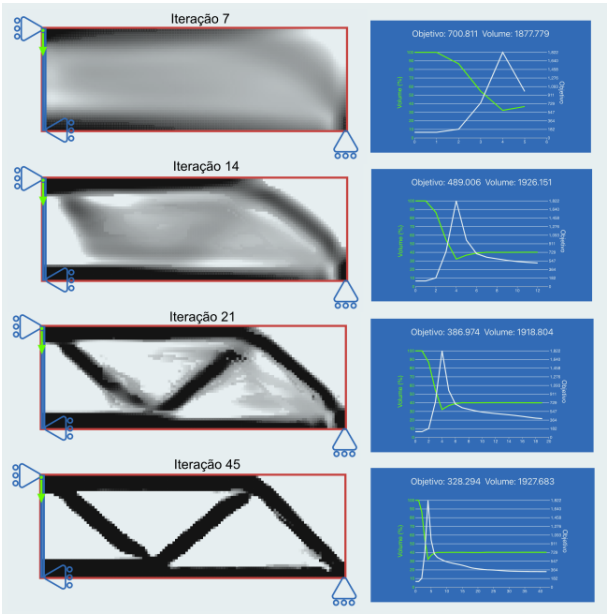
Figura 21 – Configurações iniciais do projeto da viga MBB



Fonte: O Autor (2023).

A Figura 22 apresenta os resultados intermediários do processo de otimização da estrutura de viga MBB. Nos gráficos do lado direito, é possível visualizar o processo de minimização da função custo.

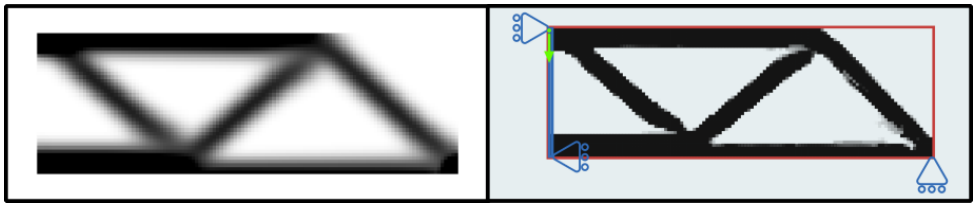
Figura 22 – Iterações do processo de otimização topológica da viga MBB



Fonte: O Autor (2023).

Na Figura 23 é apresentado um comparativo da estrutura final obtida, com a estrutura obtida utilizando a implementação de Sigmund (2001). Como pode ser observado, as estruturas resultantes são muito similares. No que se refere aos volumes, obteve-se um volume de $1927{,}683$, que é coerente com o valor de $1920{,}128$ que foi obtido a partir da implementação de Sigmund (2001).

Figura 23 – Resultado da viga MBB

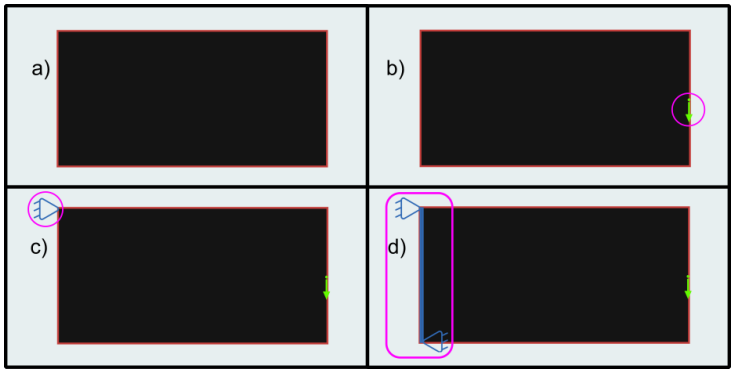


Fonte: O Autor (2023).

3.4.2 Viga suspensa

A viga suspensa é uma estrutura que envolve uma viga apoiada em uma extremidade e suspensa na outra, sendo utilizada em projetos de guindastes, guinchos, etc. Neste caso, foram realizadas simulações utilizando dimensões de 120×60 unidades (Figura 24a). Para a configuração do projeto, é aplicada uma força no centro da extremidade direita da geometria, conforme destacado na Figura 24b). Após, um suporte fixo é adicionado no canto superior esquerdo da estrutura, conforme pode ser observado na Figura 24c). Posteriormente, esse é redimensionado conforme a altura da estrutura, como pode ser visualizado na Figura 24d).

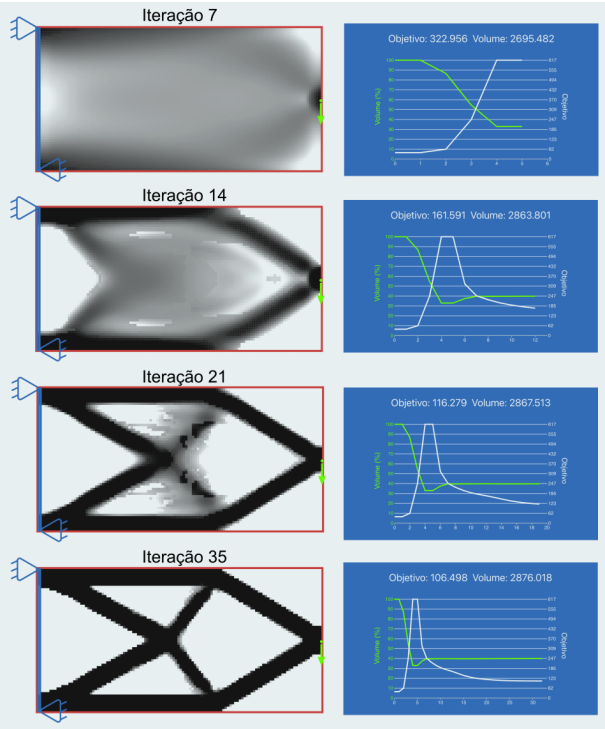
Figura 24 – Configurações iniciais do projeto da viga suspensa



Fonte: O Autor (2023).

Na Figura 25 é possível visualizar algumas iterações do processo de otimização topológica. Além disso, é possível visualizar os gráficos que demostram a minimização da função custo.

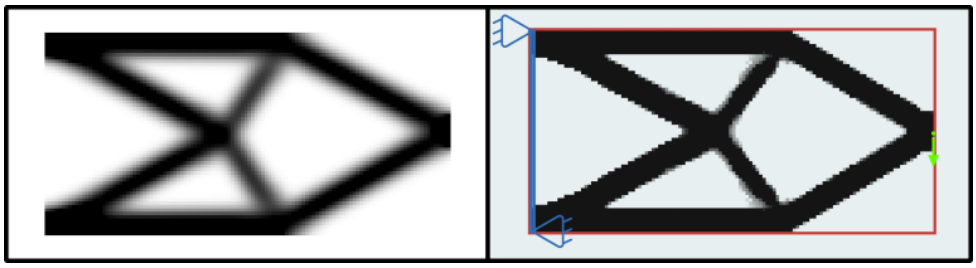
Figura 25 – Iterações do processo de otimização topológica da viga suspensa



Fonte: O Autor (2023).

Na Figura 26 é apresentado um comparativo das estruturas finais obtidas. Como pode ser observado, essas são similares. Neste caso, o volume obtido foi de $2876{,}018$, que encontra-se de acordo com o volume de $2879{,}533$, obtido por meio da implementação de Sigmund (2001).

Figura 26 – Resultado da viga suspensa



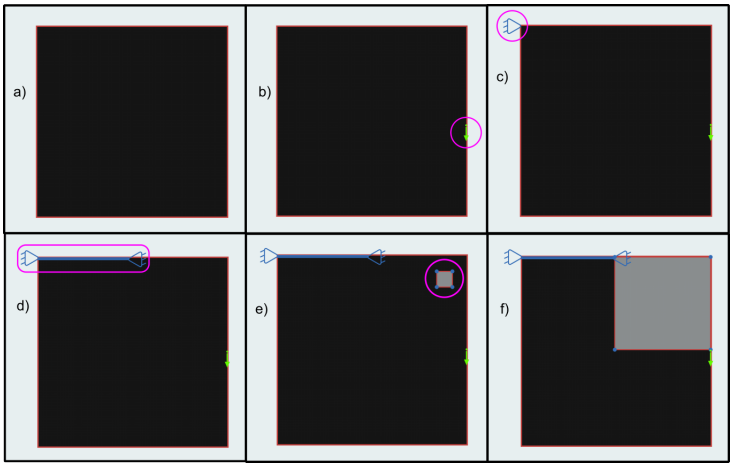
Fonte: O Autor (2023).

3.4.3 Suporte em L

O projeto do suporte em L, além do suporte em um parte do limite superior da estrutura, contém uma região de vazio no canto superior direito. Essa estrutura é utilizada como suporte em prateleiras, estruturas de fixação em construção, suportes de estantes, entre outros. Na Figura 27 tem-se os passos utilizados para criação da configuração inicial. Primeiramente, a altura da estrutura foi redimensionada de 60 para 120 unidades (Figura 27a). Após, é adicionada uma carga no centro da extremidade direita da estrutura, conforme destacado na Figura 27b). Um

suporte fixo é adicionado no canto superior esquerdo, como destacado na Figura 27c), e em seguida esse redimensionado, cobrindo a metade da largura do domínio (Figura 27d). Por fim, foi adicionado uma região de vazio (Figura 27e), sendo que essa foi reposicionada e, posteriormente, redimensionada, como pode ser observado na Figura 27f).

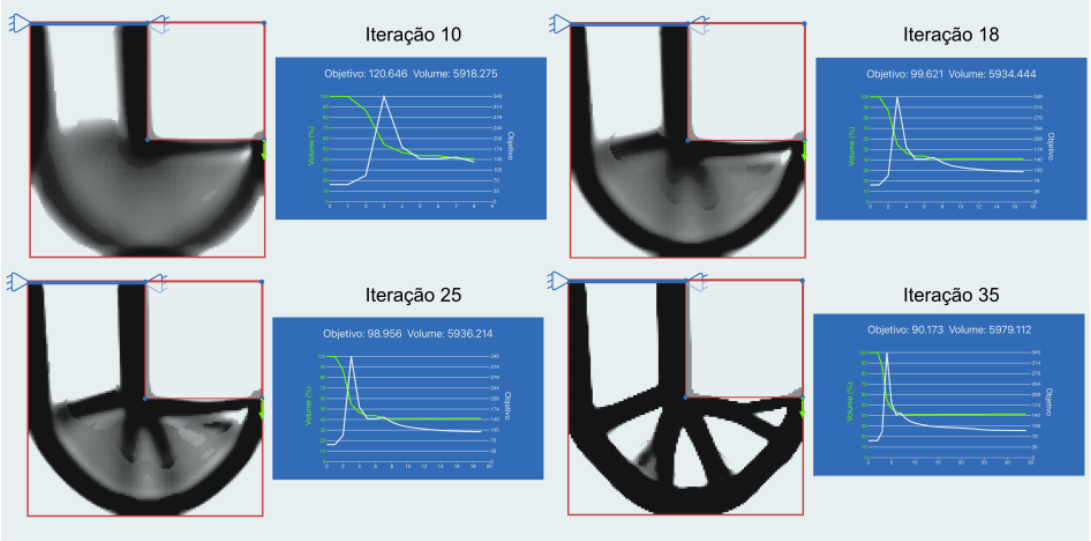
Figura 27 – Configurações iniciais do projeto do suporte em L



Fonte: O Autor (2023).

Na Figura 28 são apresentados as estruturas intermediárias obtidas, durante o processo de otimização. Além disso, é possível visualizar os respectivos gráficos de minimização da função custo.

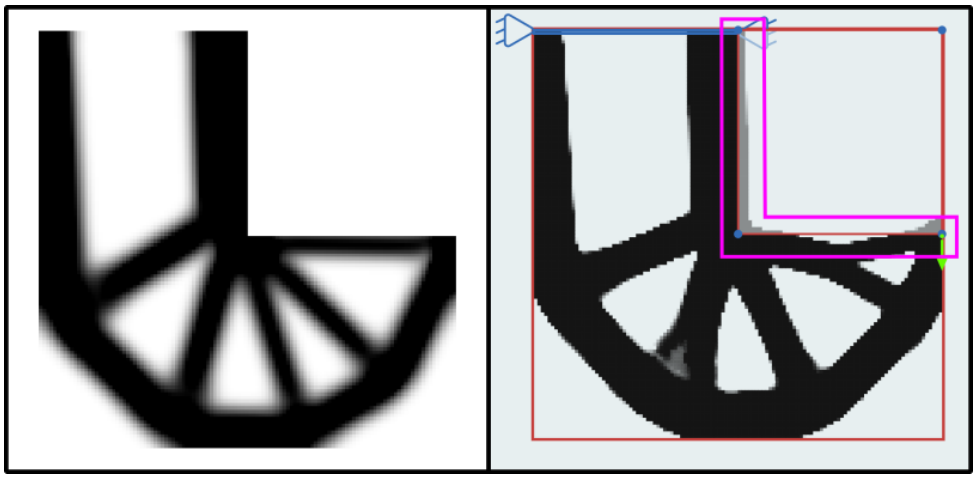
Figura 28 – Iterações do processo de otimização topológica do suporte em L



Fonte: O Autor (2023).

Na Figura 29, tem-se um comparativo entre as estruturas obtidas. Como pode ser observado, existem pequenas diferenças nos limites da peça juntos às bordas da região de vazio. A otimização realizada pela implementação de Sigmund (2001), apresentou um recorte reto. Já a otimização realizada neste trabalho resultou em limites irregulares. Neste caso, o volume obtido por meio da implementação desenvolvida neste trabalho foi de $5979{,}112$, que é 3% superior ao volume de $5762{,}642$, obtido por meio da implementação de Sigmund (2001). As diferenças gráficas e numéricas devem ser analisadas em trabalhos futuros.

Figura 29 – Resultado do suporte em L



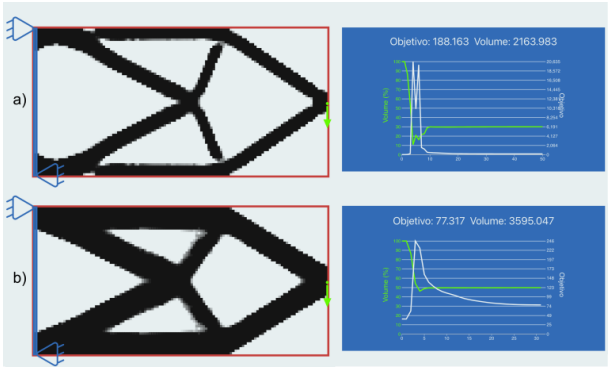
Fonte: O Autor (2023).

3.5 RESULTADOS VARIANDO OS PARÂMETROS DE CÁLCULO

A aplicação desenvolvida possibilita a variação dos parâmetros de cálculo, funcionalidade que não se encontra presente nos *softwares* analisados. Os parâmetros que podem ser modicados são os valores de penalizações, raios de filtragens, frações volumétricas máximas e propriedades do material. Essas modificações resultam em densidades diferentes para cada elemento e da matriz de rigidez, obtendo-se por fim, diferentes valores da função objetivo. Para analisar os efeitos desses parâmetros foi utilizada a estrutura de viga suspensa, descrita na Seção 3.4.2. Neste caso, espera-se que os resultados apresentem mudanças estruturais (topológicas ou de forma) e/ou numéricas.

A Figura 30 apresenta os resultados obtidos, utilizando diferentes valores para a fração de volume máxima. Neste caso, foram realizados testes utilizando valores de frações volumétricas de 30% (Figura 30a) e 50% (Figura 30b). Como pode ser observado, a geometria resultante apresentou alteração em seu contorno e espessuras, mantendo suas características topológicas. Os volumes finais obtidos para as estruturas foram $2163{,}983$ e $3594{,}047$. Esses valores estão de acordo com as respectivas frações do volume em relação ao volume máximo de 7200.

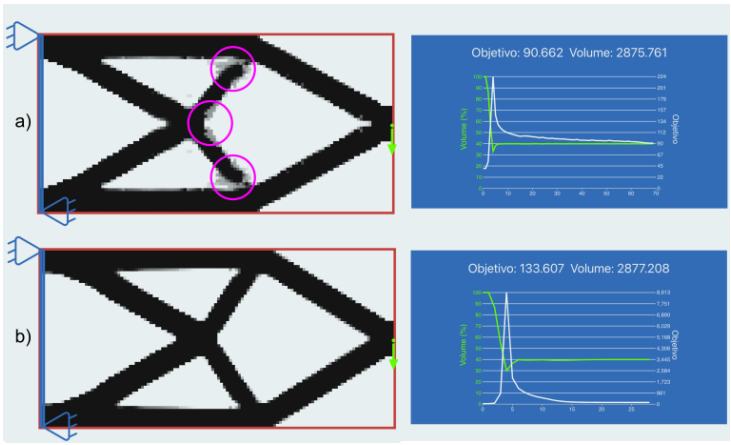
Figura 30 – Comparativo dos resultados com diferentes restrições de volume



Fonte: O Autor (2023).

A partir de alterações nos valores de penalização é possível aumentar ou diminuir as regiões de densidade intermediária. Neste caso, foram realizados testes, utilizando dois valores distintos de penalizações, sendo eles de 2 (Figura 31a) e de 5 (Figura 31b). A otimização realizada com um número menor de penalização, gera um número maior de regiões com densidades intermediárias (em destaque na Figura 31a). Entretanto, a utilização de valores de penalizações maiores, provoca a remoção destas áreas, como poder ser observado na Figura 31b).

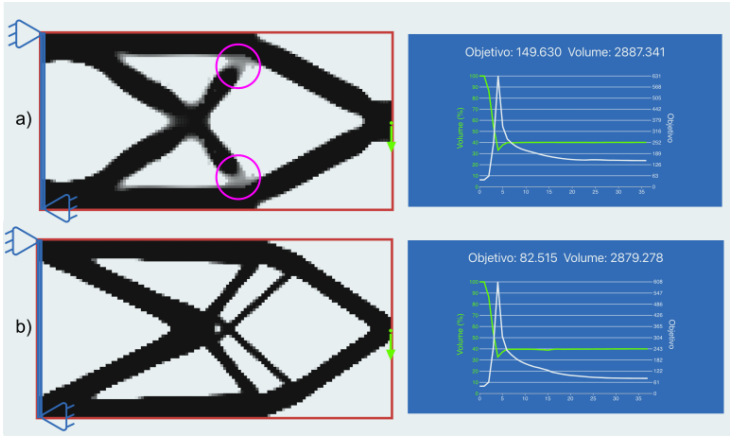
Figura 31 – Comparativo dos resultados com diferentes penalizações



Fonte: O Autor (2023).

A utilização do raio de filtragem possibilita a mitigação da instabilidade de tabuleiro na estrutura. Na Figura 32 são apresentados os resultados obtidos utilizando um raio de filtragem de 9 e um de 1,5. A Figura 32a) demonstra a utilização de um filtro maior, gerando uma concentração de densidades intermediárias na área destacada. A utilização de um raio menor provocou alterações topológicas na estrutura, criando perfurações adicionais, conforme pode ser observado na Figura 32b).

Figura 32 – Comparativo dos resultados com diferentes raios de filtragem

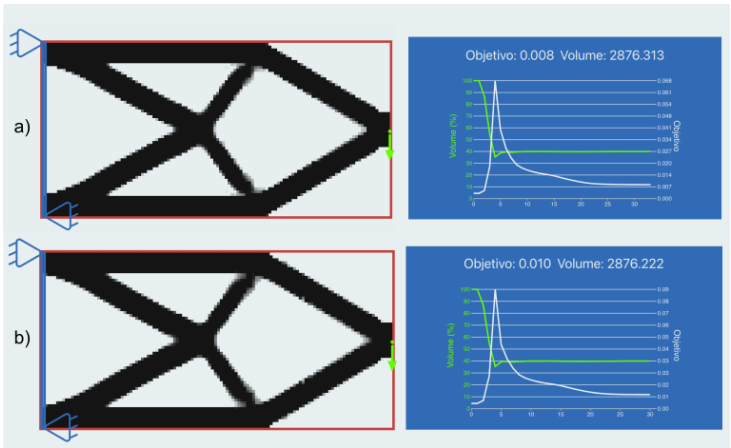


Fonte: O Autor (2023).

3.5.1 Resultados com diferentes materiais

Diferentes materiais são caracterizados por propriedades que representam a sua rigidez e comportamentos quando tensões são aplicadas. A Figura 33 apresenta resultados obtidos utilizando as propriedades do cobre (Figura 33a) e do titânio (Figura 33b). Os valores utilizados para o módulo de Young e coeficiente de Poisson do cobre foram de 115 GPa e $0{,}33$, respectivamente (ASM International, 1990). Para o titânio, considerou-se um módulo de Young de $102{,}7$ GPa e um coeficiente de Poisson de $0{,}34$ (ASM International, 1990).

Figura 33 – Resultados obtidos com propriedades materiais de cobre e titânio



Fonte: O Autor (2023).

Ambos os materiais apresentam uma maior rigidez em comparação com o material arbitrário apresentado na Seção 3.4, obtendo-se valores mais baixos para a função funções objetivo. No caso do material arbitrário, a função objetivo final atingiu $106{,}498$, enquanto para o cobre e o titânio, os valores obtidos foram $0{,}008$ e $0{,}010$, respectivamente. Já os resultados estruturais obtidos são equivalentes aos apresentados na Seção 3.4.2.

4 CONSIDERAÇÕES FINAIS

Neste trabalho foi desenvolvida uma aplicação *Web* para o ensino de métodos de Otimização Topológica. A arquitetura utilizada foi a *Backend For Frontend*. O *Frontend* foi desenvolvido para um ambiente *Web*, não necessitando de um instalação prévia e permitindo que o dispositivo do usuário tenha poucos recursos computacionais.

Para a implementação do *Backend* foi utilizada a linguagem de programação *Python*, visto que essa contém diversos pacotes que auxiliam no desenvolvimento de aplicações de Otimização Topológica. Entre esses pacotes optou-se pela utilização do *TopOpt*, que aborda o processo de otimização de maneira simplificada. Para o processo de otimização foi o escolhido o método SIMP, que é amplamente utilizado em aplicações de OT, além de ser o método implementado pelo pacote *TopOpt*. Além disso, o pacote pôde ser alterado, visto que esse é *open source*. Para esse trabalho foram adicionadas a possibilidade de edição das propriedades do material, além de modificações que possibilitam a visualização dos resultados intermediários durante o processo de otimização.

O *Frontend* foi implementado utilizando a linguagem *TypeScript* e a biblioteca *React.js*. A linguagem *TypeScript* possibilita a utilização de diversas bibliotecas, sendo que neste trabalho foi utilizada a biblioteca *D3.js*, a qual possibilita a captura das ações do usuário e a manipulação de componentes gráficos. Optou-se pela utilização da biblioteca *D3.js*, pois essa possui uma vasta documentação e uma comunidade de desenvolvimento ativa. A comunicação entre o *Backend* e o *Frontend* é realizada utilizando uma API desenvolvida com o pacote *Flask*. Optou-se pela utilização do pacote *Flask*, pois esse é de fácil utilização e não necessita de arquivos de configurações.

Para os testes foram utilizadas estruturas clássicas como, por exemplo, uma viga MBB, uma viga suspensa e uma estrutura de suporte em L. Os resultados obtidos foram comparados com os resultados obtidos com a implementação desenvolvida por Sigmund (2001), sendo que os resultados obtidos foram equivalentes.

A implementação desenvolvida neste trabalho, possui ainda suporte para a modificação dos valores de penalização e raio de filtragem, além de permitir a edição das propriedades dos materiais, mais especificamente, o módulo de Young e coeficiente de Poisson. Essas funcionalidades não se encontram presentes nos *softwares* analisados e os resultados obtidos utilizando essas funcionalidades satisfazem os comportamentos esperados.

4.1 TRABALHOS FUTUROS

Como sugestão para trabalhos futuros, tem-se:

- A implementação de uma abordagem híbrida utilizando método DSC, possibilitando uma otimização de forma após a OT. Essa funcionalidade não foi implementada devido ao limite de tempo para o desenvolvimento deste trabalho.
- Otimização de estruturas em três dimensões. Essa funcionalidade possibilitaria o uso externo dos resultados para a prototipação e fabricação. No entanto, essa também não foi implementada devido a restrições de tempo.
- Realização de uma avaliação da *interface* e da usabilidade da mesma.
- Revisão dos resultados numéricos e gráficos obtidos. Como apresentado, a implementação desenvolvida neste trabalho apresentou diferenças na estrutura final do suporte em L. Desta forma, torna-se necessário uma revisão mais cuidadosa desses resultados.

REFERÊNCIAS

AAGE, N. *et al.* Interactive topology optimization on hand-held devices. **Structural and Multidisciplinary Optimization**, Springer Science and Business Media LLC, v. 47, n. 1, p. 1–6, 2012.

ANDREASEN, C.; ELINGAARD, M.; AAGE, N. Level set topology and shape optimization by density methods using cut elements with length scale control. **Structural and Multidisciplinary Optimization**, Springer, v. 62, p. 685–707, 2020.

ANDREASSEN, E. *et al.* Efficient topology optimization in MATLAB using 88 lines of code. **Structural and Multidisciplinary Optimization**, Springer Science and Business Media LLC, v. 43, n. 1, p. 1–16, 2010.

ARANDA, E.; BELLIDO, J. C.; DONOSO, A. Toptimiz3d: A topology optimization software using unstructured meshes. **Advances in Engineering Software**, v. 148, p. 102875, 2020. ISSN 0965-9978.

ARORA, J. **Introduction to Optimum Design**. 4. ed. [S.l.]: Elsevier Science, 2016. ISBN 9780128008065.

ASM International. **ASM handbook vol. 2**. [S.l.]: ASM International, 1990. (ASM Handbooks). ISBN 9780871703781.

BAILLIE, C. **Engineering and Society: Working Towards Social Justice Part I: Engineering and Society**. [S.l.]: Springer International Publishing, 2009.

BENDSØE, M.; KIKUCHI, N. Generating optimal topologies in structural design using a homogenization method. **Computer Methods in Applied Mechanics and Engineering**, v. 71, n. 2, p. 197–224, 1988. ISSN 0045-7825.

BENDSØE, M.; SIGMUND, O. **Topology Optimization: Theory, Methods, and Applications**. 2. ed. [S.l.]: Springer, 2004. ISBN 9783662050866.

BIYIKLI, E.; TO, A. C. Proportional topology optimization: A new non-sensitivity method for solving stress constrained and minimum compliance problems and its implementation in MATLAB. **PLOS ONE**, Public Library of Science, v. 10, n. 12, p. 1–23, 2015.

COCIAN, L. **Introdução à Engenharia**. [S.l.]: Bookman Editora, 2016. ISBN 9788582604182.

COE, C. Problems on maxima and minima. **The American Mathematical Monthly**, Taylor & Francis, v. 49, n. 1, p. 33–37, 1942.

DANTZIG, G. **Linear Programming and Extensions**. Princeton: Princeton University Press, 1963. ISBN 9781400884179.

DEATON, J. D.; GRANDHI, R. V. A survey of structural and multidisciplinary continuum topology optimization: post 2000. **Structural and Multidisciplinary Optimization**, Springer Science and Business Media LLC, v. 49, n. 1, p. 1–38, jul. 2013.

DIJK, N. P. van *et al.* Level-set methods for structural topology optimization: a review. **Structural and Multidisciplinary Optimization**, Springer Science and Business Media LLC, v. 48, n. 3, p. 437–472, 2013.

DUNNING, P. D.; KIM, H. A. Introducing the sequential linear programming level-set method for topology optimization. **Structural and Multidisciplinary Optimization**, Springer Science and Business Media LLC, v. 51, n. 3, p. 631–643, 2014.

FLOUDAS, P. E. C. A. **Encyclopedia of Optimization**. 2nd. ed. [S.l.]: Springer, 2009. (Springer reference). ISBN 9780387747590.

HAFTKA, R.; GÜRDAL, Z. **Elements of structural optimization**. 3. ed. [S.l.]: Kluwer Academic Publishers, 1992. (Solid mechanics and its applications, v. 11). ISBN 0792315049.

MUNK, D. J. A bidirectional evolutionary structural optimization algorithm for mass minimization with multiple structural constraints. **International journal for numerical methods in engineering**, Wiley Subscription Services, Inc, Bognor Regis, v. 118, n. 2, p. 93–120, 2019. ISSN 0029-5981.

NGUYEN, T. T. *et al.* Efficient hybrid topology and shape optimization combining implicit and explicit design representations. **Structural and Multidisciplinary Optimization**, Springer Science and Business Media LLC, v. 62, n. 3, p. 1061–1069, 2020.

NOBEL-JØRGENSEN, M. *et al.* 3d interactive topology optimization on hand-held devices. **Structural and Multidisciplinary Optimization**, Springer Science and Business Media LLC, v. 51, n. 6, p. 1385–1391, 2014.

OLHOFF, N.; TAYLOR, J. E. On Structural Optimization. **Journal of Applied Mechanics**, v. 50, n. 4b, p. 1139–1151, 1983. ISSN 0021-8936.

OLIVEIRA, C. J. de *et al.* Structural topology optimization as a teaching tool in the architecture. **Revista de Ensino de Engenharia**, v. 37, n. 3, 2019.

PERINI, G. **Estudo de caso: aplicação de otimização topológica no desenvolvimento de um protótipo de cubo de roda**. 129 f. Dissertação (Mestrado Profissional em Engenharia Mecânica) — Programa de Pós-Graduação em Engenharia Mecânica, Universidade de Caxias do Sul, Caxias do Sul, 2013.

QUERIN, O.; STEVEN, G.; XIE, Y. Evolutionary structural optimisation (ESO) using a bidirectional algorithm. **Engineering Computations**, Emerald, v. 15, n. 8, p. 1031–1048, 1998.

REDDY, S. N. *et al.* Topology optimization software for additive manufacturing: A review of current capabilities and a real-world example. In: AMERICAN SOCIETY OF MECHANICAL ENGINEERS (ASME). **ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, IDETC/CIE 2016**. [S.l.], 2016.

REMOUCHAMPS, A. *et al.* Application of a bi-level scheme including topology optimization to the design of an aircraft pylon. **Structural and Multidisciplinary Optimization**, Springer, v. 44, p. 739–750, 2011.

SIGMUND, O. A 99 line topology optimization code written in Matlab. **Structural and Multidisciplinary Optimization**, Springer Science and Business Media LLC, v. 21, n. 2, p. 120–127, 2001.

SIGMUND, O.; PETERSSON, J. Numerical instabilities in topology optimization: a survey on procedures dealing with checkerboards, mesh-dependencies and local minima. **Structural optimization**, Springer, v. 16, p. 68–75, 1998.

TSAI, J.-F. *et al.* Optimization theory, methods, and applications in engineering. **Mathematical Problems in Engineering**, Hindawi, v. 2012, 2012.

TYFLOPOULOS, E.; HASKINS, C.; STEINERT, M. Topology-optimization-based learning: A powerful teaching and learning framework under the prism of the CDIO approach. **Education Sciences**, MDPI, v. 11, n. 7, p. 348, 2021.

WANG, M. W. S. Structural shape and topology optimization using an implicit free boundary parametrization method. **Computer Modeling in Engineering & Sciences**, v. 13, n. 2, p. 119–148, 2006. ISSN 1526-1506.

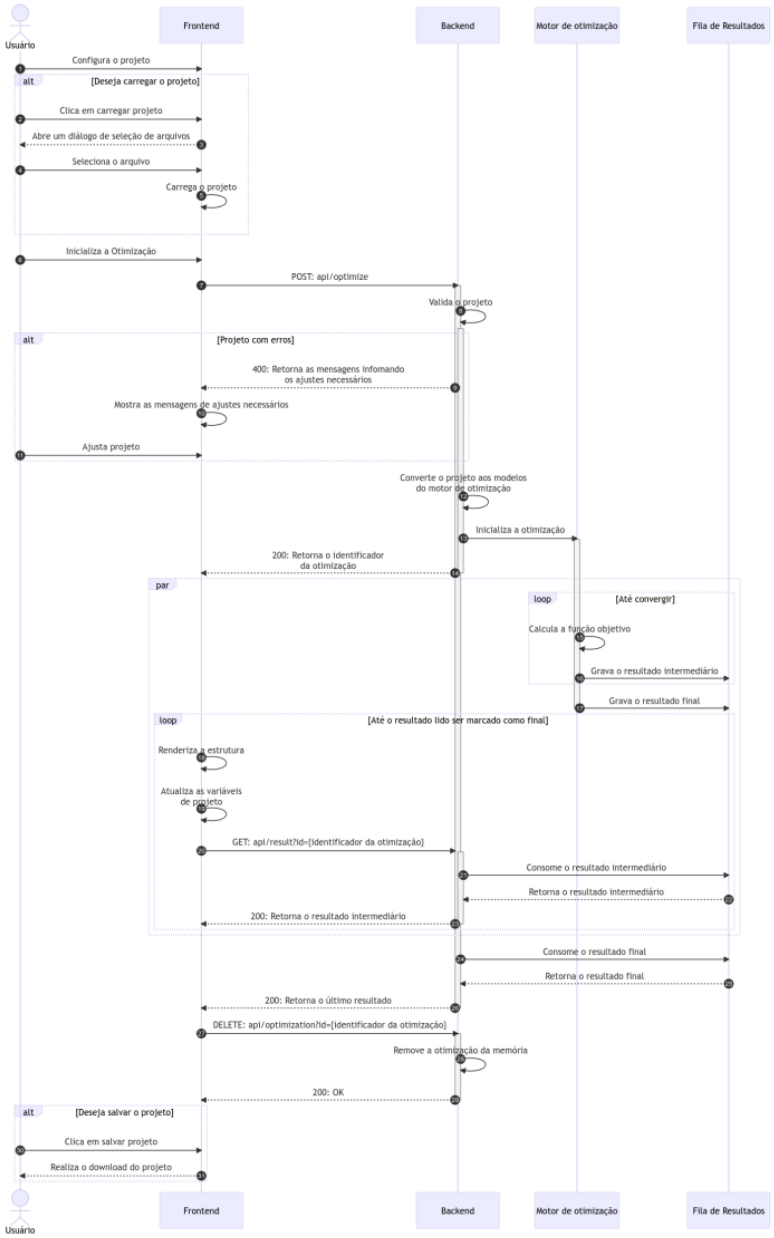
WEI, P. *et al.* An 88-line MATLAB code for the parameterized level set method based topology optimization using radial basis functions. **Structural and Multidisciplinary Optimization**, Springer Science and Business Media LLC, v. 58, n. 2, p. 831–849, 2018.

XIE, Y.; STEVEN, G. A simple evolutionary procedure for structural optimization. **Computers & Structures**, Elsevier BV, v. 49, n. 5, p. 885–896, 1993.

APÊNDICE A – REPRESENTAÇÃO DO FLUXO DE EXECUÇÃO DA APLICAÇÃO DESENVOLVIDA

A Figura 34 ilustra o fluxo único de execução da aplicação desenvolvida. É demonstrada a execução do gerenciamento de resultados intermediários, conforme geração e consumo dos mesmos em paralelo, além das alternativas de interação e demonstração dos resultados de validação do projeto.

Figura 34 – Diagrama de sequência com o fluxo de execução da aplicação



Fonte: O Autor (2023).

APÊNDICE B – EXEMPLO DE UM PROJETO SALVO

O Algoritmo 1 apresenta o projeto do suporte em L no formato de arquivo JSON.

Algoritmo 1 – Exemplo de um projeto

```
{
  "domain": {
    "materialProperties": {
      "poisson": 0.3,
      "young": 1
    },
    "dimensions": {
      "width": 120,
      "height": 120
    },
    "volumeFraction": 0.4
  },
  "boundaryConditions": {
    "supports": [
      {
        "position": {
          "x": 0,
          "y": 0
        },
        "type": 1,
        "id": 0,
        "dimensions": { "width": 60, "height": 1 }
      }
    ],
    "forces": [
      {
        "load": -1,
        "orientation": 1,
        "position": {
          "x": 120,
          "y": 60
        },
        "id": 0
      }
    ],
    "constantRegions": [
      {
        "position": { "x": 60, "y": 0 },
        "dimensions": { "width": 60, "height": 60 },
        "type": 0,
        "id": 0
      }
    ]
  },
  "penalization": 3,
  "filterRadius": 5.4
}
```

Fonte: O Autor (2023).